Proceedings of the

Computer Vision Winter Workshop 2014

Zuzana Kúkelová and Jan Heller (eds.)

ISBN 978-80-260-5641-6

CVWW 2014

Proceedings of the 19th Computer Vision Winter Workshop

February 3–5, 2014
Křtiny, Czech Republic

Zuzana Kúkelová and Jan Heller (eds.)

Czech Society for Cybernetics and Informatics
(Czech Pattern Recognition Society group)

Prague 2014

Proceedings of the Computer Vision Winter Workshop 2014

Editor

Zuzana Kúkelová and Jan Heller
Center for Machine Perception
Department of Cybernetics
Czech Technical University in Prague
Technická 2, 166 27 Prague 6, Czech Republic
tel.: +420 224 357 637, fax: +420 224 357 385
mailto:kukelova@cmp.felk.cvut.cz
mailto:hellej1@cmp.felk.cvut.cz

Publisher

Czech Society for Cybernetics and Informatics
(Czech Pattern Recognition Society group)
Pod Vodárenskou věží 4, 182 00 Prague 8,
Czech Republic

Organisation

The 19th CVWW 2014 was organized jointly by the Czech Society for Cybernetics and Informatics, and the Center of Machine Perception at CTU in Prague.

Local Organising Committee:
Eva Matysková, Center for Machine Perception

Proc. of the Computer Vision Winter Workshop 2014.
Zuzana Kúkelová and Jan Heller (editors).
Czech Society for Cybernetics and Informatics.
Prague, February 2014. 127 pages.
Design, composition, ConTEXt and LaTeX styles by Vít Zýka (http://typokvitek.com)
TEX Gyre fonts.
Electronic version is available at http://cmp.felk.cvut.cz/cvww2014.

Workshop Chairs

Zuzana Kúkelová and Jan Heller, Czech Technical University in Prague

Invited speakers

Ivan Laptev, INRIA Helmut Pottmann, King Abdullah University of Science and Technology

Programme Committee

Austrian Institute of Technology
- Beleznai, Csaba
- Mičušík, Branislav
- Pflugfelder, Roman

CMP, CTU Prague
- Čech, Jan
- Chum, Ondřej
- Drbohlav, Ondřej
- Flach, Boris
- Franc, Vojtěch
- Heller, Jan
- Hlaváč, Václav
- Kúkelová, Zuzana
- Matas, Jiří
- Matoušek, Martin
- Průša, Daniel
- Šára, Radim
- Svoboda, Tomáš

ETH Zurich
- Havlena, Michal

Google Inc.
- Wendel, Andreas

Institute of Science and Technology
- Sharmanska, Viktoriia

TU Graz
- Bischof, Horst
- Pock, Tomas

TU Wien
- Gelautz, Margrit
- Haxhimusa, Yll
- Kropatsch, Walter
- Sablatnig, Robert

University of Ljubljana
- Čehovin, Luka
- Kovačič, Stanislav
- Kristan, Matej
- Mandeljc, Rok
- Perš, Janez

Preface

The Computer Vision Winter Workshop is an annual meeting formed around four groups: the Pattern Recognition and Image Processing Group, TU Vienna, the Computer Vision Lab of the University of Ljubljana, the Institute for Computer Graphics and Vision, TU Graz, and the Center for Machine Perception, CTU Prague. The main goals of the workshop are to communicate fresh ideas within the four groups, and to provide conference experience to PhD students. Nevertheless, the workshop is open to everyone.

The 19th CVWW 2014 was organized jointly by the Czech Society for Cybernetics and Informatics and the Center of Machine Perception at CTU Prague. It was held in Křtiny, Czech Republic, February 3–5, 2014.

Besides papers selected by the review process, two invited talks were also included. We would like to express our thanks to Ivan Laptev, INRIA, and Helmut Pottmann, KAUST, for their invited contributions.

We extend our thanks to the members of the program committee for their time and the valuable feedback in their reviews.

Last but not least, we would like to thank the following people that made the organization of the workshop easier: Vít Zýka (proceedings), Daniel Večerka (software support). We thank to all senior members of the CMP group for their advice. Very special thanks belong to Eva Matysková for her immense contribution to the organisation of the workshop.

Zuzana Kúkelová
Jan Heller
Prague, January 2014

Contents

A Novel Fuzzy C-Means Based Defuzzification Approach with an Adapted Minkowski Distance
Ali Abder-Rahman, Albouy-Kissi Adélaïde, Vacavant Antoine, Grand-brochier Manuel, Boire Jean-Yves 6

jSLIC: superpixels in ImageJ
Borovec Jiří, Kybic Jan 14

A bi-level view of inpainting – based image compression
Chen Yunjin, Ranftl René, Pock Tomas 19

Violent Scenes Detection based on Automatically-generated Mid-level Violent Concepts
Goto Shinichi, Aoki Terumasa 27

Over-Segmentation of 3D Medical Image Volumes based on Monogenic Cues
Holzer Markus, Donner Rene 35

Reeb graph based examination of root development
Janusch Ines, Kropatsch Walter, Busch Wolfgang 43

Pedestrian Recognition and Localisation in Image Sequences as Bayesian Inference
Klinger Tobias, Rottensteiner Franz, Heipke Christian 51

The VOT2013 challenge: overview and additional results
Kristan Matej, Pflugfelder Roman, Leonardis Ales, Matas Jiri, Porikli Fatih, Cehovin Luka, Nebehay Georg, Fernandez Gustavo, Vojir Tomaš 59

A Few Things One Should Know About Feature Extraction, Description and Matching
Lenc Karel, Matas Jiri, Mishkin Dmytro 67

Print Localization on Transparent Pharmaceutical Capsules
Mehle Andraž, Bukovec Marko, Likar Bostjan, Pernuš Franjo, Tomaževič Dejan 75

Visual Recognition and Fault Detection for Power Line Insulators
Oberweger Markus, Wendel Andreas, Bischof Horst 81

Detecting handwritten signatures in scanned documents
Ogul Hasan, Cuceloglu Ilkhan 89

Evaluation of performance of smart mobile devices in machine vision tasks
Skocaj Danijel, Tabernik Domen, Racki Domen, Hegedic Matjaz, Vrecko Alen, Kristan Matej 95

Combining Structural and Statistical Approach to Online Recognition of Handwritten Mathematical Formulas
Stria Jan, Prusa Daniel, Hlavac Vaclav 103

Using discriminative analysis for improving hierarchical compositional models
Tabernik Domen, Kristan Matej, Boben Marko, Leonardis Ales 110

Canonical Encoding of the Combinatorial Pyramid
Torres Fuensanta, Kropatsch Walter 118

Author Index

Computer Vision Winter Workshop 2014, Zuzana Kúkelová and Jan Heller (eds.)
Křtiny, Czech Republic, February 3–5
Czech Society for Cybernetics and Informatics

A Novel Fuzzy C-Means Based Defuzzification Approach with an Adapted Minkowski Distance

Abder-Rahman Ali, Adélaïde Albouy-Kissi, Antoine Vacavant,
Manuel Grand-brochier, Jean-Yves Boire

Clermont Université, Université d'Auvergne, ISIT, BP10448, F-63000 Clermont-Ferrand, France
CNRS, UMR6284, BP10448, F-63000 Clermont-Ferrand, France
abder-rahman.ali@etu.udamail.fr, adelaide.kissi@udamail.fr,
antoine.vacavant@udamail.fr, manuel.grand-brochier@udamail.fr,
j-yves.boire@udamail.fr

Abstract *In this paper, we present a novel defuzzification approach by feature distance minimization with an adapted Minkowski distance. The proposed approach aims at providing a segmented image through the defuzzification of the fuzzy partitions of the original image. A mathematical derivation of which best pixel would be added/removed to/from the alpha-region of the crisp set is described. And, an evaluation of the proposed approach is carried out.*

1 Introduction

A gray-scale object may be considered as a fuzzy set, where each pixel is defined with a membership function [1]. The membership values can be of real interest to conduct the segmentation of an image into regions of interest, especially that they are required in fuzzy clustering approaches. During this process, one should integrate more features that will represent these regions by their geometrical properties as shape, area, perimeter, *etc*. Amongst a whole set of *N* possibly informative measurements, feature selection aims at selecting a subset of *n* features from the given set of *N* measurements, where $n<N$ [2].

Preserving some relevant features of the original object in the fuzzy discrete representation aids in the recovering of a crisp object when defuzzifying that representation. Defuzzification by feature distance minimization achieves such preservation of the relevant features in the fuzzy discrete representation. In order to get an output object that resembles the original crisp object, the selection of features that will be included in the defuzzification should take into account their relevance (*i.e*. shape preservation and application), and how well is their preservation in fuzzification. Intuitively, points with high membership degrees to the fuzzy object should be included in its crisp representation, and those with low membership degrees should be assigned to the background. This can be achieved by using the membership degree values of the points as features in the distance measure [3]. Here, one can work on *similarity space* instead of feature space (*i.e*. for the purpose of clustering). Thus, if one can find a similarity measure derived from the object features which is considered appropriate for the problem domain, then a single number can capture the essential closeness of a given pair of objects, and any further analysis can be based only on those numbers [4]. The effectiveness of object recognition is highly dependent on the accurate identification of shapes of clusters. which are determined by the choice of the distance measure [5]. For example, the Euclidean distance is often used to reflect dissimilarity between two patterns and is known to work well when all clusters are spheroids or when all clusters are well separated [6]. The use of the Minkowski dissimilarity measure in the paper was due to its allowance of varying the assumptions of the shape of the clusters by varying the order *m*. The most often used value is $m = 2$ that assumes a circular cluster shape. Using $m = 1$ assumes that the clusters are in the shape of a (rotated) square in two dimensions or a diamond like shape in three or more dimensions. For $m = \infty$, the clusters are assumed to be in the form of a box with sides parallel to the axes [7].

In this paper, we present a novel defuzzification approach by feature distance minimization with an adapted Minkowski distance. The idea is based on the use of floating point search for finding the best crisp set from Fuzzy C-Means clustering using Minkowski distance.

The novelty of the proposed approach lies in the use of a new defuzzification that embeds an improved Minkowski distance, which takes into account both the membership degree values of the elements of the fuzzy set, and the spatial relationships.

The paper is organized as follows: Section 2 gives an overview of different dissimilarity measures, and the reasons behind using Minkowski distance. Section 3 introduces notations used in the paper, fuzzy and crisp sets, and Fuzzy C-Means. In Section 4, we describe the proposed approach, combining Fuzzy C-Means, the new defuzzification process, and an improved Minkowski distance. We evaluate the proposed approach in Section 5, and conclude the paper in Sec-

tion 6.

2 Dissimilarity Measures

Given two sequences of measurements from $X = \{x_i : i = 1, ..., n\} \in \mathbb{R}^n$ and $Y = \{y_i : i = 1, ..., n\} \in \mathbb{R}^n$ such that *n* is the size of the input image. The dissimilarity between *X* and *Y* is a measure that quantifies the independency between the sequences [8], For the purpose of this paper, we assume that *X* and *Y* represent the fuzzy set and the crisp set, respectively, provided that x_i and y_i represent the membership degrees and the pixel intensities, respectively.

The term *distance* is often used informally to refer to a dissimilarity measure *M* derived from the characteristics describing the objects (*i.e*. Euclidean distance) [9].

A *metric M(X,Y)* is considered a dissimilarity measure if a higher value is produced as corresponding values in *X* and *Y* become less dependent, and which satisfies the following for all $X \in \mathbb{R}^n$ and $Y \in \mathbb{R}^n$ [8]:

- Non-negativity: $M(X, Y) \geq 0$;
- Reflexivity: $M(X, Y) = 0$ *if an only if* $X = Y$;
- Symmetry: $M(X, Y) = M(Y, X)$;
- Triangle inequality: $M(X, Y) + M(Y, Z) \geq M(X, Z)$.

Thus, in order to measure dissimilarity, one of the parameters that can be used is *distance*. This category of measures is known as separability, divergence, or discrimination measures [9].

In clustering analysis, choosing the appropriate dissimilarity measure is required. The most commonly used measures in clustering analysis are: (1) Euclidean distance, (2) Manhattan distance, (3) Minkowski distance, and (4) Mahalanobis distance.

Euclidean distance $d_2(\mathbf{x}_i, \mathbf{x}_j)$, where $\mathbf{x}_i$ and $\mathbf{x}_j$ are p-dimensional features, with $p \in \mathbb{N}_+^*$ is the most popular measure of dissimilarity, and the most common distance metric used. It is the usual manner in which distance is measured in the real world [10], and is defined by [11]:

$$d_2(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{k=1}^{p} (\mathbf{x}_{ik} - \mathbf{x}_{jk})^2} \quad (1)$$

Euclidean distance works when each cluster has a shape of a hyper-sphere in space, but, has poor performance when the cluster has a shape of a hyper-ellipsoid [11].

The drawback of Euclidean distance is that it ignores the similarity between attributes, as each attribute is treated as totally different from all other attributes, and does not work well in high dimensions and for categorical variables [10]. In order to overcome this drawback, Gustafon and Kessel distance can be utilized, which is able to discriminate ellipsoidal cluster shapes [4].

Manhattan distance gets its name from the rectangular grid patterns of streets in midtown Manhattan, and is defined by [7]:

$$d_{Manhattan}(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{p} |\mathbf{x}_{ik} - \mathbf{x}_{jk}| \quad (2)$$

In some situations, this metric is more preferable to Euclidean distance, since the distance along each axis is not squared, and thus, a large difference in one dimension will not dominate the total distance [7].

Mahalanobis distance is the distance between an observation and the centre for each group in a p-dimensional space defined by *p* variables and their covariance. Thus, a small value of Mahalanobis distance increases the chance of an observation to be closer to the groups center, and the more likely it would be assigned to that group [14]. Mahalanobis distance is defined by [14]:

$$d_{Mahalanobis}(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T \sum^{-1} \mathbf{x}_i - \mathbf{x}_j} \quad (3)$$

where $\sum^{-1}$ is the inverse covariance matrix.

If there are two non-correlated variables, the Mahalanobis distance between the points of the variables in a 2D scatter plot is the same as the Euclidean distance [15].

Unlike most other distance measures, Mahalanobis distance is not dependent upon the scale on which the variables are measured since it is normalized [14].

Minkowski distance is a generalization of Euclidean and Manhattan distances [12], and is defined by [11]:

$$d_m(\mathbf{x}_i, \mathbf{x}_j) = \sqrt[m]{\sum_{k=1}^{p} |\mathbf{x}_{ik} - \mathbf{x}_{jk}|^m} \quad (4)$$

where *m* is a real number, such that $m \geq 1$. When $m = 1$, it represents the Manhattan distance, and when $m = 2$, it represents the Euclidean distance [13].

Minkowski distance provides a concise, parametric distance function that generalizes many of the distance functions used in the literature. The advantage of using this distance is that mathematical results can be shown for the whole class of distance functions, and the user can adapt the distance function to suit the needs of the application by modifying the Minkowski parameter *m* [12].

3 Background Notations

In this section, the notations of crisp sets, fuzzy sets, core of a fuzzy set, support of a fuzzy set, $\alpha - cut$ of a fuzzy, and Fuzzy C-Means will be explained (sets are usually denoted by upper case letters, and their members by lower case letters [16]).

A crisp set *A* in the universe of discourse *U* has a binary membership function, and thus, has no uncertainty. It is defined as a set of ordered pairs [17]:

$$A = \{(x, \phi_A(x)) \mid x \in U\} \quad (5)$$

where $\phi_A(x)$ is the binary membership function:$\phi_A(x) = 1$ if $x \in A$, and $\phi_A(x) = 0$ if $x \notin A$. $\phi_A(x) \in \{0, 1\}$ [17].

A fuzzy set $A^{\sim}$ in the universe of discourse *U* has a fuzzy membership function, in which fuzziness (uncertainty) exists. It is defined as a set of ordered pairs [17]:

$$A^{\sim} = \{(x, \mu_{A^{\sim}}(x)) \mid x \in U\} \quad (6)$$

where $\mu_{A^\sim}$ is the fuzzy membership function that maps *x* to a membership degree between 0 and 1. $\mu_{A^\sim}(x) \in [0,1]$ [17].

The *core* of a fuzzy set $A^\sim$ in the universe of discourse *U* is a crisp set that contains all the elements of *U* that have membership values in $A^\sim$ equal to 1, that is [18]:

$$core\,(A^\sim) = \{(x \in U) \mid \mu_{A^\sim}(x) = 1\} \tag{7}$$

The *support* of a fuzzy set $A^\sim$ in the universe of discourse *U* is a crisp set that contains all the elements of *U* that have nonzero membership values in $A^\sim$, that is [19]:

$$supp\,(A^\sim) = \{(x \in U) \mid \mu_{A^\sim}(x) > 0\} \tag{8}$$

An $\alpha - cut$ a fuzzy set $A^\sim$ is a crisp set $A^\sim_\alpha$ that contains all the elements in *U* that have membership values in *A* greater than or equal to α, that is [19]:

$$A^\sim_\alpha = \{x \in U \mid \mu_{A^\sim}(x) \geq \alpha\} \tag{9}$$

Let $X = \{x_1, ..., x_b, ..., x_n\}$ be a set of *n* objects, and $V = \{v_1, ..., v_b, ..., v_c\}$ be a set of *c* centroids in a *p*-dimensional feature space. The Fuzzy C-Means partitions *X* into *c* clusters by minimizing the following objective function [4]:

$$J = \sum_{j=1}^{n}\sum_{i=1}^{c} (\mathbf{u}_{ij})^m \left\|\mathbf{x}_j - \mathbf{v}_i\right\|^2 \tag{10}$$

where $1 \leq m \leq \infty$ is the *fuzzifier*, $\mathbf{v}_i$ is the i^{th} centroid corresponding to cluster β_i, $\mathbf{u}_{ij} \in [0,1]$ is the fuzzy membership of the pattern $\mathbf{x}_j$ to cluster β_i, and $\|.\|$ is the distance norm such that,

$$\mathbf{v}_i = \frac{1}{n_i}\sum_{j=1}^{n} (\mathbf{u}_{ij})^m x_j \quad \text{where} \quad n_i = \sum_{j=1}^{n} (\mathbf{u}_{ij})^m \tag{11}$$

and

$$\mathbf{u}_{ij} = \frac{1}{\sum_{k=1}^{c}\left(\frac{d_{ij}}{d_{kj}}\right)^{\frac{2}{m-1}}} \quad \text{where} \quad d_{ij}^2 = \left\|\mathbf{x}_j - \mathbf{v}_i\right\|^2 \tag{12}$$

FCM starts by randomly choosing *c* objects as centroids (means) of the *c* clusters. Memberships are calculated based on the relative distance (Euclidean distance) of the object $\mathbf{x}_j$ to the centroids using Eq. (12). After the memberships of all objects have been found, the centroids of the clusters are calculated using Eq. (11). The process stops when the centroids from the previous iteration are identical to those generated in the current iteration [4].

4 Defuzzification by Feature Distance Minimization

Defuzzification by distance minimization *D(A)* of a fuzzy set *A* on a reference set *X*, with respect to the distance *d*, is [3]:

$$D\,(A) \in \{C \in P\,(X) \mid d\,(A,C) = min_{B \in P(X)}\ [d\,(A,B)\ ]\} \tag{13}$$

where *P(X)* is the set of crisp subsets of a power set, and *d* is the Minkowski distance between the vector representations of both the fuzzy set and the crisp subset, such that the fuzzy set is represented by its membership values that serve as separate features for every individual pixel [3].

This type of defuzzification of a fuzzy segmented image can be seen as an alternative to crisp segmentation, where, instead of crisp segmentation of gray level images, fuzzy segmentation is performed, and then followed by defuzzification [3].

4.1 Minimizing the distance between fuzzy and crisp sets

Floating search methods - SFFS (Sequential Forward Floating Selection) and SBFS (Sequential Backward Floating Selection) - are considered a development of the *l-r* algorithm, in which the values (features) of *l* and *r* are allowed to *float*, that is, they may change at different stages of the selection procedure [20]. Thus, floating search makes it flexible to change features so as to approximate the optimal solution as much as possible [21].

In *FCM-FloatingSearch* (see algorithm below), the enhancement of the floating search methods is the introduction of the fuzzy membership values and the neighbourhood information. Provided that both adding and removing pixels during region growing is allowed to happen [3].

```
FCM-FLOATINGSEARCH()
 1  call fuzzycmeans;
 2  input grayscale image;
 3  fuzzySet ← fuzzy segmented image;
 4  membershipMatrix ← μ; /* μ is the degree of
 5  membership */
 6  n ← area(support(fuzzySet)\
 7  α − cut(fuzzySet);
 8  C_0 = α − cut(fuzzySet);
 9  for i = 1 to n
10     do
11        C_i ← φ; /* empty set */
12  k ← 0;
13  /* add best pixel that minimizes the distance
14  between the fuzzy set and the crisp set */
15  while k < n
16     do
17        among the pixel p being 4 − neighbourhood
18         of C_k and not in C_k;
19        if pixel p ∈ support(fuzzySet)
20          then
21                p ← degree of membwership;
22        select the pixel p with the highest degree
23        of membership; /* minimizes
24        d̃_m (fuzzySet, C_k ∪ {p}) */
25        C_new ← Ck ∪ {p};
26        if C_{k+1} ← 0 ‖ d̃_m (fuzzySet, C_new) <
27          d̃_m (fuzzySet, C_{k+1})
28          then
29                C_{k+1} ← C_new;
30        k ← k + 1;
```

31 $/* \ possibly\ remove\ pixel\ */$
32 **do**
33 $change \leftarrow false;$
34 $among\ the\ pixels\ p\ being\ 4-neighbourhood\ of$
35 $\sim C_k and\ not\ in\ \alpha - cut(fuzzySet);\ /* \ \sim C_k\ is$
36 $the\ complement of\ C_k\ */$
37 **if** $pixels \in C_k$
38 **then** $p \leftarrow degree\ of\ membership;$
39 $select\ the\ pixel\ p\ with\ lowest\ degree\ of$
40 $membership;\ /* \ minimizes\ d_m^{\sim}(fuzzySet,$
41 $C_k \backslash \{p\})\ */$
42 $C_{new} \leftarrow C_k \setminus \{p\};$
43 $k \leftarrow k - 1;$
44 $change \leftarrow true;$
45 **while** $change\ \&\&\ k > 0$

In each iteration of the *FCM-FloatingSearch* algorithm, either for adding or removing a pixel, that selected pixel would be the pixel being a 4-neighbourhood of a crisp set C_k, and the one that minimizes the distance between the fuzzy set (fuzzy partition) and the crisp set (segmented image), including the added pixel in the iteration where we add pixels, and between the fuzzy set and the crisp set excluding the removed pixel in the iteration where we would possibly remove pixels. Fig.1 depicts the *FCM-FloatingSearch* algorithm general process. In sections 4.2 and 4.3, we show what would be considered the best added/removed pixel that minimizes the distance between the fuzzy set and the crisp set.

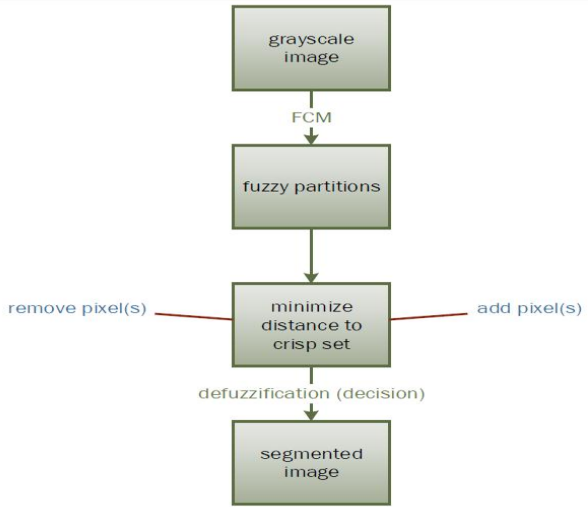


Figure 1: FCM-FloatingSearch

4.2 Adding best pixel

The pixel to be added to the crisp set at each iteration of the floating search algorithm has to meet three conditions: (1) belonging to the support and not the $\alpha - cut$ of the fuzzy set, (2) being a 4-neighbourhood of the crisp set, and (3) minimizing the distance between the fuzzy set and the crisp set including the added pixel.

The Minkowski distance between the fuzzy set f (*i.e.* fuzzy segmented image) and the crisp set C_k at iteration *k* is:

$$d_m(C_k, f) = \sqrt[m]{\sum_{i=1}^{length(C_k)} \mid C_k(i) - f(i) \mid^m} \quad (14)$$

After adding a pixel p_i to the crisp set, the Minkowski distance between f and the new crisp set $C_{new}\left(C_{k\cup\{p_i\}}\right)$becomes:

$$d_m(C_{new}, f) = \sqrt[m]{\sum_{i=1}^{length(C_k)} \mid C_{new}(i) - f(i) \mid^m} \quad (15)$$

Eq. (15) can be rewritten as:

$$d_m(C_{new}, f) = (\mid C_{new}(p_i) - f(p_i) \mid^m + \sum_{i=1; i\neq p_i}^{length(C_k)} \mid C_k(i) - f(i) \mid^m)^{1/m} \quad (16)$$

The difference between C_k and C_{new} is only in the element at location p_i. Thus, in order to include p_i in C_k, $|C_k(p_i) - f(p_i)|^m$ would be subtracted from $|C_{new}(p_i) - f(p_i)|^m$, and the condition $j \neq p_i$ in the summation of Eq. (16) would be removed.

As the singleton pixel *pi* is added to C_{new}, $C_{new}(p_i) = 1$, and the corresponding location in $C_k(p_i) = 0$. By substituting those values , we get:

$$d_m^{\sim}(C_{new}, f) = (\mid 1 - f(p_i) \mid^m - \mid -f(p_i)\mid^m + \sum_{i=1}^{length(C_k)} \mid C_k(i) - f(i) \mid^m)^{1/m} \quad (17)$$

Assuming that $f(p_i)$ represents the degree of membership of some pixel, which, at the same time, represents the feature of that pixel, it can be noticed from Eq. (17) that as the value of $f(p_i)$ increases, the distance value decreases.

So, when selecting the pixel that minimizes the distance between the fuzzy set and the crisp set, the pixel with the *highest* degree of membership is the pixel which will be added.

4.3 Removing pixels

The pixel to be removed from the crisp set at each iteration of the floating search algorithm has to meet three conditions: (1) belonging to the crisp set and not to the α-cut of the fuzzy set, (2) being a 4-neighbourhood of the crisp set complement, and (3) minimizing the distance between the fuzzy set and the crisp set excluding the removed pixel.

After removing a pixel p_i from the crisp set, the Minkowski distance between f and the new crisp set $C_{new}\left(C_{k\setminus\{p_i\}}\right)$ becomes:

$$d_m(C_{new}, f) = \sqrt[m]{\sum_{i=1}^{length(C_k)} \mid C_{new}(i) - f(i) \mid^m} \quad (18)$$

Following the procedure as that in section 4.2, we conclude:

$$d_m^{\sim}(C_{new}, f) = (\mid -f(p_i) \mid^m - |1 - f(p_i)|^m + \sum_{i=1}^{length(C_k)} \mid C_k(i) - f(i) \mid^m)^{1/m} \quad (19)$$

It can be noticed from Eq. (19) that as the value of $f(p_i)$ decreases. the distance value decreases.

So, when selecting the pixel that minimizes the distance between the fuzzy set and the crisp set, the pixel with the *lowest* degree of membership is the pixel which will be removed.

4.4 Novelty of the proposed distance

The proposed distance, in contrast to other distance measures, takes into account the *membership degree* values of the elements in the fuzzy set, in addition to the *spatial relationship* (neighbourhood information). It is also very flexible in trying to minimize the distance, as the approach tries to enhance the result when adding/removing pixels.

Example: $A = \{1.3, 5.4, 3.7, 2.1, 4.5\}$*;* $B = \{6.8, 2.3, 7.9, 10.1, 3.7\}$*;* $C = \{5.3,\ 9.4,\ 3.1,\ 4.8,\ 9.9\}$*;* *(assumptions:* $m = 2$, $degree\ of\ membership\ when\ adding\ pixel = 1$, $degree\ of\ membership\ when\ removing\ pixel = 0$*)*

For comparing the result of the proposed distance with the Euclidean distance, Manhattan distance, and the classical Minkowski distance, two approaches can be used (Since we need to know the covariance of the data for Mahalanobis distance, which is not always available, Mahalanobis distance was not used in this example as it is beyond its scope) :
(i) Calculate the distance between the *median* of the two sets:
d_2 (A,B) = 3.1
$d_{Manhattan}$ (A,B) = 3.1
d_m (A,B) = 3.1
$d_m^{\sim}$(A,B) = 2,93

(ii) Find the average of the distances between each pair:
d_2 (A,B) = 4.32
$d_{Manhattan}$ (A,B) = 4.32
d_m (A,B) = 4.32
$d_m^{\sim}$(A,B) = 4.12

It can be noticed that the proposed distance gives the lowest value among the other distances, which is an important characteristic in *FCM-FloatingSearch*.

The proposed distance can be considered as a *metric*, as it complies with the metric postulates mentioned in *section 2*, that is, *non-negativity*, *reflexivity*, *symmetry*, and *triangle inequality*. Provided that, in the case of reflexivity, the fuzzy set is considered a crisp set as they are equal, and can be treated as the classical Minkowski distance (fuzzy membership values omitted).

- Non-negativity: $d_m^{\sim}$(A,B) = 11.00
- Reflexivity: $d_m^{\sim}$(A, A) = 0
- Symmetry: $d_m^{\sim}$(A,B) = $d_m^{\sim}$(B,A) = 11.00
- Triangle inequality: $d_m^{\sim}$(A,B) + $d_m^{\sim}$(B,C) $\{22.88\} > d_m^{\sim}$(A,C) $\{8.23\}$

5 Evaluation

In this section, we will quantitatively evaluate our segmentation results produced by the *FCM-FloatingSearch* algorithm. We will follow a low level supervised evaluation criteria, that is, the segmentation output is what would only be considered in the evaluation, and the original image information will not be taken into account. Such assessment of the quality of segmentation is achieved by comparing the segmentation output to a ground truth [24].

5.1 Image database

An image database [25] composed of synthetic images having a ground truth was used. The database[1] includes 8400 images. Images used in the study were specifically extracted from the *B0U* group which is composed of 100% textured regions.

5.2 Influence of the FCM fuzzifier

The fuzzifier $m \in [1, +\infty)$ has a significant impact on the performance of FCM. It controls the amount of fuzziness of the final C-partition in the FCM algorithm [22]. Pal and Bezdek suggested that the value of *m* is probably in the interval [1.5, 2.5] [23]. Huang et al [22] suggest that the range of values of *m* that create significant changes in the FCM membership values, and which are considered as effective boundaries for the level of fuzziness, is approximately [1.4, 2.6]. They also recommend that an analyst should not be concerned about the changes of the membership values outside of these boundaries, as they encapsulate the uncertainty associated with the level of fuzziness parameter. For *m*, most researchers adopt $m = 2$ when performing the FCM algorithm [22]. Although, if we take the average of the above two suggested boundaries: (1.5+2.5)/2=2 and (1.4+2.6)/2=2. Thus, the fuzziness parameter value chosen in this paper is 2.

5.3 Segmentation results

Images extracted from the *B0U* group of the database were segmented according to the *FCM-FloatingSearch* process shown in Fig. 1. In order to analyze the segmentation results (quality, accuracy, extracted information, *...etc*), nine observation criteria have been used [26]: *Precision*, *Recall*, and the *Dice index* (or *F-measure*), which characterize the overall quality of the segmentation area. The *Manahattan* (or *Matching*) *index* gives the ability to study the similarity rate of the entire image. The *Jaccard* (or *Tanimoto*) *index* studies the similarity rate between two segmentation areas. The above criteria are based on statistical tests of true or false positives, denoted TP and FP, respectively. And, true or false negatives, denoted, TN and FN, respectively.

[1] http://www.ecole.ensicaen.fr/~rosenber/ressources_UK.html

Precision and *Recall* are defined by:

$$Precision = \frac{TP}{TP + FP} \quad and \quad Recall = \frac{TP}{TP + FN} \tag{20}$$

The *Dice index* is defined by:

$$Dice\ index = 2.0 \times \frac{Precision \times Recall}{Precision + Recall} \tag{21}$$

The *Manhattan index* is defined by:

$$Manhattan\ index = \frac{TP + TN}{TP + FP + TN + FN} \tag{22}$$

and, the *Jaccard index* is defined by:

$$Manhattan\ index = \frac{TP}{TP + FP + FN} \tag{23}$$

The other criteria are: *Hamming measure*, which calculates the number of disparities between two images, and is defined by:

$$M_H\left(I_1 \Rightarrow I_2\right) = n - \sum_{R_2 \in I_2} max_{R_1 \in I_1} \left|R_2 \cap R_1\right| \tag{24}$$

where R_1 and R_2 are segmentation areas in the images I_1 and I_2, respectively. And, n is the number of pixels of one image.

The *mean absolute distance* (*MAD*), which analyzes the contour points, and thus, the shape of the segmentation, is defined by:

$$MAD\left(R_1, R_2\right) = \frac{1}{M} \sum_{m=1}^{M} \left\|x_m - y_m\right\| \tag{25}$$

where x_m and y_m are contour points of R_1 and R_2, respectively.

And, the *structural similarity* (*SSIM*) for the extracted structural information, is defined by:

$$SSIM\left(R_1, R_2\right) = \frac{\left(2m_1m_2 + k_1\right)\left(2cov_{1,2} + k_2\right)}{\left(m_1^2 + m_2^2 + k_1\right)\left(\sigma_1^2 + \sigma_2^2 + k_2\right)} \tag{26}$$

where m_1 and m_2 are the average values of R_1 and R_2. σ_1^2 and σ_2^2 are the variance, $cov_{1,2}$ is the covariance. k_1 and k_2 are two coefficients proportional to the dynamic range of the pixel values.

Fig. 2 shows that we can obtain a perfect defuzzification, similar to that proposed in [3], when the input image is fuzzified.

Fig. 3 shows the results of the proposed approach, applied on a specific image, using different $\alpha - cut$ values, along with its fuzzy partition.

Fig. 4 shows results of applying the proposed approach on more complex images with different number of textures (*B0UnR*, where *n* represents the number of textures), along with the $\alpha - cut$ vlaues being used.

Fig. 5 shows a cell image without a ground truth.

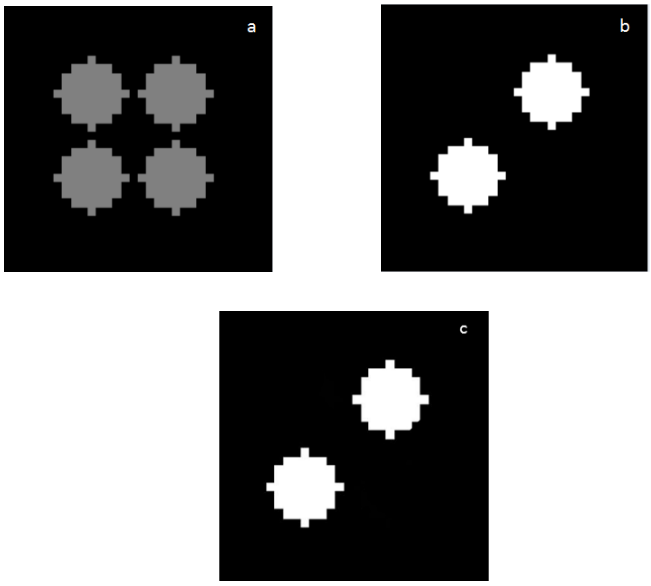


Figure 2: (a) digital disk synthetic image; (b) result of the approach in [2]; (c) result of the proposed approach

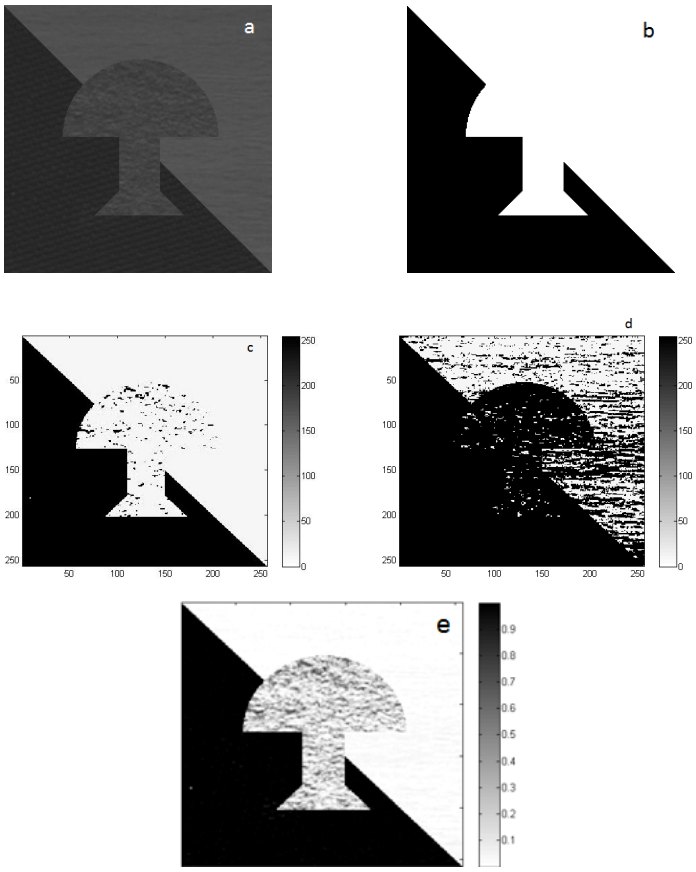


Figure 3: Results of applying the FCM-FloatingSearch process: (a) three-textured image; (b) ground truth of (a); (c) defuzzification with $\alpha - cut = 0.5$ of (a); (d) defuzzification with $\alpha - cut := 1$ of (a); and (e) fuzzy partition of (a).

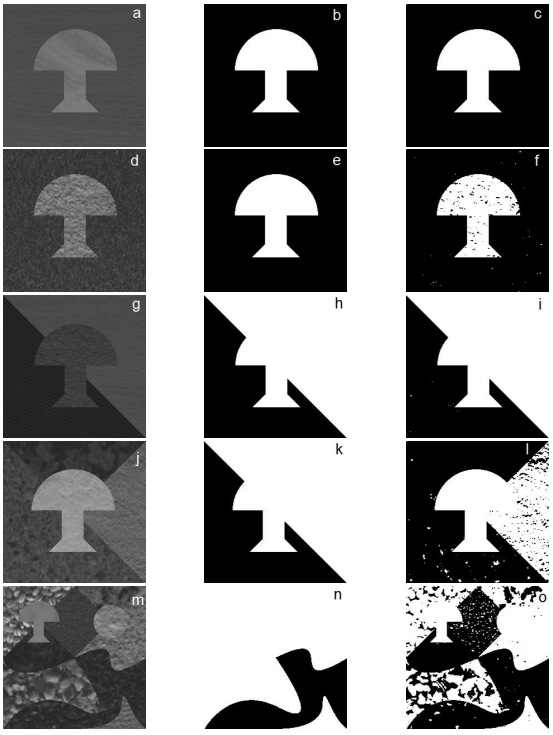


Figure 4: (a) B0U2R_1 image; (b) ground truth of (a) ; (c) defuzzification $\alpha - cut : 0.5$ of (a) ; (d) B0U2R_5 image; (e) ground truth of (d); (f) defuzzification $\alpha - cut : 0.5$ of (d); (g) B0U3R_26 image; (h) ground truth of (g); (i) defuzzification $\alpha - cut : 0.1$ of (g); (j) B0U4R_59 image; (k) ground truth of (j); (l) defuzzification $\alpha - cut : 0.1$ of (j); (m) B0U10R_4 image ; (n) ground truth of (m); (o) defuzzification $\alpha - cut : 0.1$ of (m)

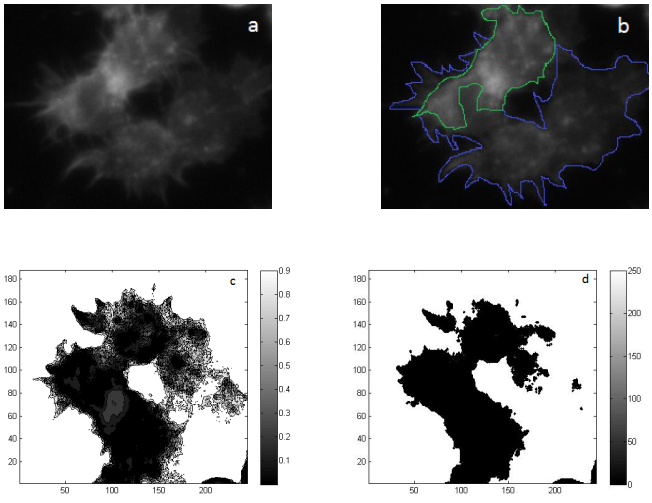


Figure 5: (a) cell image; (b) rough sketch of core $\alpha - cut : \ = 1$ (green border) and support (blue border); fuzzy partition of (a); defuzzification with $\alpha - cut = 0.5$ of (a).

	B0U2R_1	B0U2R_5	B0U3R_26	B0U4R_59	B0U10R_4
Precision	100%	99.09%	100%	65.64%	47.05%
Recall	100%	99.91%	99.96%	98.69%	98.64%
Dice	100%	99.50%	99.98%	78.84%	63.71%
Jaccard	100%	99.01%	99.96%	65.08%	46.75%
Manhattan	100%	99.18%	99.98%	78.51%	71.54%
Hamming	1	536	11	14083	18651
Coeff. Vinet	0.002%	0.82%	0.02	21.49%	28.46%
MAD	0	8.10	0.62	17.69	23.79
SSIM	0.9997	0.9118	0.9951	0.6315	0.5185

Table 1: Segmentation results (evaluation) of FCM-FloatingSearch

Tab. 1 summarizes segmentation results (evaluation) of the proposed approach. Dice index emphasizes the quality of the segmentation. The Mean Absolute Distance highlights the shape of the segmentation, and the SSIM highlights the quantity of extracted information. Jaccard and Manhattan indices highlight the problems of sub- or over-segmentation. The number of Hamming is the number of disparity between the segmentation and ground truth. Thus, as can be noticed from the table, *FCM-FloatingSearch* provides good segmentation results for the first three images despite noise or other factors. For the remaining two images, although over-segmentation is introduced, *FCM-FloatingSearch* extracts almost all of the information corresponding to the ground truth. Over-segmentation could be solved by extracting the relevant ROI (Region of Interest) as a preprocessing step to *FCM-FloatingSearch*. That is, selecting the best crisp set manually.

6 Conclusion

In this paper, we have presented a novel defuzzification approach, based on Fuzzy C-Means, with an adapted Minkowski distance. For an image without ground truth, the algorithm removes additional unnecessary data, providing a more clearer segmentation. It can be noticed that the choice of a proper $\alpha - cut$ value is essential to the defuzzification output, provided that it is not necessary that the highest $\alpha - cut$ value implies a better defuzzification. Thanks to the ground truth, FCM-FloatingSearch was able to be evaluated for the ability of detecting that ground truth. For images with ground truth, and of different textures, FCM-FloatingSearch was able to provide good segmentation despite noise and other factors. And, where over-segmentation occurred, the proposed approach was able to extract almost all the information corresponding to the ground truth. This can be enhanced by selecting the best crisp set manually. The adapted Minkowski distance proposed in the paper, which took into account the fuzzy membership degrees and the neighbourhood information, showed better results compared to other distance measures. A prospect for this work is to combine the algorithm proposed with different fuzzifier values in FCM. Also, since the defuzzification in this paper was based on one cluster, a future work would be to adapt the algorithm for a number of clusters higher than 2 (foreground/background) as a first

step, and then, explore the possibility to integrate the proposed defuzzification approach for other clustering methods such as RFCM, Competitive Agglomerative Clustering, ... *etc*.

References

[1] Judith Prewitt. *Object Enhancement and Extraction*. Picture Process Psychopict, pages 75-194, 1970.

[2] Pavel Pudil, Jana Novovičová, and Josef Kittler. *Floating Search Methods in Feature Selection*. Pattern Recognition Letters, volume 15, pages 1119-1125, 1994.

[3] Nataša Sladoje, Jokaim Lindblad, and Ingela Nyström. *Defuzzification of Spatial Fuzzy Sets by Feature Distance Minimization*. Image and Vision Computing, volume 29, no. 2-3, pages 127-141, 2011.

[4] Pradipta Maji and Sankar Pal. *Maximum Class Separability for Rough-Fuzzy C-Means Based Brain MR Image Segmentation*. T. Rough Sets, 9, pages 114-134, 2008.

[5] Lisa Serir, Emmanuel Ramasso, and Noureddine Zerhouni. E2GK: *Evidential Evolving Gustafsson-Kessel Algorithm for Data Streams Partitioning Using Belief Functions*. ECSQARU, pages 326-337, 2011.

[6] Alexander Strehl and Joydeep Ghosh. *Relationship-based Clustering and Visualization for High-dimensional Data Mining*. INFORMS Journal on Computing, 15(2), pages 208 230, 2003.

[7] Michael Berry and Gordon S. Linoff . *Data Mining Techniques: For Marketing, Sales and Customer Relationship Management, 2nd Edition*. Wiley Publishing, 2009.

[8] Ardeshir Goshtasby. *Image Registration: Principles, Tools and Methods*. Advances in Computer Vision and Pattern Recognition, Springer, ISBN 978-1-4471-2457-3, pp. I-XVIII, 1-441, 2012.

[9] David Hand, Heikki Mannila, and Padhraic Smyth. *Principles of Data Mining*. MIT Press, 2001.

[10] Levent Ertöz, Michael Steinbach, and Vipin Kumar. *Finding Clusters of Different Sizes, Shapes, and Densities in Noisy, High Dimensional Data*. Clustering and Information Retrieval, pages 83-104, 2003.

[11] Sergiy Butenko, W Art Chaovalitwongse, and Panos M Pardalos. *Clustering Challenges in Biological Networks*. World Scientific, 2009.

[12] Patrick Groenen,Uzay Kaymak, and Joost Rosmalen. *Fuzzy Clustering with Minkowski Distance Functions*. Econometric Institute Report, 2006.

[13] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques, 3rd Edition*. Elsevier, 2011.

[14] Geoffrey McLachlan. *Mahalanobis Distance*. Reasonance 4, pages 20-26, 1999.

[15] Paul Lewicki and Thomas Hill. *Statistics: Methods and Applications : a Comprehensive Reference for Science, Industry, and Data Mining*. StatSoft Inc, 2006.

[16] A.K. Sharma. *Introduction To Set Theory*. Discovery Publishing House, 2010.

[17] Hossein Bidgoli. The Internet Encyclopedia: Volume 2. John Wiley Sons, 2004.

[18] M. Ganesh. *Introduction to Fuzzy Sets And Fuzzy Logic*.PHI Learning Pvt. Ltd, 2006.

[19] Li-Xin Wang. *A Course in Fuzzy Systems and Control*. Prentice Hall, 1997.

[20] test Andrew Webb and Keith Copsey. *Statistical Pattern Recognition*. John Wiley Sons, 2011.

[21] Huan Liu and Hiroshi Motoda. *Feature Extraction Construction and Selection: A Data Mining Perspective*. Springer, 1998.

[22] Ming Huang, Zhixun Xia, Hongbo Wang, Qinghua Zeng, and Qian Wang. *The range of the value for the fuzzifier of the fuzzy c-means algorithm*. Pattern Recognition Letters, volume 33, issue 16, pages 2280-2284, 2012.

[23] Nikhil Pal and James Bezdek. *On Cluster Validity for the Fuzzy C-Means Model*. IEEE Trans. Fuzzy Syst 3, pages 370-379, 1995.

[24] M. Beauchemin, K. Thomson, and G. Edwards. *On the Hausdorff Distance Used For the Evaluation of Segmentation Results*. CJRS, volume 24(1), pages 38, 1998.

[25] Sebastien Chabrier, Bruno Emile, Christophe Rosenberger, and Helene Laurent. *Unsupervised Performance Evaluation of Image Segmentation*. EURASIP Journal on Applied Signal Processing, Special Issue on Performance Evaluation in Image Processing, 1-12, 2006.

[26] Manuel Grand-Brochier, Antoine Vacavant, Guillaume Cerutti, Kvin Bianchi and Laure Tougne. *Comparative Study of Segmentation Methods for Tree Leaves Extraction*. In ACM ICVS 2013, Workshop: VIGTA, Saint Petersburg, Russia, 2013.

Computer Vision Winter Workshop 2014, Zuzana Kúkelová and Jan Heller (eds.)
Křtiny, Czech Republic, February 3–5
Czech Society for Cybernetics and Informatics

jSLIC: superpixels in ImageJ

Jiří Borovec and Jan Kybic

Faculty of Electrical Engineering,
Czech Technical University in Prague, Czech republic
jiri.borovec@fel.cvut.cz

Abstract *This paper presents the implementation and particular improvements on the superpixel clustering algorithm - SLIC (Simple Linear Iterative Clustering). The main contribution of the jSLIC is a significant speed-up of the original clustering method, transforming the compactness parameter such that the value is image independent, and a new post-processing step (after clustering) which now gives more reliable superpixels - the newly established segments are more homogeneous. The improvements of speed and quality are shown on real images. We implemented the new jSLIC in Java and made the source code publicly available. Also we created a plug-in in ImageJ/Fiji which is commonly used as a research and development platform in biology and medical imaging.*

1 Introduction

The amount of data in medical imaging to be processed is increasing - images in histology can easily have 50.000×50.000 pixels or even more. The segmentation or registration of these large images is very demanding. The complexity of segmentation and registration can be reduced by using superpixels [7, 4].

In the past, several superpixel algorithms were introduced which were based on, for example the watershed approach, level-set based geometric flow, mode-seeking segmentation scheme or graph-based (a comparison is presented in [2, 9]). Recently, SLIC (Simple Linear Iterative Clustering) [2] was introduced for general images and presented as a powerful intermediate phase for further image segmentation, classification and registration.

We chose SLIC because of its universality and linear (and low) complexity (see Sec. 1.1). This fact is important for pre-processing large images. SLIC has a high rate in boundary recall and a low rate of under-segmentation error [2]. Another benefit is the low number of parameters to be set and an opportunity to influence the size and compactness of the resulting superpixels.

The main contribution of this work is a significant speed-up over the original clustering method and also providing a multi-thread version. Moreover, the regularisation parameter is transformed into the range $\langle 0, 1 \rangle$ to be more image independent. We also propose a new post-processing step which gives more reasonable superpixel shapes even for larger superpixel grid spacing.

While ImageJ [1, 5, 11] (and Fiji, derivation of ImageJ) is commonly used as a research and development platform in medical imaging, there is no implementation of superpixels. We decided to implement SLIC in Java and call it **jSLIC**. The Java source code and a ready to install Fiji plug-in are publicly available[1].

In Sec. 1.1, we briefly introduce the general SLIC algorithm. Then, we discuss the implementation and proposed speeds-ups together with the explanation and gain of each partial procedure in Sec. 2. Later in Sec. 3, we speak about the post-processing phase where we define the problem, summarize the existing approach and introduce ours and present the differences on the atcome using both methods.

1.1 SLIC superpixels

SLIC [2] is an adaptation of the k-means [6] algorithm for superpixel generation with two important distinctions: (a) the weighted distance measure

$$D = \sqrt{d_c^2 + \left(\frac{d_s}{S}\right)^2 m^2} \tag{1}$$

combines colour d_c (using the CIELAB colour space, which is widely considered as perceptually uniform for small colour distances) and spatial proximity d_s and (b) the search space is reduced by limiting to a region $2S \times 2S$, proportional to the superpixel size S. The search space reduction has a great impact on the speed of whole algorithm, resulting on a complexity of only $O(N)$ instead of $O(NkI)$ for standard k-means, where N is the number of pixels in a image, k is the number of clusters and I is the number of iterations [3].

2 Implementation and speed-ups

Several implementations of SLIC already exist. The author of [2] provides a C source code[2] which was wrapped into Python[3] (we use this code as the reference of SLIC). Other implementations can be found also for Matlab (VLFeat[4] library). For real-time computer vision problems, SLIC has been also transformed to be fully processed on graphic cards with some minor improvements as gSLIC [10].

[1] http://fiji.sc/CMP-BIA_tools
[2] http://ivrg.epfl.ch/research/superpixels
[3] https://github.com/amueller/slic-python
[4] http://www.vlfeat.org/

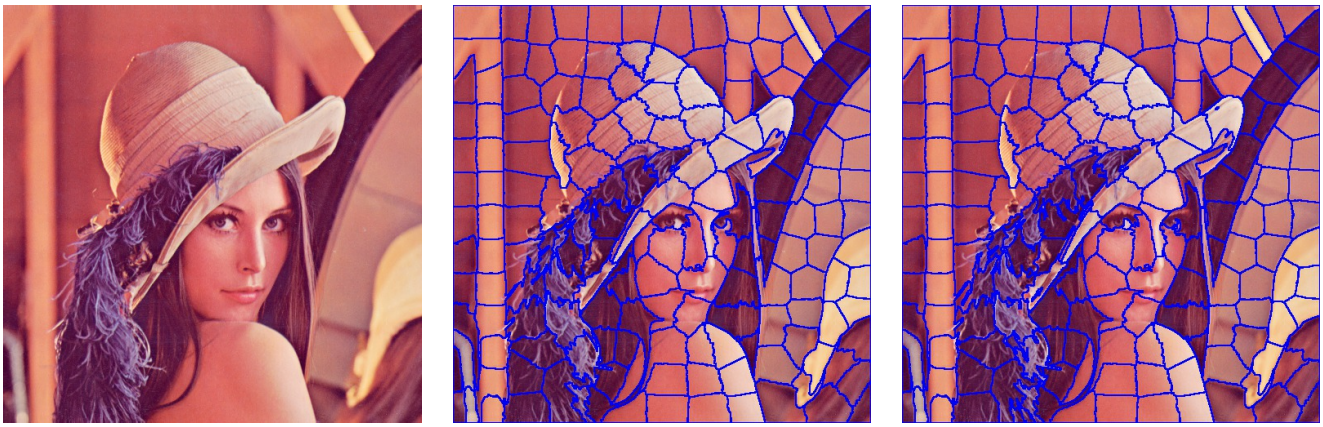


Figure 1: Sample image - Lena (image size 512×512 pixels) clustered by the original SLIC (middle) and our jSLIC (right) method. You can see that most of the superpixels are equal except those around Lena's eyes where jSLIC added extra superpixels for the white which we consider as the right choice.

We implement the plug-in in Java with maximal focus on compatibility with ImageJ. We used the ImageJ API and as much as possible we had to use the native Java structures a few times to keep the clustering process as fast as possible.

2.1 Regularisation constant

SLIC contains a regularisation parameter f which influences the compactness of clustered superpixels. This constant f weights the space distance d_s and it is expressed as $f = \left(\frac{m}{S}\right)^2$ from eq. (1) where (according to the notation in [2]) S is the initial superpixel size and m is a parameter related to maximal colour distance N_c in the range $(0, \infty)$. We propose instead to use a parameter r defined in the range $\langle 0, 1 \rangle$, where 0 means the minimal and 1 the maximal compactness.

$$f = S \cdot r^2 \tag{2}$$

In our experiments we found that the optimal default regularisation value $r = 0.2$ works well for most cases. It is a good compromise between the superpixel compactness and fitting boundaries of the expected object in image.

2.2 Using Look-Up Tables

We analysed the possibility of using precomputed Look-Up Tables (LUTs) to avoid repetitive computing of the same distances d_s in eq. (1) or converting the same colours again. We found that we can achieve significant speed-up in specific cases (especially for colour conversion) mentioned below.

Spatial distance in regular grid. The metric used in SLIC clustering contains a proximity distance

$$d_s = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$$

where $[x_i, y_i]$ and $[x_j, y_j]$ are coordinates of the cluster centre and a pixel respectively. In a regular image grid, these distances are the same for all cluster centres and its proportional subset of neighbouring pixels. Using this precomputed distance LUT, we gain a 5% speed-up.

Colour conversion. Most commonly used images are in RGB colour space and we compute the colour distance in CIELAB colour space (see Sec. 1.1). It means that each image needs to be converted from RGB to CIELAB which is quite time consuming. We found that the number of used unique colours in images is usually smaller than the number of pixels in the image. Average images has at maximum 50% of unique colours/pixels (e.g. Lena with size 512×512 pixels). For medical images, the ratio is even smaller. For example, a common image of stained histological section (see [4]) contains less than 5% of unique colours/pixels. So we create the conversion LUT just when it is needed so each used colour is computed just once. It gives us a speed-up of about 60%.

method	speed-up
original SLIC	0%
jSLIC initial	26.6%
spatial proximity LUT	33.7%
colour conversion LUT	217.3%
jSLIC fast (distance & colour)	264.9%
jSLIC parallel (4 threads)	495.4%

Table 1: Table presents the speed-ups of each proposed procedure. All following ratios are mean speed-ups evaluated over several histological images with different image size (see Fig. 5) and they express the relative speed-up to the original SLIC. On the beginning the jSLIC (implementation according [2]) is about 27% faster then the original SLIC implemented in C. Later the pre-computation of distances and converting each colour just once brings 5% and 58% speed-up respectively comparing to the initial jSLIC and about 64% both together. In the end, the parallelisation for 4-threads gives another speed-up of 37% to the fast jSLIC.

2.3 Multi-threading

As the clustering is computed locally (for each superpixel only its proportional $2S \times 2S$ area, see Sec. 1.1), is quite simple to split the process by subsets of superpixels and/or image blocks into independent threads in both phases (assignment and update).

We apply the parallelism usually on the main loop in the given phase - in the assignment phase each thread takes only a subset of all superpixels/clusters, and the update is computed per image blocks, such that each thread processes one image block.

We perform this parallelisation on a computer with 8-

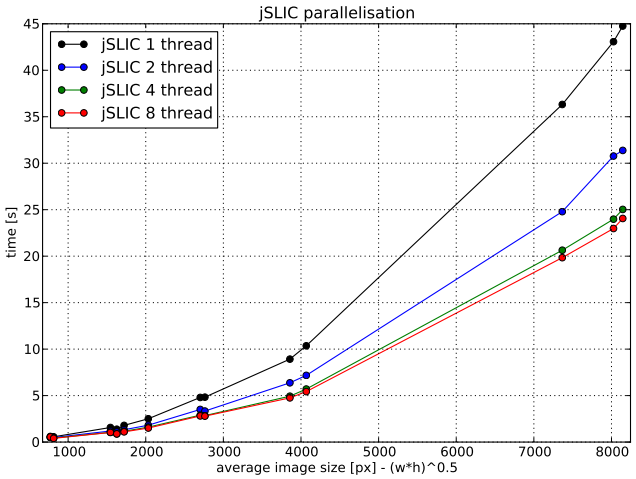


Figure 2: We ran a benchmark on several histological images with different image size and parallelism on $1-8$ threads. You can see that with the increasing number of used threads the processing time also decrease. The most significant speed-up is between the single and 4-thread version.

cores and the results are presented in Fig. 2. The most significant speed-up is between the single and 4-thread version. You can see that the 8-thread version for small images takes even more time which is due to multi-threading overhead.

3 Post-processing of outliers

The SLIC clustering generates a quite large number of unconnected components (small regions which belong to a superpixel but they are not connected to it). The number of unconnected regions depends on superpixel compactness but in average (for regularisation $r = 0.2$) there are about $3M$ unconnected regions where $M = \frac{w \cdot h}{S^2}$ is number of expected superpixels depending on image size and initial superpixel size S.

At first, all connected components c_i have to be found. We use a region growing method to compute all independent components c_i (assuming 4-neighbour). Then, for each component c_i we find a set of neighbouring components Ω_i. This early stage is the same for the original SLIC as it is for jSLIC post-processing.

Original SLIC post-processing [2]. The authors measure the relative area $c_i^s = \frac{|c_i|}{S^2}$ of each component and merge small components if $c_i^s < 0.25$. For relabelling they simply use the label c_i^* of the first component from Ω_i such that c_i^* is the neighbouring component of the first pixel belonging to c_i.

We found out that this simple approach is not sufficient (see Fig. 3), because some unconnected components are merged to a superpixels even it would be more reasonable to merge then into another neighbouring superpixel or introduce them as new superpixels. The author deals with this issue by estimating smaller superpixels [7] and setting the superpixel size smaller than the smallest detail in the image that they want to distinguish.

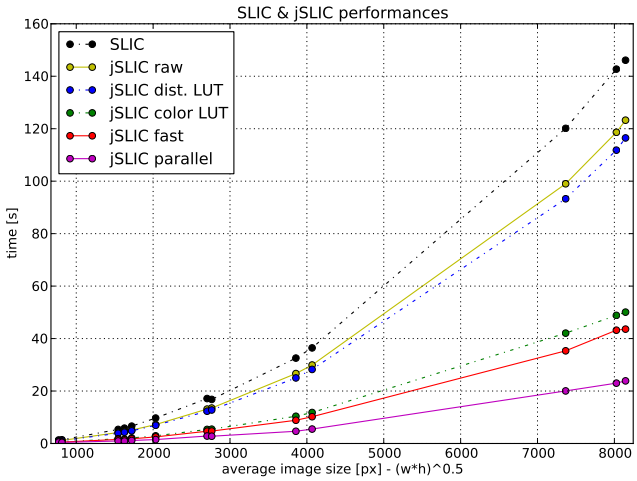


Figure 5: The chart presents the time dependency of complete superpixel clustering by SLIC and different variants of jSLIC depending on the number of pixels in the image. In average, the parallel jSLIC is 6 times faster than the original SLIC implementation.

Proposed jSLIC post-processing. We propose a different post-processing step which takes into account all surrounding components and their similarity by colour and area. We compute mean colours c_i^{lab} and relative area c_i^s for all components. Then, we find the most similar component c_i^* by computing the difference $l_i\left(c_j\right)$ between the colours of the components (3) and choosing the closest component (4) with minimal distance

$$l_i\left(c_j\right) = \frac{||c_i^{lab} - c_j^{lab}||_2}{c_j^s} \quad (3)$$

$$c_i^* = \arg\min_{c_j \in \Omega_i} \left(l_i\left(c_j\right)\right) \quad (4)$$

where the $||\cdot||_2$ is the Euclidean distance.

We experimented with the SLIC relabelling condition for unconnected components (see Fig. 3). We found the original $c_i^s < \epsilon$ condition insufficient even with various threshold values ϵ, because it does not take into account the colour similarity. We propose a condition which solves this problem - the unconnected regions are merged if

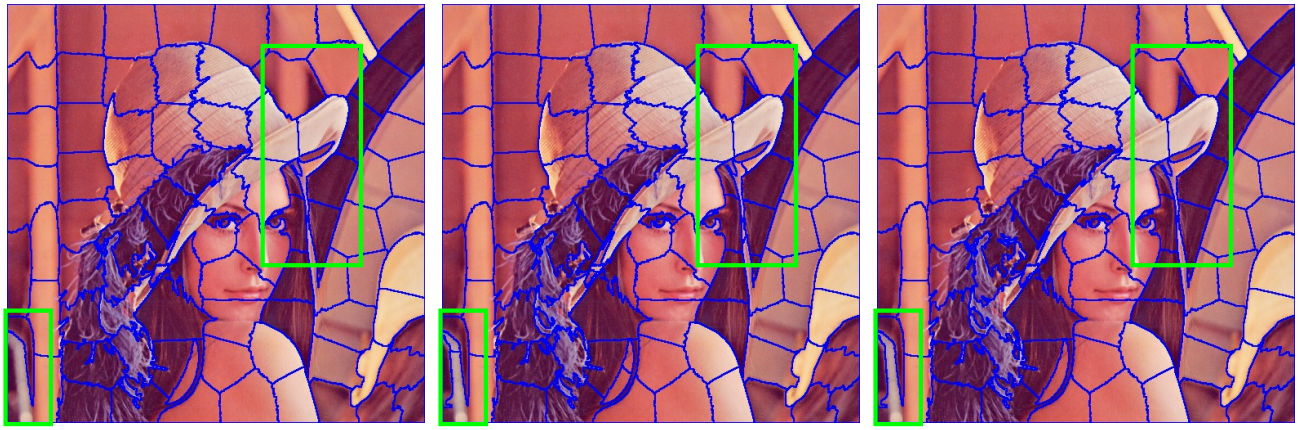
$$\left(\frac{c_i^s}{4}\right)^2 \cdot \left(1 + l_i\left(c_i^*\right)\right) < \varepsilon \quad (5)$$

where $\frac{c_i^s}{4}$ expresses the relative superpixel size to the maximal superpixel size $2S \times 2S$. Experimentally, we set the threshold $\varepsilon = 0.25$.

4 Comparison and discussion

We applied jSLIC on several histological images of various image sizes, up to about 8.000×8.000 pixels, on a standard computer with a 4-core processor and 8Gb RAM. As a reference we used the original SLIC implementation in C and compared it to our jSLIC in Java. The time dependency of all partial speed-ups on image size is presented in Fig. 5. In average, we found the parallel jSLIC to be 6 times faster than the original SLIC implementation.

The experiments with parallelism show that the jSLIC is optimal when using up to 4-threads. Using more threads due



(a) (b) (c)

Figure 3: We compared the original SLIC condition for merging unconnected components $c_i^s < \epsilon$ applying two different thresholds - original $\epsilon = 0.25$ in (a) and decreased $\epsilon = 0.06$ in (b). For all relabelling, we used our choosing of most similar neighbouring component c_i^* defined in eq. (4). In (c) we introduced also our condition for merging described in eq. (5). You can see that most of the superpixels in (a, ,b, c) are the same. The difference can be seen in the right upper part of the image. Original SLIC (a) just holds one large superpixel comparing to (b, c) which reasonably adds one more superpixel. On the other hand (b) adds some other small superpixels in nearly homogeneous areas, while (c) holds still single superpixels.

to the threading overhead, does not brings bigger improvements in performance.

For the evaluation of the proposed post-processing step, we used a few images from the Berkeley Segmentation Dataset [8] and some stained histological images (see Fig. 4). We made a visual evaluation of segmented superpixels with respect to the amount of detail extracted from a given image. For both methods we set the same configuration - the same initial superpixel size $S = 30$ and regularisation constant $r = 0.2$. To present the differences, we chose a detail in each image where the improvements can be easily seen (the rest of the image is usually segmented equally).

The advantage of the jSLIC post-processing is the ability to segment also smaller details than the initial superpixel size S in region it is needed and the ability to keep larger superpixels in more uniform image parts (see Sec. 3). We benefit from this fact when segmenting large histological images, where a big reduction of problem complexity is needed. For instance, have a look at the sample of histological image (Fig. 4 bottom), where the jSLIC is clearly capable of estimating the hole in the tissue comparing to original SLIC method.

5 Conclusion

We presented a Java-based open source implementation of jSLIC superpixel clustering with better performance than the original SLIC. Moreover, we proposed a different regularisation parameter, which influences the compactness of resulting superpixels and propose a default value $r = 0.2$. The new post-processing step gives more reliable superpixels shapes, with no need of decreasing superpixel size.

Acknowledgement

The authors were supported by The Czech Science Foundation under project P202/11/0111 and by The Grant Agency of the CTU Prague under project SGS12/190/OHK3/3T/13.

References

[1] M.D. Abramoff, P.J. Magalhães, and S.J. Ram. Image processing with ImageJ. *Biophotonics international*, 11(7):36–42, 2004.

[2] R. Achanta and A. Shaji. SLIC Superpixels Compared to State-of-the-art Superpixel Methods. *Pattern Analysis and Machine Intelligence, IEEE*, 34(11):2274 – 2282, 2012.

[3] R. Achanta, A. Shaji, K. Smith, and A. Lucchi. Slic superpixels. Technical report, 2010.

[4] J. Borovec. Fully automatic segmentation of stained histological cuts. In Libor Husník, editor, *17th International Student Conference on Electrical Engineering*, pages 1–7, Prague, 2013. CTU in Prague.

[5] T.J. Collins. ImageJ for microscopy. *Biotechniques*, 43(S1):S25–S30, July 2007.

[6] J.A. Hartigan and M.A. Wong. Algorithm AS 136: A K-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, Oct. 1979.

[7] A. Lucchi, K. Smith, and R. Achanta. Supervoxel-Based Segmentation of Mitochondria in EM Image Stacks With Learned Shape Features. *Medical Imaging, IEEE*, 31(2):474 – 486, 2012.

[8] D. Martin and C. Fowlkes. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *International Conference on Computer Vision, IEEE*, number July, 2001.

[9] P. Neubert and P. Protzel. Superpixel Benchmark and Comparison. Technical report, 2012.

[10] C.Y. Ren and I. Reid. gSLIC: a real-time implementation of SLIC superpixel segmentation. Technical report, 2011.

[11] C. Schneider, W.S. Rasband, and K.W. Eliceiri. NIH Image to ImageJ: 25 years of image analysis. *Nature Methods*, 9(7):671–675, June 2012.

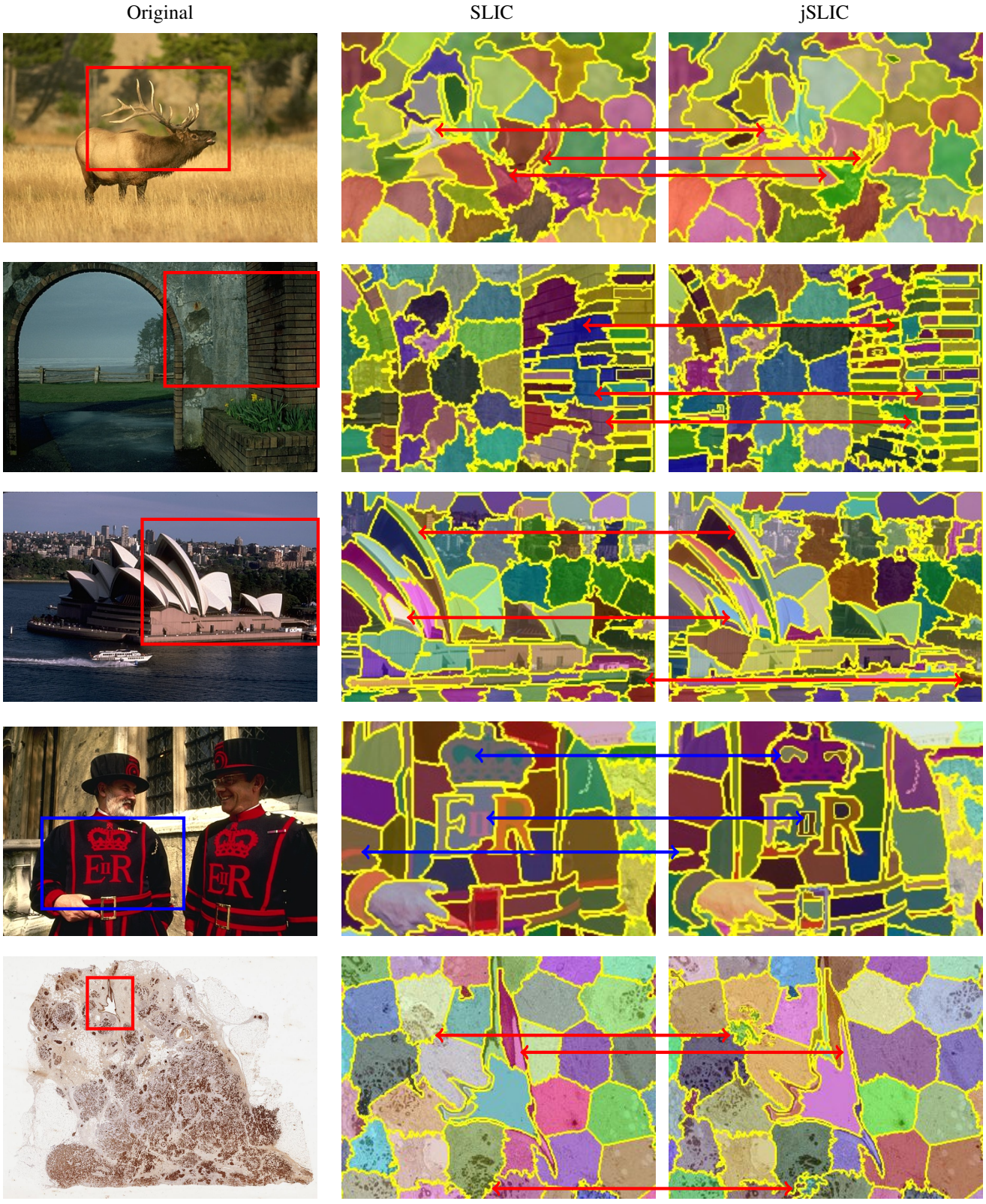


Figure 4: We run the original SLIC (middle) and jSLIC (right) on the Berkeley Segmentation Dataset [8] and some stained histological images using the same configuration for both. To present the differences we chose from each image only a part/detail where improvements can be easily seen (the rest of the image is usually segmented equally). The reason for more reliable superpixels by jSLIC is, because it takes into account all neighbouring connected components and their similarity by colour.

Computer Vision Winter Workshop 2014, Zuzana Kúkelová and Jan Heller (eds.)
Křtiny, Czech Republic, February 3–5
Czech Society for Cybernetics and Informatics

A bi-level view of inpainting - based image compression

Yunjin Chen, René Ranftl, and Thomas Pock

Institute for Computer Graphics and Vision,
Graz University of Technology, Austria
{cheny,ranftl,pock}@icg.tugraz.at

Abstract *Inpainting based image compression approaches, especially linear and non-linear diffusion models, are an active research topic for lossy image compression. The major challenge in these compression models is to find a small set of descriptive supporting points, which allow for an accurate reconstruction of the original image. It turns out in practice that this is a challenging problem even for the simplest Laplacian interpolation model. In this paper, we revisit the Laplacian interpolation compression model and introduce two fast algorithms, namely successive preconditioning primal dual algorithm and the recently proposed iPiano algorithm, to solve this problem efficiently. Furthermore, we extend the Laplacian interpolation based compression model to a more general form, which is based on principles from bi-level optimization. We investigate two different variants of the Laplacian model, namely biharmonic interpolation and smoothed Total Variation regularization. Our numerical results show that significant improvements can be obtained from the biharmonic interpolation model, and it can recover an image with very high quality from only 5% pixels.*

1 Introduction

Image compression is the task of storing image data in a compact form by reducing irrelevance and redundancy of the original image. Image compression methods roughly fall into two main types: lossless compression and lossy compression. In this paper, we focus on lossy compression methods. The objective of lossy compression methods is to reduce the original image data as much as possible while still providing a visually acceptable reconstruction from the compressed data. Lossy image compression can be handled with two different approaches: (1) reducing the data in the original image domain, i.e. by removing a majority of the image pixels; (2) reducing data in a transform domain, such as Discrete cosine transform (DCT) or Wavelet transform. The remaining data (compressed data) is used to reconstruct the original image. It is well known that the former approach is named as image inpainting in the literature [5, 2, 15], and the latter strategy is exploited in the currently widely used standard image compression techniques such as JPEG and JPEG2000 [12, 16]. In this paper, we focus on the strategy of reducing the data in the image domain and then recovering an image from a few data points, i.e., image inpainting.

There are thousands of publications studying the topic of image inpainting in the literature, see e.g., [5, 2, 15, 6] and references therein. In most cases, one does not have influence on the chosen data points. In the context of image inpainting, one usually randomly selects a specific amount of pixels which act as supporting points for the inpainting model, e.g., 5%. In order to get high quality reconstructions in such a scenario, one has to rely on sophisticated inpainting models. However, the task of image inpainting is to recover an image from only a few observations, and therefore, if the randomly selected data points do not carry sufficient information of the original image, even sophisticated inpainting models will fail to provide an accurate reconstruction.

This observation motivated researchers to consider a different strategy for building inpainting based compression models, i.e. to find the optimal data points required for inpainting, given a specific inpainting model. Prior work in this direction can be found in [7, 10, 1, 14, 8, 9]. Belhachmi *et al.* [1] propose an analytic approach to choose optimal interpolation data for Laplacian interpolation, based on the modulus of the Laplacian. The work in [10] demonstrates that carefully selected data points can result in a significant improvement of the reconstruction quality based on the same Laplacian interpolation, when compared to the prior work [1]. However, this approach takes millions of iterations to converge and therefore is very time consuming. The very recent work [8] pushed forward this research topic, where the task of finding optimal data for Laplacian interpolation was explicitly formulated as an optimization problem, which was solved by a successive primal dual algorithm. While their work still requires thousands of iterations to reach a meaningful solution, this new model shed light on the possibility of employing optimization approaches and shows state-of-the-art performance for the problem of finding optimal data points for inpainting based image compression.

The work of [8] is the starting point of this paper. In this paper, we extend the model of finding optimal data for Laplacian interpolation to a more general model, which comprises the model in [8] as a special case. We introduce two novel models to improve the compression performance, i.e., to get better reconstruction quality with the same amount of pixels. Finally, we introduce efficient algorithms to solve the corresponding optimization problems. Namely, we make the following two main contributions in this paper:

(1) We comprehensively investigate two efficient algo-

rithms, which can be applied to solve the corresponding optimization problems, including successive preconditioning primal dual [13] and a recently published algorithm for nonconvex optimization - iPiano [11].

(2) We explore two variants of Laplacian interpolation based image compression to improve the compression performance, namely, a model based on the smoothed TV regularized inpainting model and biharmonic interpolation. It turns out that biharmonic interpolation can lead to significant improvements over Laplacian interpolation.

2 Extension of the Laplacian interpolation based image compression model

The original Laplacian interpolation is formulated as the following boundary value problem:

$$\begin{aligned} -\Delta u &= 0, \quad \text{on } \Omega \setminus I \\ u &= g, \quad \text{on } I \\ \partial_n u &= 0, \quad \text{on } \partial\Omega \setminus \partial I\,, \end{aligned} \tag{1}$$

where g is a smooth function on a bounded domain $\Omega \subset \mathbb{R}^n$ with regular boundary $\partial\Omega$. The subset $I \subset \Omega$ denotes the set with known observations and $\partial_n u$ denotes the gradient of u at the boundary. Δ denotes the Laplacian operator.

It is shown in [10, 8] that the problem (1) is equivalent to the following equation

$$\begin{aligned} c(x)(u(x) - g(x)) - (1 - c(x))\Delta u(x) &= 0, \quad \text{on } \Omega \\ \partial_n u(x) &= 0, \quad \text{on } \partial\Omega \setminus \partial I\,, \end{aligned} \tag{2}$$

where c is the indicator function of the set I, i.e., $c(x) = 1$, if $x \in I$ and $c(x) = 0$ elsewhere. By using the Neumann boundary condition, the discrete form of (2) is given by

$$C(u - g) - (\mathcal{I} - C)\Delta u = 0\,, \tag{3}$$

where the input image g and the reconstructed image u are vectorized to column vectors, i.e., $g \in \mathbb{R}^N$ and $u \in \mathbb{R}^N$, $C = \mathrm{diag}(c) \in \mathbb{R}^{N\times N}$ is a diagonal matrix with the vector c on its main diagonal, $\Delta \in \mathbb{R}^{N\times N}$ is the Laplacian operator and $\mathcal{I}$ is the identity matrix. The underlying philosophy behind this model is to inpaint the region $(\Omega \setminus I)$ by using the given data in region I, such that the recovered image is second-order smooth in the inpainting region, i.e., $\Delta u = 0$.

Note that the inpainting mask c in (3) is binary. However, as shown in [8], equation (3) still makes sense when c is relaxed to a continuous domain such as $\mathbb{R}$. Due to this observation, the task of finding optimal interpolation data can be explicitly formulated as the following optimization problem:

$$\begin{aligned} &\min_{u,c} \frac{1}{2}\|u - g\|_2^2 + \lambda\|c\|_1 \\ &\text{s.t. } C(u - g) - (\mathcal{I} - C)\Delta u = 0\,, \end{aligned} \tag{4}$$

where the parameter λ is used to control the percentage of pixels used for inpainting. When $\lambda = 0$, the optimal solution of (4) is $c \equiv 1$, i.e., all the pixels are used; when $\lambda = \infty$, the optimal solution is $c \equiv 0$, i.e., none of the pixel are used.

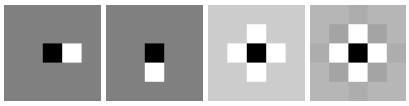


Figure 1: Linear operators shown as filters of size 5×5: from **left** to **right**, ∇_x, ∇_y, Δ and biharmonic operator (Δ^2)

Compared to the original formulation in [8], we omit a very small quadratic term $\frac{\varepsilon}{2}\|c\|_2^2$, because we found that it is not necessary in practice.

Observe that if $c \in \mathcal{C} = [0, 1)^N$, we can multiply the constraint equation in (4) by a diagonal positive-definite matrix $(\mathcal{I} - C)^{-1}$, which results in

$$B(c)(u - g) - \Delta u = 0\,, \tag{5}$$

where $B(c) = \mathrm{diag}(c_1/(1 - c_1), \cdots, c_N/(1 - c_N))$. It is clear that the constraint equation (5) can be equivalently formulated as the following minimization problem

$$u(c) = \arg\min_u \frac{1}{2}\|\nabla u\|_2^2 + \frac{1}{2}\|B(c)^{\frac{1}{2}}(u - g)\|_2^2\,, \tag{6}$$

where ∇ is the gradient operator, and $\Delta = -\nabla^\top\nabla$. Therefore, it turns out that the Laplacian interpolation is exactly the Tikhonov regularization technique for image inpainting, where the first term can be seen as the regularization term based on the gradient operator, and the second term as the data fidelity term.

Now, let us consider how to improve the performance of the regularization based inpainting model (6). The only thing we can change is the regularization term. There are two possible directions: (1) considering higher-order linear operators, e.g., Δ, to replace the first-order derivative operator ∇; (2) replacing the quadratic regularization with more robust penalty functions, such as ℓ_p quasi-norm with $p \in (0, 1]$.

The linear operators ∇ and Δ can be interpreted as linear filters, the corresponding linear filters are shown in Figure 1. If we make use of ∇ in the inpainting model (6), the resulting operator Δ makes the inpainting process only involve the information from its nearest neighborhood; however, if we turn to the Δ operator, the resulting operator Δ^2 (biharmonic operator) can involve more information from larger neighborhood, see Figure 1. In principle, this should bring some improvement of inpainting performance; besides this, the biharmonic operator is mathematically meaningful in itself, implying higher-order smoothness of the solution u.

Regarding the penalty function, quadratic function is known to generate over smooth results, especially for edges, and therefore many other edge-aware penalty functions have been proposed. A straightforward extension is to make use of the ℓ_1 norm, which leads to the well-known Total Variation (TV) regularization (still convex model). Since exact TV regularization suffers from the drawback of piece-wise constant solutions, we employ the following smoothed version of TV regularization, which is parameterized by a small smoothing parameter ε:

$$\|\nabla u\|_\varepsilon = \sum_{i=1}^{N} \sqrt{(\nabla_x u)_i^2 + (\nabla_y u)_i^2 + \varepsilon^2}\,,$$

where $\nabla_x u$ and $\nabla_y u$ denote the gradient in x direction and y direction, respectively. We will show in the next section that this smooth technique is also necessary for optimization.

Using these considerations, we arrive at a general formulation of the inpainting-based image compression model, which is given by the following bi-level optimization problem:

$$\begin{aligned} &\min_{c\in\mathcal{C}} \frac{1}{2}\|u(c)-g\|_2^2 + \lambda\|c\|_1 \\ &\text{s.t. } u(c) = \arg\min_u \mathcal{R}(u) + \frac{1}{2}\|B(c)^{\frac{1}{2}}(u-g)\|_2^2\,, \end{aligned} \tag{7}$$

where the upper level problem is defined as the trade-off between the sparsity of the chosen data and the reconstruction quality, while the lower-level problem is given as the regularization based inpainting model. In the lower-level problem, $\mathcal{R}(u)$ defines a regularization on u, and in this paper we investigate three different regularizers

$$\mathcal{R}(u) = \begin{cases} \frac{1}{2}\|\nabla u\|_2^2 & \text{Laplacian interpolation} \\ \frac{1}{2}\|\Delta u\|_2^2 & \text{Biharmonic interpolation} \\ \|\nabla u\|_\varepsilon & \text{Smoothed TV regularization} \end{cases} \tag{8}$$

3 Efficient algorithms for solving inpainting based image compression problems

In the prior work [8], a successive primal dual algorithm was used in order to solve the Laplacian interpolation based image compression problem (7), where tens of thousands inner iterations and thousands of outer iterations are required to reach convergence. Since this is too time consuming for practical applications, we first investigate efficient algorithms to solve problem (7).

3.1 Successive Preconditioning Primal Dual algorithm (SPPD)

A straightforward method to accelerate the algorithm in [8] is to make use of the diagonal preconditioning technique [13] for the inner primal dual algorithm, while keeping the outer iterate unchanged. The basic principle of the successive primal dual algorithm, is to linearize the constraint of (7), i.e., the lower-level problem. For smooth regularization terms $\mathcal{R}(u)$, the lower-level problem of (7) can be equivalently written using its first-order optimality conditions:

$$T(u,c) = \frac{\partial \mathcal{R}(u)}{\partial u} + B(c)(u-g) = 0. \tag{9}$$

Using Taylor expansion, we linearize (9) around a point $(\hat{u},\hat{c})$:

$$T(u,c) \approx T(\hat{u},\hat{c}) + \left(\frac{\partial T}{\partial u}\Big|_{\hat{u}}\right)^\top (u-\hat{u}) + \left(\frac{\partial T}{\partial c}\Big|_{\hat{c}}\right)^\top (c-\hat{c}) = 0. \tag{10}$$

Let $(\hat{u},\hat{c})$ be a feasible point of constraint (9), i.e., $T(\hat{u},\hat{c}) = 0$, and substitute the linearized constraint back into the initial problem (7), we arrive at the following constrained optimization problem

$$\begin{aligned} &\min_{c\in\mathcal{C},u} \frac{1}{2}\|u-g\|_2^2 + \lambda\|c\|_1 + \frac{\mu_1}{2}\|c-\hat{c}\|_2^2 + \frac{\mu_2}{2}\|u-\hat{u}\|_2^2 \\ &\text{s.t. } D_u u + D_c c + q = 0\,, \end{aligned} \tag{11}$$

where $D_u = \left(\frac{\partial T}{\partial u}\big|_{\hat{u}}\right)^\top$, $D_c = \left(\frac{\partial T}{\partial c}\big|_{\hat{c}}\right)^\top$, $q = -D_u\hat{u} - D_c\hat{c}$. Note that the linearized constraint is only valid around a small neighborhood of $(\hat{u},\hat{c})$, and therefore we have to add two additional penalty term $\frac{\mu_1}{2}\|c-\hat{c}\|_2^2$ and $\frac{\mu_2}{2}\|u-\hat{u}\|_2^2$ to ensure that the solution (u^*,c^*) is in the vicinity of $(\hat{u},\hat{c})$. The saddle-point formulation of (11) is written as

$$\begin{aligned} \max_p \min_{(u,c)} \left\langle K\begin{pmatrix}u\\c\end{pmatrix} + q, p\right\rangle + \frac{1}{2}\|u-g\|_2^2 + \lambda\|c\|_1 + \\ \frac{\mu_1}{2}\|c-\hat{c}\|_2^2 + \frac{\mu_2}{2}\|u-\hat{u}\|_2^2 + \delta_{\mathcal{C}}(c)\,, \end{aligned} \tag{12}$$

where $K = (D_u, D_c)$, $\delta_{\mathcal{C}}(c)$ is the indicator function of set $\mathcal{C}$, and $p \in \mathbb{R}^N$ is the Lagrange multiplier associated with the equality constraint in (11).

Remark 1. Note that for Laplacian and biharmonic interpolation, we do not restrict c to the set $\mathcal{C}$, and we make use of the original constraint in (4), i.e.,

$$C(u-g) - (\mathcal{I}-C)Lu = 0\,,$$

where $L = -\Delta$ for Laplacian interpolation, and $L = -\Delta^2$ for biharmonic interpolation. Therefore, the indicator function $\delta_{\mathcal{C}}(c)$ in equation (12) can be dropped for these models. However, for the TV regularized model or other possible regularization techniques, we have to strictly rely on (12).

Remark 2. It was stated in previous work [8] that there is no need to introduce an additional penalty term for variable u, because u continuously depends on c. However, we find that for biharmonic interpolation, we have to keep the penalty term for u, otherwise, the resulting algorithm will suffer from zigzag behavior when it gets close to the optimal solution.

It is easy to work out the Jacobi matrices D_u and D_c for Laplacian and biharmonic interpolation, which are given as

$$\begin{cases} D_u(\hat{u},\hat{c}) = \operatorname{diag}(\hat{c}) - (\mathcal{I} - \operatorname{diag}(\hat{c}))L, \\ D_c(\hat{u},\hat{c}) = \operatorname{diag}(\hat{u} - g + L\hat{u})\,. \end{cases}$$

For smooth TV regularized inpainting model, the constraint (9) is written as

$$\nabla^\top \cdot \begin{pmatrix} \frac{\nabla_x u}{\rho} \\ \frac{\nabla_x u}{\rho} \end{pmatrix} + B(u-g) = 0\,,$$

where $\rho = \sqrt{\nabla_x^2 u + \nabla_y^2 u + \varepsilon^2}$, and $\nabla = \binom{\nabla_x}{\nabla_y}$. The Jacobi matrices D_u and D_c are given by

$$\begin{cases} D_c(\hat{u},\hat{c}) = \operatorname{diag}(\frac{1}{(1-c)^2}) \cdot \operatorname{diag}(u-g), \\ D_u(\hat{u},\hat{c}) = \binom{\nabla_x}{\nabla_y}^\top \cdot \operatorname{diag}\left(\frac{\frac{\nabla_y^2 u+\varepsilon^2}{\rho^3}}{\frac{\nabla_x^2 u+\varepsilon^2}{\rho^3}}\right) \cdot \binom{\nabla_x}{\nabla_y} - \\ \qquad \binom{\nabla_y}{\nabla_x}^\top \cdot \operatorname{diag}\left(\frac{\frac{\nabla_x u \odot \nabla_y u}{\rho^3}}{\frac{\nabla_x u \odot \nabla_y u}{\rho^3}}\right) \cdot \binom{\nabla_x}{\nabla_y} + B\,, \end{cases} \tag{13}$$

where $\odot$ denotes point-wise multiplication.

We make use of the diagonal preconditioning technique of [13] to choose the preconditioning matrices Γ and Σ.

$$\Gamma = \operatorname{diag}(\tau),\ \Sigma = \operatorname{diag}(\sigma)\,,$$

Algorithm 3.1 Preconditioning PD for solving problem (12)

(1) Compute the preconditioning matrices Γ and Σ and choose $\theta \in [0,1]$

(2) Initialize (u,c) with $(\hat{u},\hat{c})$, and $\bar{p}=0$.

(3) Then for $k \geq 0$, update (u^k,c^k) and p^k as follows:

$$\begin{cases} p^{k+1} = p^k + \Sigma\left(K\binom{u^k}{c^k} + q\right) \\ \bar{p}^{k+1} = p^{k+1} + \theta(p^{k+1}-p^k) \\ \binom{u^{k+1}}{c^{k+1}} = (\mathcal{I}+\Gamma\partial G)^{-1}\left(\binom{u^k}{c^k} - \Gamma K^\top \bar{p}^{k+1}\right) \end{cases} \tag{14}$$

where $\tau_j = \frac{1}{\sum_{i=1}^N |K_{i,j}|^{2-\gamma}}$, $\sigma_i = \frac{1}{\sum_{j=1}^{2N}|K_{i,j}|^\gamma}$. The we employ the preconditioning primal dual Algorithm 3.1 to solve problem (12).

For Laplacian and biharmonic interpolation, the function $G(u,c)$ in (14) is given as

$$G(u,c) = \frac{1}{2}\|u-g\|_2^2 + \frac{\mu_2}{2}\|u-\hat{u}\|_2^2 + \lambda\|c\|_1 + \frac{\mu_1}{2}\|c-\hat{c}\|_2^2.$$

It turns out that the proximal map with respect to G simply poses point-wise operations, which is given as

$$\begin{aligned} \binom{u}{c} = (\mathcal{I}+\Gamma\partial G)^{-1}\binom{\tilde{u}}{\tilde{c}} \Longleftrightarrow \\ \begin{cases} u_i = \frac{\tilde{u}_i + \tau_i^1 g_i + \mu_2\tau_i^1\hat{u}_i}{1+\tau_i^1+\mu_2\tau_i^1} \quad i = 1\cdots N \\ c_i = \text{shrink}_{\frac{\lambda\tau_i^2}{1+\tau_i^2\mu_1}}\left(\frac{\tilde{c}_i+\tau_i^2\mu_1\hat{c}_i}{1+\tau_i^2\mu_1}\right), \end{cases} \end{aligned} \tag{15}$$

where the soft shrinkage operator is given by $\text{shrink}_\alpha(x) = \text{sgn}(x)\cdot\max(|x|-\alpha,0)$, and $\tau = \binom{\tau^1}{\tau^2}$.

For smooth TV regularization, the function G is given by

$$G(u,c) = \frac{1}{2}\|u-g\|_2^2 + \frac{\mu_2}{2}\|u-\hat{u}\|_2^2 + \lambda\sum_{i=1}^N c_i + \frac{\mu_1}{2}\|c-\hat{c}\|_2^2 + \delta_{\mathcal{C}}(c).$$

The proximal map for u is the same as in (15), the solution for c can be computed by

$$c_i = \text{Proj}_{\mathcal{C}}\left(\frac{\tilde{c}_i + \tau_i^2\mu_1\hat{c}_i - \tau_i^2\lambda}{1+\tau_i^2\mu_1}\right)$$

3.2 iPiano

Observe that in problem (7) the lower-level problem can be solved for u, and the result can be substituted back into the upper-level problem. It turns out that this results in an optimization problem which only depends on the variable c. It is demonstrated in our previous work [11] that this optimization problem can be solved efficiently by using the recently proposed algorithm - iPiano. Our experiments will show that this strategy is more efficient than the successive preconditioning primal dual algorithm.

For Laplacian and biharmonic interpolation, we can solve u in closed form, i.e., $u = A^{-1}Cg$. This results in the following optimization problem, which only depends on variable c:

$$\min_c \frac{1}{2}\|A^{-1}\text{diag}(c)g - g\|_2^2 + \lambda\|c\|_1\,, \tag{16}$$

where $A = C + (C-\mathcal{I})L$. Casting (16) in the form of iPiano algorithm, we have $F(c) = \frac{1}{2}\|A^{-1}\text{diag}(c)u - g\|_2^2$, and $G(c) = \lambda\|c\|_1$. As shown in [11], the gradient of F with respect to c is given as:

$$\nabla F(c) = \text{diag}(-(\mathcal{I}+L)u + g)(A^\top)^{-1}(u-g)\,.$$

For smooth TV regularization, $F(c) = \frac{1}{2}\|u(c)-g\|_2^2$, $u(c)$ is the solution of the lower-level TV regularized inpainting model. In order to calculate the gradient of F with respect to c, we can make use of the implicit differentiation technique, see [3] for more details. The gradient is given as

$$\nabla F(c)\big|_{c^*} = -D_c(u^*,c^*)(D_u(u^*,c^*))^{-1}(u^*-g)\,,$$

where u^* is the optimal solution of the lower-level problem in (7) at point c^*. As stated in [3], in order to get an accurate gradient $\nabla F(c)$, we need to solve the lower-level problem as accurately as possible. To that end, we exploit Newton's method to solve the lower-level problem.

Now we can make use of iPiano to solve this optimization problem. The algorithm is summarized below:

Algorithm 3.2 iPiano for solving problem (12)

(1) Choose $\beta \in [0,1)$, $l_{-1} > 0$, $\eta > 1$, and initialize $c^0 = 1$ and set $c^{-1} = c^0$.

(2) Then for $n \geq 0$, conduct a line search to find the smallest nonnegative integers i such that with $l_n = \eta^i l_{n-1}$, the following inequality is satisfied

$$\begin{aligned} F(c^{n+1}) \leq F(c^n) + \left\langle \nabla F(c^n), c^{n+1}-c^n\right\rangle \\ + \frac{l_n}{2}\|c^{n+1}-c^n\|_2^2\,, \end{aligned} \tag{17}$$

where c^{n+1} is calculated from (18) by setting $\beta = 0$. Set $l_n = \eta^i l_{n-1}$, $\alpha_n < 2(1-\beta)/l_n$, and compute

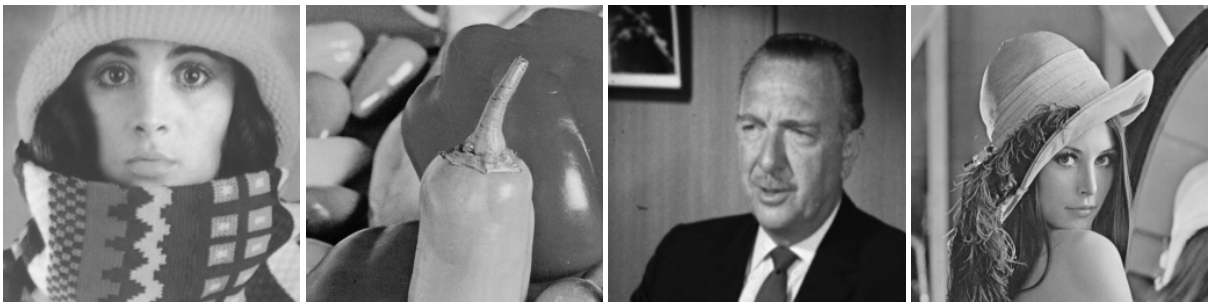
$$c^{n+1} = (I+\alpha_n\partial G)^{-1}(c^n - \alpha_n\nabla F(c^n) + \beta(c^n - c^{n-1}))\,. \tag{18}$$

4 Numerical experiments

In this section, we first discuss how to choose an efficient algorithm for solving the model (7) for different cases. Then we investigate the inpainting performance for different models under the unified assumption that we only make use of 5% pixels. All the experiments were conducted on a server with Intel X5675 processors (3.07GHz), and all the investigated algorithms were implemented in pure Matlab code. We exploited three different test images ("Trui", "Walter" and "Peppers") which are also used in previous works [10, 8].

4.1 Implementation details

In our implementation, the parameter γ of preconditioning technique is chosen as $\gamma = 10^{-6}$. For the SPPD algorithm, the parameter μ_1 and μ_2 are set as follows: (1) for the Laplacian interpolation based compression model, $\mu_1 =$



(a) Trui (b) Peppers (c) Walter (d) Lena

Figure 2: Four test images used in our experiments

$0.05,\ \mu_2 = 0$; (2) for biharmonic interpolation based model, $\mu_1 = 0.1,\ \mu_2 = 0.2$; and (3) for smoothed TV based model, $\mu_1 = 0.05,\ \mu_2 = 0.1$. The set $\mathcal{C}$ is defined in the range of $[0,\ c_{max}]$ with $c_{max} = 1 - 10^{-6}$.

For the iPiano algorithm, we make use of the following parameter settings:

$$l_{-1} = 1,\ \eta = 1.2,\ \beta = 0.75,\ \alpha_n = 1.99(1-\beta)/l_n\,.$$

In order to exploit possible larger step size in practice, we use the following heuristic: If the line search inequality (17) is fulfilled, we decrease the evaluated Lipschitz constant L_n slightly by using a factor 1.02, i.e., setting $l_n = l_n/1.02$.

4.2 Choosing appropriate algorithm for each individual model

For Laplacian interpolation based compression model, we found that when using the proposed preconditioning technique, the required iterations can be reduced to about 150 outer iterations and 2000 inner iterations, which is a tremendous decrease compared to prior work [8]. However, for this problem, the iPiano algorithm can do better. Our experiments show that usually 700 iterations are already enough to reach a lower energy. Concerning the run time, the SPPD algorithm needs about 2400s, but iPiano only takes about 622s. We conclude that iPiano is clearly a better choice for solving the Laplacian interpolation based compression model.

Let us turn to the biharmonic interpolation based compression model. Even though the linear operator is only slightly changed, when compared to the Laplacian model, it turns out that the corresponding optimization problem becomes much harder to solve. The SPPD algorithm still works for this problem; however, as mentioned before, we have to introduce an additional penalty term on variable u, otherwise the convergence behavior is very bad. Besides, we have to run the algorithm much longer, usually about 300 outer iterations and 4000 inner iterations. For the iPiano algorithm applied to this case, we have to significantly increase the amount of required iterations, typically, we have to run about 3500 iterations to reach convergence.

For the biharmonic interpolation based compression model (16), the difference between the results obtained by above two algorithms becomes more obvious. For instance, for the test image "Trui" with parameter $\lambda = 0.0028$, by using the SPPD algorithm, we arrive a final energy of 15.34; however, the final energy of iPiano is much lower, about 13.48, which basically implies that iPiano solves the corresponding optimization problem better. Concerning the run time, for this case, iPiano takes more computation time than Laplacian interpolation case. There are two reasons: (1) the amount of required iterations is increased by a factor of 5; (2) for iPiano, we have to solve two linear equation $Ax = b$ and $A^\top x = b$ in each iteration and line search[1], which becomes much more time consuming from Laplacian to biharmonic interpolation. Therefore, for this case, both algorithms show a similar runtime (about 5000s). Since iPiano achieves a lower energy with similar computational effort this algorithm is preferable for the biharmonic model.

For the case of smoothed TV regularization, it becomes even harder to solve the lower-level problem and thus more time consuming. It is therefore advisable not to make use of iPiano. The SPPD algorithm is a better choice for this model. Solving smoothed TV regularization based model also needs about 5000s.

4.3 Reconstruct an image only using $\sim$5% pixels

We evaluate the performance of three considered compression models based on three test images. For each individual model, we search optimal data points used for inpainting with the same amount of about 5%, and then reconstruct an image by using these optimal points. In order to control the sparsity of selected data points to be 5% approximately, we have to carefully choose the parameter λ for each model and for each processing image. The found optimal mask c is continuous, and then we binarize it by a threshold parameter $\varepsilon_T = 0.01$.

Concerning the measurement of reconstruction quality, we make use of the mean squared error (MSE) to keep consistent with previous work, which is given by

$$MSE(u,g) = \frac{1}{N}\sum_{i=1}^{N}(u_i - g_i)^2\,.$$

The MSE is computed with the assumption that the image gray value is in the range of [0, 255]. As shown in previous work [8], for Laplacian interpolation, it is straightforward to consider an additional post-processing step, which is called

[1] In our implementation we use the Matlab "backslash" operator.

(a) 10% random chosen data (b) Smoothed TV (276.37) (c) Laplacian (244.48) (d) Biharmonic (208.92) (e) Learned prior (165.01)

Figure 3: Inpainting results of the degraded "Lena" image with 10% randomly chosen pixels by using different methods. The number in the bracket is the resulting MSE. For randomly selected points, the inpainting model with learned MRF prior gives the best reconstruction result.

gray value optimization (GVO) to further improve the reconstruction quality. We also consider this strategy for Laplacian and biharmonic interpolation after binarising the mask c, which is formulated as following optimization problem

$$\arg\min_{x\in\mathbb{R}^M} \|A^{-1}S^\top x - g\|_2^2\,, \tag{19}$$

where A is defined in the same way as in (16). $S \in \mathbb{R}^{M\times N}$ is the sampling matrix derived from the diagonal matrix $\mathrm{diag}(c)$ by deleting the rows whose elements are all zero. M is the number of points in the mask c with a value of 1. Obviously, (19) is a least squared problem, which has the closed form solution

$$x = \left(S(A^\top)^{-1}A^{-1}S^\top\right)^{-1} S(A^\top)^{-1}g.$$

However, in practice it turns out that this computation is very time consuming because we have to calculate A^{-1} explicitly. Therefore, we turn to L-BFGS algorithm to solve this quadratic optimization problem.

For smoothed TV regularization model, we also consider this GVO post-processing step, which is given by the following bi-level optimization problem

$$\begin{aligned}&\min_{x\in\mathbb{R}^M} l(x) = \frac{1}{2}\|u(x) - g\|_2^2 \\ &\text{s.t. } u(x) = \arg\min_u \|\nabla u\|_\varepsilon + \frac{1}{2}\|B^{\frac{1}{2}}(u - S^\top x)\|_2^2\,,\end{aligned} \tag{20}$$

where the sampling matrix $S \in \mathbb{R}^{M\times N}$ is the same as in (19). We also make use of L-BFGS to solve this problem. To that end, we need to calculate the gradient of l with respect to x, which is given as

$$\nabla l(x)\big|_{x^*} = -D_x(u^*, x^*)(D_u(u^*, x^*))^{-1}(u^* - g)\,,$$

where D_u is the Hessian matrix given in (13), $D_x = -SB$, u^* is the solution of the lower-level problem at point x^*.

We summarize the results in Figure 4. One can see that starting from the initial Laplacian interpolation based image compression model, we can achieve significant improvements of inpainting performance for all test images by using biharmonic interpolation based model, at the expense of computation time; however, switching to the smoothed TV regularization based model can not bring any improvement even with more computation time. To the best of our knowledge, concerning the inpainting performance of the biharmonic interpolation model, it is the first time to achieve such an accurate reconstruction by using only 5% pixels.

5 Conclusion and future work

In this paper, we extended the Laplacian interpolation based image compression model to more general inpainting based compression model. Starting from the Laplacian interpolation, we investigated two variants, namely biharmonic interpolation and smoothed TV regularization inpainting model, to improve the compression performance. In order to solve the corresponding optimization problems efficiently, we introduced two fast algorithms: (1) successive preconditioning primal dual algorithm and (2) a recently proposed nonconvex optimization algorithm - iPiano. Based on these algorithms, for each model, we found the most useful 5% pixels, and then reconstructed an image from the optimal data. Numerical results demonstrate that (1) biharmonic interpolation gives the best reconstruction performance and (2) the smoothed TV regularization model can not generate superior results over the Laplacian interpolation method.

Future work consists of two aspects: (1) more efficient algorithm to solve the corresponding optimization problems. Even though the introduced algorithms are fast, they are still very time consuming for complicated models, e.g., biharmonic interpolation and smoothed TV regularization models. (2) exploiting more sophisticated inpainting models to further improve the compression model. A possible candidate is to make use of the inpainting model with a learned MRF prior [3, 4], which is shown to work well for image inpainting with randomly selected points. Figure 3 presents an example to show the inpainting performance of the learned model for randomly selected data points. One can see that in this random case, the inpainting model with learned MRF prior can generate the best result, and therefore, we believe that it can achieve better result for image compression.

References

[1] Zakaria Belhachmi, Dorin Bucur, Bernhard Burgeth, and Joachim Weickert. How to choose interpolation data in images. *SIAM Journal on Applied Mathematics*, 70(1):333–352, 2009.

[2] Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. Image inpainting. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 417–424. ACM Press/Addison-Wesley Publishing Co., 2000.

[3] Y.J. Chen, T. Pock, R. Ranftl, and H. Bischof. Revisiting loss-specific training of filter-based MRFs for image restoration. In *German Conference on Pattern Recognition (GCPR)*, 2013.

[4] Y.J. Chen, R. Ranftl, and T. Pock. Insights into analysis operator learning: from patch-based models to higher-order MRFs. *To appear in IEEE Transactions on Image Processing*, 2014.

[5] Antonio Criminisi, Patrick Pérez, and Kentaro Toyama. Region filling and object removal by exemplar-based image inpainting. *Image Processing, IEEE Transactions on*, 13(9):1200–1212, 2004.

[6] Julia A Dobrosotskaya and Andrea L Bertozzi. A wavelet-laplace variational technique for image deconvolution and inpainting. *Image Processing, IEEE Transactions on*, 17(5):657–663, 2008.

[7] I. Galic, J. Weickert, M. Welk, A. Bruhn, A. G. Belyaev, and H.-P. Seidel. Image compression with anisotropic diffusion. *Journal of Mathematical Imaging and Vision*, 31(2-3):255–269, 2008.

[8] L. Hoeltgen, S. Setzer, and J. Weickert. An optimal control approach to find sparse data for Laplace interpolation. In *International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)*, pages 151–164, 2013.

[9] M. Mainberger, A. Bruhn, J. Weickert, and S. Forchhammer. Edge-based compression of cartoon-like images with homogeneous diffusion. *Pattern Recognition*, 44(9):1859–1873, 2011.

[10] M. Mainberger, S. Hoffmann, J. Weickert, C. H. Tang, D. Johannsen, F. Neumann, and B. Doerr. Optimising spatial and tonal data for homogeneous diffusion inpainting. In *International Conference on Scale Space and Variational Methods in Computer Vision (SSVM)*, pages 26–37, 2011.

[11] Peter Ochs, Yunjin Chen, Thomas Brox, and Thomas Pock. ipiano: Inertial proximal algorithm for non-convex optimization. *Preprint*, 2013.

[12] William B Pennebaker. *JPEG: Still image data compression standard*. Springer, 1992.

[13] Thomas Pock and Antonin Chambolle. Diagonal preconditioning for first order primal-dual algorithms in convex optimization. In *International Conference on Computer Vision (ICCV 2011)*, 2011.

[14] C. Schmaltz, J. Weickert, and A. Bruhn. Beating the quality of JPEG 2000 with anisotropic diffusion. In *DAGM-Symposium*, pages 452–461, 2009.

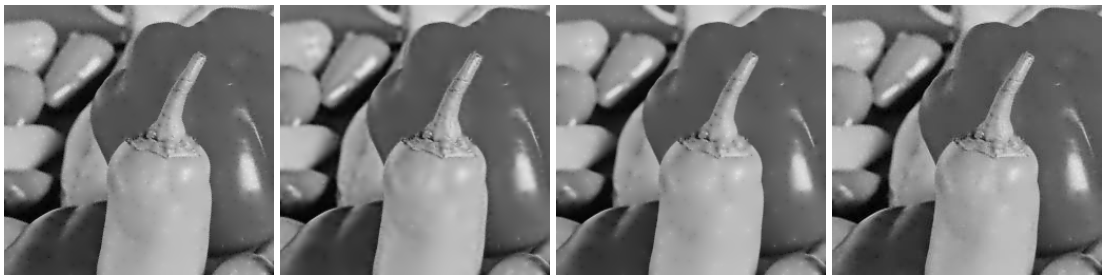
[15] Jianhong Shen. Inpainting and the fundamental problem of image processing. *SIAM news*, 36(5):1–4, 2003.

[16] D Taubman and MW Marcellin. JPEG2000: Image compression fundamentals, practice and standards. *Massachusetts: Kluwer Academic Publishers*, pages 255–258, 2002.

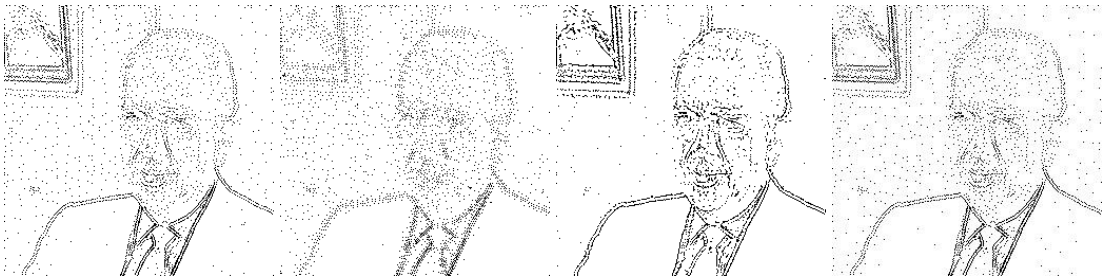
(a) 4.98% (b) 4.98% (c) 6.90% (d) 4.95%

(e) MSE: 16.89 (f) MSE: 10.60 (g) MSE: 17.98 (h) MSE: 16.95

(i) 4.84% (j) 4.89% (k) 5.69% (l) 5.02%



(m) MSE: 18.99 (n) MSE: 17.81 (o) MSE: 21.14 (p) MSE: 18.44



(q) 4.82% (r) 4.59% (s) 5.86% (t) 5.00%

(u) MSE: 8.03 (v) MSE: 4.85 (w) MSE: 10.52 (x) MSE: 7.59

Computer Vision Winter Workshop 2014, Zuzana Kúkelová and Jan Heller (eds.)
Křtiny, Czech Republic, February 3–5
Czech Society for Cybernetics and Informatics

Violent Scenes Detection based on Automatically-generated Mid-level Violent Concepts

Shinichi Goto[1] and Terumasa Aoki[2]

[1]Graduate School of Information Sciences, Tohoku University, Miyagi, Japan
s-goto@riec.tohoku.ac.jp

[2]New Industry Creation Hatchery Center, Tohoku University, Miyagi, Japan
aoki@riec.tohoku.ac.jp

Abstract *Violent scenes detection in videos is a challenging problem because of the ambiguity of the word "violence." In this paper we introduce Mid-level Violence Clustering to solve this problem. Assuming three resource layers exist, it automatically generates mid-level violent concepts to infer violence without manually annotated tags of violent concepts such as fire, fights, etc. Our work is based on the combination of visual and audio features with machine learning at fixed segment-level. Multiple Kernel Learning is applied so that multimodality of data can be maximized, and finally a violence-score for each shot is calculated. We trained the whole system on a dataset from MediaEval 2013 Affect Task and evaluated it by its official metric MAP@100. The obtained results outperformed the best score in Affect Task.*

1 Introduction

Violent scenes detection is a task to detect violent actions in videos. It has been gathering attention just as MediaEval Affect Task [10] represents, which is intended to detect violent scenes in movies. MediaEval is a benchmarking workshop dedicated to evaluating systems for multimedia analysis and retrieval, including Affect Task, in which Technicolor [1] proposes the need of a system which enables users to choose movies that are suitable for their children by providing a preview of violent segments beforehand. Though even children can easily reach violent contents on the Internet nowadays, manually tagging or removing them is almost impossible because of their enormous number. This fact also makes it essential to develop the automatic classification system for violent videos.

The performance of previously proposed systems for violent scenes detection, however, is still unsatisfactory because of its complexity, as well as its ambiguous definition: e.g. Chen et al. defines violence as "a series of human actions accompanying with bleeding" in [5], though Giannakopoulos et al. defines it as "violent-related classes such as shots, fights and screams" in [11]. Simultaneously, rather than simply classifying each segment, it is required to claim which segment is more violent. This is the difference from general video classification problem. As a matter of fact, in Affect Task participants are asked to submit scores for violent segments.

The purpose of this study is to propose a novel system for shot-level violence classification and scoring in videos and to compare it with other algorithms. We use the definition of violence by 2013 Affect Task, which is "*physical violence accident resulting in human injury or pain*." Our system is based on fixed segment-level processing, which means first videos are divided into segments, each of which contains a fixed number of frames. Both of visual and audio feature vectors for each segment are extracted, and they are used to train classifiers. In order to make the most use of multimodality of data, Multiple Kernel Learning is applied to our system. In addition, Mid-level Violence Clustering is proposed in order for mid-level violent concepts to be learned automatically, without using manually annotated tags of concepts such as *fire*, *fight*, etc. "Mid-level" means this layer lies between low-level features and high-level final targets. Classifiers produce segment-level violence-scores, and finally they are converted to shot-level scores. Our system is trained and tested on a dataset from 2013 Affect Task, and evaluated by its official metric $MAP@100$. The effectiveness of Mid-level Violence Clustering is evaluated, and fusion methods are compared as well. We also compare our results with results by other participants who did not use external data. Finally, an investigation for each mid-level violence cluster is performed for further understanding.

2 Previous Work

Relatively few researches have been done for violent scenes detection. Some works used only audio information such as energy entropy and zero crossing [11], or utilized only visual features such as Bag-of-Visual-Words (BoVW) [6], Space Time Interest Points (STIP) [14] and camera motion [4, 5]. After extracted usually they are fed as input for Machine Learning such as Support Vector Machine (SVM) to give classification on test videos.

On the other hand, adopting both of visual and audio features is the current mainstream and have shown to improve results. The work by Nam et al. at 1998 [18] utilized this multimodality, proposing that violent signatures are represented as the combination of multiple features. Their fea-

ture extraction is based on flame detection, blood detection and audio features. In [15] PLSA was adopted to locate audio violence. PLSA is a probabilistic model utilizing the Expectation Maximization algorithm. For visual violence they used a linear weighted model fed with the results of violent event detection such as motion intensity, frame, explosion and blood. Finally co-training is carried out to utilize both modalities. Penet et al. compared two modality-fusion methods, namely Early Fusion and Late Fusion [20]. Early Fusion concatenates features from both modalities before machine learning, while Late Fusion fuses probabilities of both modalities already calculated. They reported Late Fusion was superior to Early Fusion. Derbas et al. [17] proposed Joint Audio-Visual Words representation, which constructs a codebook in the context of Bag-of-Words (BoW) by combining audio and visual features. Dai et al. [8] used external data from ImageNet and MIT scene dataset in addition to usual training and testing videos, in order to detect part-level attributes in each frame, each of which is expected to represent the likelihood of containing a certain object. Combining them with other low-level features from both of visual and audio modalities, the SVM classifier is built.

Researches above tried to detect violence directly from low-level features. Instead, some works have used violent concepts such as *fire*, *fights* and so on. Those concepts are manually annotated by humans and given in MediaEval Affect Task [10]. Ionescu et al. proposed a frame-level violence prediction, applying a multi-layer perceptron in order to utilize these concepts [12, 23]. They put the first layer for the concept prediction, and the second layer for the violence prediction. In addition to those provided concepts, Tan and Ngo [26] have utilized extra 42 violence concepts such as bomb and war from ConceptNet [16]. ConceptNet is composed of nodes representing concepts in the form of words or short phrases with their relationships. On their system those extra concepts are trained using YouTube videos which are crawled additionally. Afterwards a graphical model of those concepts are generated, and Conditional Random Fields [28] refines it by using relationships in ConceptNet and co-occurrence information of concepts. Their $MAP@100$ result was the first place in 2013 Affect Task with external data.

3 Violent Scenes Detection Based on Mid-level Violence Clustering

3.1 Approach Overview

Fig. 1 illustrates the overview of our approach. Feature extraction and training/classification are carried out at fixed segment-level. Here we define a segment as a sequence of 20 frames, and this means its time length is for 0.8 seconds if FPS is 25. The reason 20 is chosen is that if its length is too short, it might lack statistical meaning for its features, especially for its audio features. Or if the length is too long, features might get affected too much by changes of environments in scenes, such as a switch from a violent scene to a non-violent scene or camera motion.

First all training videos are divided into segments. Then both of visual and audio features are extracted for each segment (described in 3.2). Segments tagged as violent are gathered and divided into $K(> 0)$ clusters. As described in 3.3, we assume that each mid-level cluster implicitly represents a concept or a combination of concepts led to violence. Multiple Kernel Learning (MKL) is applied to generate a classifier for each cluster.

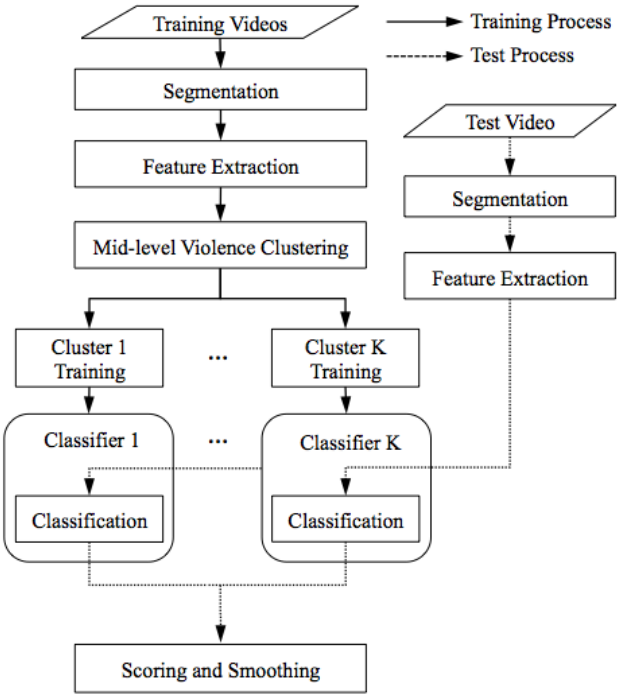


Figure 1: The overview of our approach.

In the test process, segmentation and feature extraction are performed in the same way as the training process. Classifiers in all clusters evaluate each segment, producing violence-scores. Then scores are integrated to generate one segment-level score for that segment. Smoothing is applied in order to take the context of videos into account, and finally segment-level scores are converted to shot-level scores. The following sections explain our feature vectors and training system more precisely.

3.2 Low-level Feature Extraction

Recent works on violent scenes detection such as [26] and [8] have shown the effectiveness of using trajectory-based features as visual information and MFCC-based features as audio information. Similar to those researches, in total six feature spaces exist on our system: Trajectory, HOG, MBHx, MBHy, RGB and Audio.

Dense Trajectory

Trajectories have been used to capture local motion of videos, especially in the field of action recognition. We use Dense Trajectory [29], a trajectory to which dense sampling is applied. Except for those in homogeneous areas, densely sampled points are tracked by calculating optical flows in each spatial scale until they reach the length of $L = 15$ frames. Every frame newly sampled points are added if no tracked point is found in the neighborhood of each pixel. We use 32 for a neighbour

range, 5 for a sampling step, and 6 for a spatial scale size. Displacement vectors of trajectories are extracted for both of x-direction and y-direction and concatenated (30-dimension). Following [29], descriptors are extracted around each trajectory: HOG, MBHx, MBHy, and RGB-histogram. Although originally HOF (Histograms of Oriented Optical Flow) is extracted as well, expected to have poor contribution on our task because of its frequent camera motion, it is removed.

HOG

HOG (Histograms of Oriented Gradients) is a descriptor for local gradient orientations, and is largely used for object detection. In the same way as [29], the neighbour of trajectories are divided into $2 \cdot 2$ areas. For each area 8-dimensional HOG is calculated and averaged every 5 frame. Since each trajectory has 15 frames length, concatenating all of them generates $12 (= 2 \cdot 2 \cdot 3)$ histograms, resulting in $8 \cdot 12 = 96$ dimensional vectors.

MBHx and MBHy

MBH (Motion Boundary Histograms) was originally proposed in the field of human detection by Dalal et al. [9] to represent the changes in the optical field, namely local motion information independent of camera motion, by calculating the gradient of the optical flow. MBH is generated separately along the vertical direction (MBHx) and the horizontal direction (MBHy). Since each MBH is represented as an 8-dimensional vector, similar to HOG, both of MBHx and MBHy are described as 96-dimensional vectors around trajectories.

RGB-histogram

Although originally RGB information are not extracted in [29], in violence detection since color information is expected to be helpful just as blood and flame detection have contributed to the results in some previous researches [15, 18], 64-bin RGB histograms around trajectories are calculated every 5 frame. Spatial division is not carried out for RGB-histogram, and it results in a 192-dimensional vector.

Audio

Similar to Bag-of-Audio-Words in [19], MFCC (Mel Frequency Cepstrum Coefficients) and the log energy are first extracted every 10ms with 5ms overlap for audio features. The first derivative of MFCC and its energy are also calculated as delta-MFCC, producing a 26-dimensional feature.

Trajectory-based features are assigned to a segment in which their trajectory has reached 15 frames length. For each segment these features are gathered, converted to the BoW form by using already calculated codebooks, and normalized. Codebooks are generated by using randomly selected 100,000 features and k-means++ algorithm beforehand in each feature space respectively. Finally 200-dimensional Trajectory, 400-dimensional HOG, 200-dimensional MBHx, 200-dimensional MBHy, 400-dimensional RGB histogram, and 200-dimensional Audio vectors are obtained.

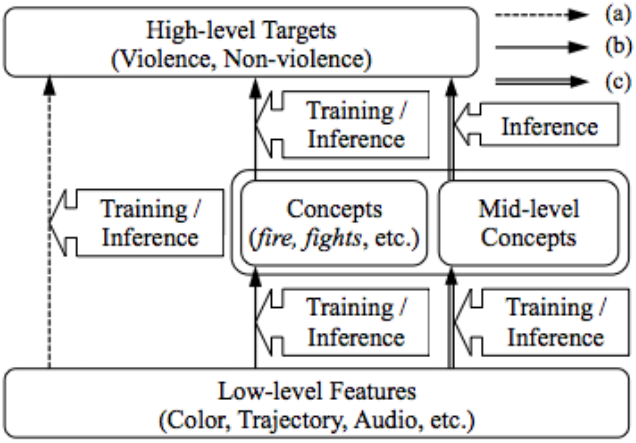


Figure 2: Three layers in violence detection and our actions: (a) trains and infers violence directly from low-level features, (b) trains and infers violent concepts using annotations first, and uses them to train and infer violence, (c) trains and infers mid-level concepts without annotations of concepts first, and uses them to infer violence. Our system follows (c).

3.3 Mid-level Violence Clustering

We assume there are three layers in violence detection as Fig. 2 displays. Most of previous works only used low-level features to directly train and infer high-level targets, namely Violence and Non-violence (Fig. 2(a)). Some works have started using manually annotated violent concepts. Using those given concepts, they train and infer concepts in test videos to train and infer violence finally (Fig. 2(b)). The reason why this mid-level layer is needed that the diversity of "violence" is huge: even though two segments are annotated as violent, their low-level features might be largely different depending on their characteristics of violence. For instance, although explosion scenes labelled as violent might have distinctive visual features, those of scream scenes might not similar even if they are also labelled as violent. Our system, however, takes the approach Fig. 2(c). Instead of actual violent concepts, it detects violence using *mid-level concepts* which have been automatically inferred without annotations of violent concepts. We apply Mid-level Violence Clustering and generate clusters for mid-level concepts prediction.

Fig. 3 illustrates the process of Mid-level Violence Clustering. First all of the violent segments in training videos are gathered. Then they are divided into $K (> 0)$ clusters, each of which is expected to contain similar segments. Then non-violent training segments are assigned to those clusters sequentially, whose results construct clusters for mid-level violence classifiers. After concatenating feature vectors of them, k-means++ algorithm with Euclidean distance is applied to generate clusters. Here we assume that feature vectors for violent segments are capable of representing one or multiple concepts related to violence in each cluster. If mid-level concepts are correctly clustered, "training violence in one cluster" corresponds to "training one or multiple mid-level violent concepts." This means manual annotations for violent concepts are unnecessary if our previous assumption is correct. Mid-level Violence Clustering also contributes to reduction in complexity. Since the number of feature points

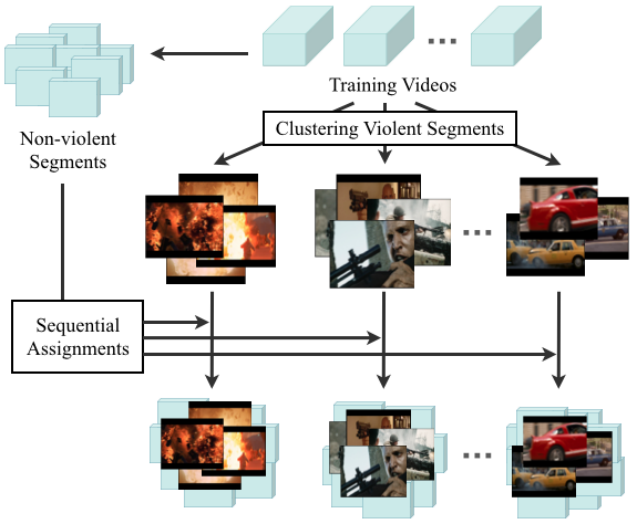


Figure 3: The process of Mid-level Violence Clustering.

and dimensions are huge in the task of Violent Scenes Detection, the cost for training is expensive. Training multiple classifiers needs much less time compared to training one classifier using all feature vectors. The process of actual training, classification and scoring are described in the following sections.

3.4 Multiple Kernel Learning

BoVW with SVM [6] has contributed greatly to the filed of image classification over the last few years. For the task of violence detection, however, multiple feature spaces have to be handled, and then simply concatenating feature vectors and training classifiers might not always be the best way, according to the work by Penet et al. [20]. As they studied, when multimodal features exist, there are two available fusion schemes: Early Fusion (EF) and Late Fusion (LF). EF concatenates features from both modalities before training, meaning it can take correlations of those feature spaces into account while training, though it is dealing with each feature space uniformly. On the other hand, on LF training and classification are performed for each feature space independently. Generated results, which are violence probabilities in their experiment, are fused afterwards. They compared EF with LF when two modalities exist (visual and audio). Although they concluded that LF has more effectiveness, if more modalities exist just as our system, it has some drawbacks: 1) it cannot take correlations of multimodal features into account while training and classification, 2) how to fuse both results has to be decided manually beforehand. Besides, in [20] low-level audio features and higher-level visual features such as shot length were directly combined, and this might have resulted in poor correlations between two modalities.

To cope with this problem and to maximize the multimodality of data, we apply Multiple Kernel Learning (MKL), which can be regarded as a kind of EF, but aims at finding optimized weights for each feature space when multiple SVM kernels are applied [25]. This means MKL considers correlations of multiple feature spaces while training and classification, but at the same time it considers differences of violent characteristics among multiple feature spaces. In MKL the whole kernel is composed of multiple sub-kernels, and is defined as a following equation:

$$\boldsymbol{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_p \beta_p \boldsymbol{K}_p(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{1}$$

where $\boldsymbol{K}_p$ are sub-kernels, and β_p is a weight for p-th sub-kernel. Sub-kernels for Trajectory, HOG, MBHx, MBHy, RGB-histogram and Audio are prepared in our case. The dual for the MKL primary problem is proposed by Bach et al. [2] and parameters can be learned.

Histogram Intersection Kernel (HIK), which has been reported to perform well on histogram-based features [3], is adopted as a sub-kernel. HIK is defined as follows:

$$\boldsymbol{K}_{int}(\boldsymbol{A}, \boldsymbol{B}) = \sum_{i=1}^{m} min(a_i, b_i) \tag{2}$$

where $\boldsymbol{A} = [a_1, a_2, \ldots, a_m]$ and $\boldsymbol{B} = [b_1, b_2, \ldots, b_m]$. It measures the degree of similarity between two histograms. MKL is applied to all K clusters, generating K classifiers. SHOGUN Toolbox [24] is used for our MKL implementation.

3.5 Scoring and Smoothing

Each segment in test videos is classified as violent or non-violent by K classifiers. If a video has N segments, results obtained by k-th classifier $(1 \leq k \leq K)$ are:

$$\boldsymbol{C}_k = [c_{1,k}, c_{2,k}, \ldots, c_{N,k}] \tag{3}$$
$$\boldsymbol{D}_k = [d_{1,k}, d_{2,k}, \ldots, d_{N,k}] \tag{4}$$

where $c_{n,k} \in \pm 1$ $(1 \leq n \leq N)$ represents that n-th segment is violent if its value is $+1$, while it represents non-violent if its value is -1. $d_{n,k}$ denotes a distance between a feature point of n-th segment and a hyperplane which has classified it. Using $\boldsymbol{D}_k$, we define $\boldsymbol{S}_k$, scores for all segments by k-th classifier as follows:

$$\boldsymbol{S}_k = [s_{1,k}, s_{2,k}, \ldots, s_{N,k}], \tag{5}$$
$$s_{n,k} = \begin{cases} d_{n,k} \ (if \ c_{n,k} = +1) \\ 0 \ (if \ c_{n,k} = -1) \end{cases} \tag{6}$$

They are integrated to produce pre-final scores $\boldsymbol{S}$:

$$\boldsymbol{S} = [s_1, s_2, \ldots, s_N], \tag{7}$$
$$s_n = \frac{\sum_{l=1}^{K} s_{n,l}}{K_{vio}} (1 \leq n \leq N) \tag{8}$$

where K_{vio} is the number of classifiers whose $c_{n,k}$ is +1, in other words, the number of classifiers which classify n-th segment as violent. This means for each segment, if no cluster classifies it as violent, its violence-score is zero, while the mean value of violence-scores of classifiers which classify it as violent is assigned if one or more clusters classify it as violent.

In order to take the context of a video into account, scores are smoothed as a final step. Although in [8] the average value over a three-shot window is calculated, we adopt a

moving average calculation so that the further neighbour segments are positioned, the lesser their effects are considered. Smoothed scores $\boldsymbol{S}'$ are calculated by using pre-final scores $\boldsymbol{S}$ as follows:

$$\boldsymbol{S}' = [s'_1, s'_2, \ldots, s'_N], \tag{9}$$

$$s'_i = \frac{s_i + \sum_{m=1}^{M} \alpha^n \cdot (s_{i-m} + s_{i+m})}{2M+1} \tag{10}$$

where α $(0 < \alpha < 1)$ is a smoothing coefficient, and M is a neighbor range around a segment. We used 0.5 for α and 2 for M.

Scores for shots are calculated by converting segment-level scores after calculating frame-level scores. Because the numbers of frames in segments are consistent except for a final segment of a video, frame-level scores are simply given as scores for segments which have those frames. Then for each shot, scores for frames it contains are summed and divided by the number of frames. This score is used as a final score for each shot.

4 Experiment

Though multiple tasks exist in 2013 Affect Task, we focus on shot-level violence detection in movies with objective definition, which is "*physical violence accident resulting in human injury or pain*," without external data. For the evaluation, shot-level violence-scores have to be generated rather than merely classifying shots.

In order to ascertain the improvement by Mid-level Violence Clustering, multiple numbers of K are tried. Additionally a system with clusters constructed for each training movie instead of using Mid-level Violence Clustering is tested, in which multiple violent concepts are mixed in each cluster. We call this as Training Movie Grouping, and compare it with Mid-level Violence Clustering. Results are also compared with runs by other participants. To evaluate the effect of MKL, EF and LF are performed using normal SVM with HIK. Though various kinds of implementations are possible for LF, on our system visual (Trajectory, HOG, MBHx, MBHy and RGB) features are concatenated, and trained separately from audio features. Since results by classifiers are scores rather than probabilities, simply a higher score is used as a score by that cluster.

4.1 Dataset

With automatically generated shot boundaries by Technicolor's software [10], 18 training movies and 7 test movies are provided. (Training movies: *Armageddon, Billy Elliot, Eragon, Harry Potter 5, I am Legend, Leon, Midnight Express, Pirates of the Caribbean 1, reservoir Dogs, Saving Private Ryan, The Sixth Sense, The Wicker Man, Kill Bill 1, The Bourne Identity, The Wizard of Oz, Dead Poets Society, Fight Club and Independence Day*. Test movies: *Fantastic Four, Fargo, Forrest Gump, Legally Blond, Pulp Fiction, The God Father 1 and The Pianist*.) Training movies are given with frame-level violence ground truth annotated by several human assessors. Though in 2013 Affect Task participants were allowed to use prepared violent concepts, our algorithm uses only low-level features extracted from movies, violence ground truth and shot boundaries. To reduce the complexity, all frames are resized to half of their original size as pre-processing.

4.2 Evaluation Metric

MediaEval 2013 Affect task adopted Mean Average Precision at the 100 top ranked violent shots ($MAP@100$) as its official metric. MAP is the most standard evaluation of ranked retrieval results among the TREC (Text Retrieval Conference) community [7], and it provides a single-figure measure of quality across recall levels. MAP is the mean value of the Average Precision (AP), which can consider the order which targets are presented in. It computes the average value of $Precision$ over the interval from $n = 1$ to $n = N$:

$$AP@N = \frac{1}{N} \sum_{n=1}^{N} Precision(\boldsymbol{R}_n) \tag{11}$$

where N is the maximum rank number one wants to calculate, and $\boldsymbol{R}_n$ is the set of ranked retrieval violent segments from the top result to the n-th result. Then MAP is calculated as follows:

$$MAP@Q = \frac{1}{Q} \sum_{q=1}^{Q} AP@q \tag{12}$$

For instance, if ranked results are judged as $[true, false]$, $AP@1 = 1 \cdot 1 = 1$, $AP@2 = (1 \cdot 1 + 1 \cdot 0.5)/2 = 0.75$, and $MAP@2 = (AP@1 + AP@2)/2 = (1 + 0.75)/2 = 0.875$.

4.3 Results and Discussion

For the number of clusters K, we tried every 5 numbers from 5 to 120. Fig. 4 displays these results before smoothing since smoothing improved scores largely, especially when the numbers of clusters were low, making it difficult to evaluate the effectiveness of Mid-level Violence Clustering. Results with small numbers of clusters seem to be unstable compared to results with high numbers. This reveals the effectiveness of Mid-level Violence Clustering, since a small number means there are not enough clusters to represent violent concepts. This figure also compares them with a result from Training Movie Grouping, and scores by Mid-level Violence Clustering were superior to its score. On Training Movie Grouping, each cluster might have multiple violent concepts whose feature vectors can be largely different each other, and then it also proves the effect of Mid-level Violence Clustering.

While we expected low scores for results with high numbers of clusters (e.g. $K = 100$), they seem to be able to keep promising scores. This is because when the number of clusters was high, some clusters had only a few violent segments assigned to themselves due to their anomalies. For instance, when we chose $K = 100$, the smallest number of violent segments in one cluster was 2, although other clusters tended to contain about 50-100 violent segments. Even though this cluster could not find violent segments, it did not affect a final score either because our scoring equation (6) depends only on scores by classifiers that have classified a target segment as violent.

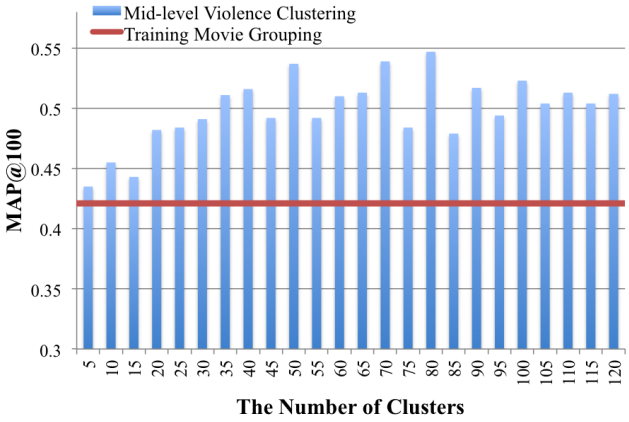


Figure 4: Results of Mid-level Violence Clustering with multiple numbers of clusters without smoothing, and a result of Training Movie Grouping.

Run	MAP@100
LIG [17]	0.520
FAR [23]	0.496
Fudan [8]	0.492
NII [13]	About 0.400
Technicolor [21]	0.338
VISILAB [22]	0.150
MTM [27]	0.070
Our system (K=50)	0.558
Our system (K=80)	0.577

Table 1: Comparison with other teams on MediaEval 2013 Affect Task without external data. Our result outperformed them.

Smoothing always improved scores, and then only two best results are shown in Tab. 1 with results by other participants in Affect Task 2013. Although some teams used given concepts for their runs, one can find our score outperforms their scores.

Fusion methods are compared in Fig. 5. All of them are results after the smoothing step. MKL was always better than or equal to EF, and always superior to LF, proving the effectiveness of MKL. Partly because of the difference of how to implement LF, EF was always superior to LF, being different from a report in [20]. This can be considered as being caused by the difference of features. In [20] low-level audio features such as zero crossing rate and energy were used, although higher-level visual features such as shot duration and number of flashes were chosen. This led to few correlations among these two modalities as authors mentioned. In our case, however, since both features were low-level, there seem to have existed correlations among them, which could be maximized when they were combined by applying MKL or EF.

For the run with ($K = 50$), example frames of shots that had high scores are shown in Fig. 6. Among these 4 shots, (a), (b) and (c) were correct estimations for scenes containing explosions, gunshots and car chases. However, the estimation (d) was wrong. In this shot multiple people start

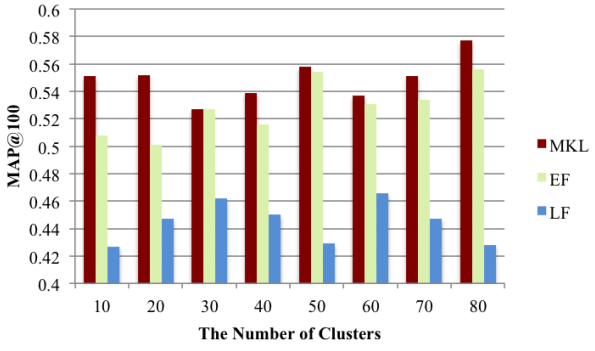


Figure 5: Comparison of fusion methods with smoothing. MKL=Multiple Kernel Learning, EF=Early Fusion, LF=Late Fusion.

standing up suddenly and cheering. Just as this example, shots that contain multiple people, sudden motion and big sound tended to be miss-classified as violent. Meanwhile, common missed violent shots were violent scenes without sound, such as a scene in which a man is wringing other man's neck.

Though our system has achieved promising scores, its performance is still insufficient and multiple points can be argued. The first point to be considered here is that our feature vectors might not be distinct enough. Although we have used trajectory-based features as visual information, they can be easily affected by camera motion. Even though features such as MBH, which are supposed to be robust to camera motion, were extracted, they might be noisy if trajectories themselves are unreliable. Similarly, shot boundaries are not considered on our system although shots often change in the middle of segments and are expected to affect visual features. The second point is that though Euclidean distance is used for the similarity while Mid-level Violence Clustering, Histogram Intersection is used for sub-kernels in MKL. Performing clustering by using Histogram Intersection might be essential to keep consistency.

4.4 Extensive Study

In order to confirm our assumption in 3.3, we examined the amount of violence-related concepts in it using the cluster number $K = 50$. In 2013 Affect Task, participants were provided with violent concepts annotated at frame-level by human assessors. They consist of 7 visual concepts: *presence of blood, presence of fire, fights, gory scenes, presence of firearms, presence of cold weapons car chases*, and 3 audio concepts: *explosions, presence of screams, gunshots*. It should be noted that these concepts are not always related to violence ground truth, and often multiple concepts are tagged in one frame. Since they are at frame-level, we converted them to segment-level annotations by simply tagging each segment if half of frames it contains are annotated.

The ratios of segments annotated by each concept in each cluster are shown in Fig. 7. Since it is inadequate to display ratios for all 50 clusters and for all concepts in this figure due to the limit of the available spaces, only 6 representative clusters are displayed. Also annotation *car chases* is

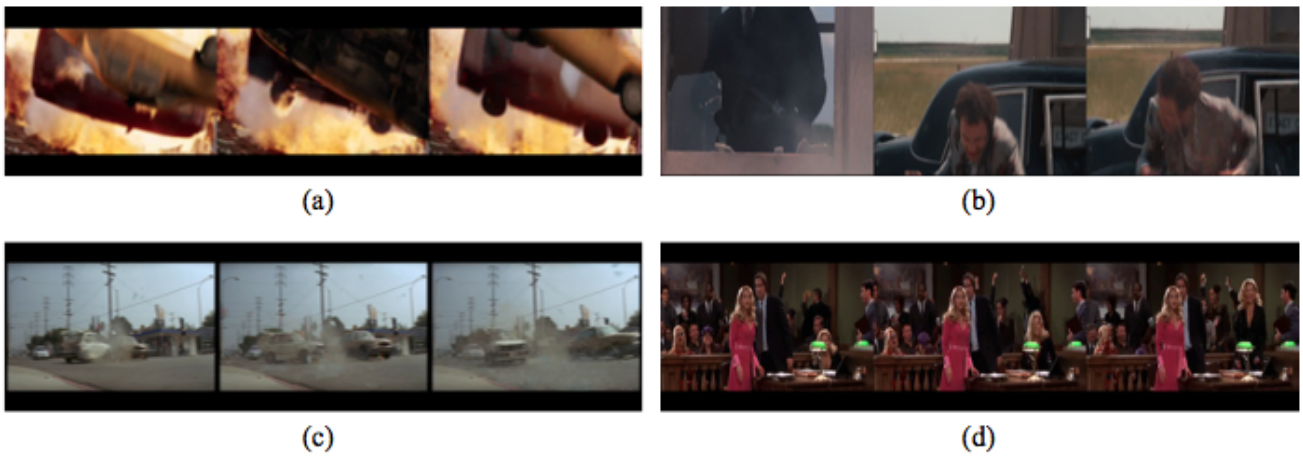


Figure 6: Example frames of shots which had high violence-scores on our system. ($K = 50$): (a) a car is blown away by the explosion, (b) a man is shot multiple times from a window and his shirt gets soaked with blood, (c) a car crashes into another, (d) people start standing up and cheering. Note in (b) the camera perspective seems to change but this is because shot boundaries were automatically generated.

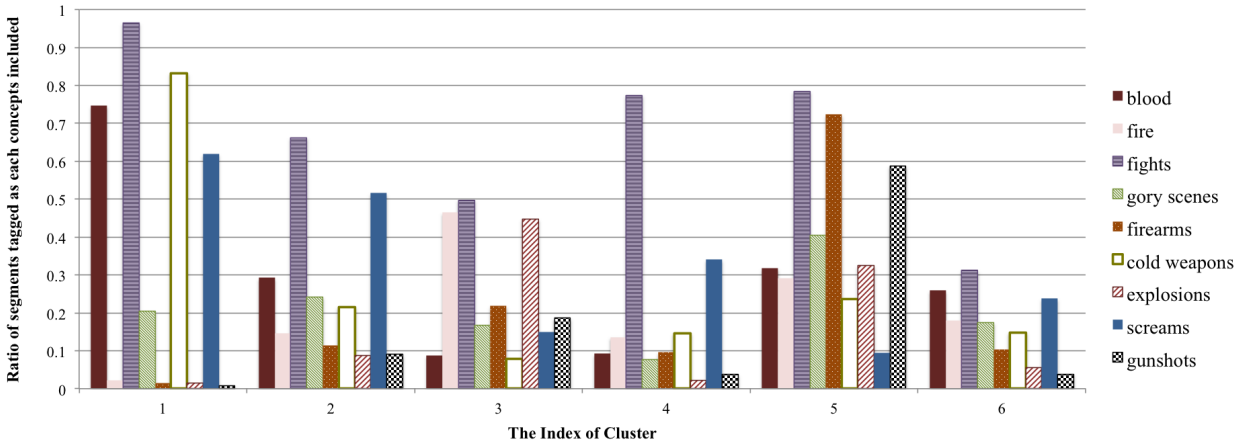


Figure 7: Ratios of segments annotated by each concept for clusters ($K = 50$). Note only 6 representative clusters are shown and a tag "*car chases*" is excluded.

excluded due to its low number. By studying this figure one can find some clusters reflect violent concepts. For instance, although both of Cluster 1 and Cluster 4 have high ratios for *fights*, Cluster 1 has more *blood* and *cold weapons*, meaning these two clusters represent different kinds of violence. Cluster 5 includes a high number of segments tagged as *firearms* and *gunshots*. We investigated this cluster and found it contains gunfire scenes.

On the other hand, there exist clusters which seem not to reflect actual violent concepts like Cluster 6. Though clusters generated by our system and concepts in MediaEval are unrelated essentially, and so characteristics of clusters in Fig. 7 do not always have to be distinctive, it could have been caused by the lack of distinctiveness of features or inconsistency of the clustering method.

5 Summary and Conclusions

In this paper, we proposed a novel system to detect violent scenes in videos by using Mid-level Violence Clustering with multimodal features. Our experiments showed that automatic inference of mid-level concepts is effective for this task, and results outperformed the best $MAP@100$ in MediaEval 2013 Affect Task even without manually annotated concepts. In addition, comparison of fusion methods, as well as investigation for concepts in mid-level violence clusters were performed. Future work is to find more discriminative feature vectors, as well as to adopt more suitable clustering method in the context of our system.

Acknowledgement

We acknowledge MediaEval 2013 Affect Task: Violent Scenes Detection http://www.multimediaeval.org/ for providing a dataset that has been supported, in part, by the Quaero Program http://www.quaero.org.

References

[1] Technicolor. http://www.technicolor.com. Last visited Dec. 2013.

[2] Francis R. Bach, Gert R. G. Lanckriet, and Michael I. Jordan. Multiple Kernel Learning, Conic Duality, and

the SMO Algorithm. In *ICML '04 Proceedings of the twenty-first international conference on Machine Learning*, 2004.

[3] A. Barla, F. Odone, and A. Verri. Histogram Intersection Kernel for Image Classification. In *Proceedings of ICIP 2003.*, pages 513–516, Sept. 2003.

[4] E. Bermejo, O. Deniz, G. Bueno, and R. Sukthankar. Violence Detection in Video Using Computer Vision Techniques. In *CAIP'11 Proceedings of the 14th international conference on Computer analysis of images and patterns - Volume Part II*, pages 332–339, 2011.

[5] Liang-Hua Chen, Hsi-Wen Hsu, Li-Yun Wang, and Chih-Wen Su. Violence Detection in Movies. In *Computer Graphics, Imaging and Visualization (CGIV), 2011 Eighth International Conference*, Aug. 2011.

[6] Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and Cdric Bray. Visual Categorization with Bags of Keypoints. In *Proc. of European Conference on Computer Vision (ECCV)*, pages 1–22, 2004.

[7] Christopher D., Manning, Prabhakar Raghavan, and Hinrich Schutze. In *Introduction to Information Retireval*, 2008.

[8] Qi Dai, Jian Tu, Ziqiang Shi, Yu-Gang Jiang, and Xiangyang Xue. Fudan at MediaEval 2013: Violent Scenes Detection Using Motion Features and Part-Level Attributes. In *Working Notes Proceedings of the MediaEval 2013 Workshop*, Barcelona, Spain, Oct. 2013.

[9] N. Dalal, B. Triggs, and C. Shmid. Human detection using oriented histograms of flow and appearance. In *European conference on computer vision*, 2006.

[10] C. Demarty, C. Penet, M. Schedl, B. Ionescu, V.L. Quang, and Y. Jiang. The Mediaeval 2013 Affect Task: Violent Scenes Detection. In *Working Notes Proceedings of the MediaEval 2013 Workshop*, Barcelona, Spain, October 18-19 2013.

[11] Theodoros Giannakopoulos, Dimitrios Kosmopoulos, Andreas Aristidou, and Sergios Theodoridis. Violence Content Classification Using Audio Features. In *SETN'06 Proceedings of the 4th Helenic conference on Advances in Artificial Intelligence*, pages 502–507, 2006.

[12] Bogdan Ionescu, Jan Schluter, Ionut Mironica, and Markus Schedl. A Naive Mid-level Concept-based Fusion Approach to Violence Detection in Hollywood Movies. In *ICASSP - 37th International Conference on Acoustics, Speech, and Signal Processing*, 2012.

[13] Vu Lam, Duy-Dinh Le, Sang Phan, Shin'ichi Satoh, and Duc Anh Duong. NII-UIT at MediaEval 2013 Violent Scenes Detection Affect Task. In *Working Notes Proceedings of the MediaEval 2013 Workshop*, Barcelona, Spain, Oct. 2013.

[14] I. Laptev. On Space-Time Interest Points. In *International Journal of Computer Vision*, volume 64, pages 107–123, 2005.

[15] Jian Lin and Weiqiang Wang. Weakly-Supervised Violence Detection in Movies with Audio and Video Based Co-training. In *PCM '09 Proceedings of the 10th Pacific Rim Conference on Multimedia: Advances in Multimedia Information Processing*, pages 930–935, 2009.

[16] H. Liu and P. Singh. ConceptNet — a practical commonsense reasoning tool-kit. In *BT Technology Journal*, volume 22, pages 211–226, Oct. 2004.

[17] Bahjat Safadi Nadia Derbas and Georges Quenot. LIG at Mediaeval 2013 Affect Task: Use of a Generic Method and Joint Audio-Visual Words. In *Working Notes Proceedings of the MediaEval 2013 Workshop*, Barcelona, Spain, Oct. 2013.

[18] Jeho Nam, Masoud Alghoniemy, and H. Tewfik. Audio-Visual Content-Based Violent Scene Characterization. In *International Conference on Image Processing*, Oct. 1998.

[19] Stephanie Pancoast and Murat Akbacak. Bag-of-audio-words Approach for Multimedia Event Classification. In *Interspeech Conference*, 2012.

[20] Cedric Penet, Claire-Helene Demarty, Guillaume Gravier, and Patrick Gros. Multimodal Information Fusion and Temporal Integration for Violence Detection in Movies. In *ICASSP - 37th International Conference on Acoustics, Speech, and Signal Processing*, 2012.

[21] Cedric Penet, Claire-Helene Demarty, Guillaume Gravier, and Patrick Gros. Technicolor/INRIA Team at the MediaEval 2013 Violent Scenes Detection Task. In *Working Notes Proceedings of the MediaEval 2013 Workshop*, Barcelona, Spain, Oct. 2013.

[22] Ismael Serrano, Oscar Deniz, and Gloria Bueno. VISILAB at MediaEval 2013: Fight Detection. In *Working Notes Proceedings of the MediaEval 2013 Workshop*, Barcelona, Spain, Oct. 2013.

[23] Mats Sjoberg, Jan Schluter, Bogdan Ionescu, and Markus Schedl. FAR at MediaEval 2013 Violent Scenes Detection: Concept-based Violent Scenes Detection in Movies. In *Working Notes Proceedings of the MediaEval 2013 Workshop*, Barcelona, Spain, Oct. 2013.

[24] Soeren Sonnenburg, Gunnar Raetsch, Sebastian Henschel, Christian Widmer, Jonas Behr, Alexander Zien, Fabio de Bona, Alexander Binder, Christian Gehl, and Vojtech Franc. The SHOGUN Machine Learning Toolbox. In *Journal of Machine Learning Research, 11*, pages 1799–1802, June 2010.

[25] S. Sonnenburg, G. Raetsch, C. Schaefer, and B. Schoelkopf. Large Scale Multiple Kernel Learning. In *Journal of Machine Learning Research*, 2006.

[26] Chun Chet Tan and Chong-Wah Ngo. The Vireo Team at MediaEval 2013: Violent Scenes Detection by Mid-level Concepts Learnt from Youtube. In *Working Notes Proceedings of the MediaEval 2013 Workshop*, Barcelona, Spain, October 18-19 2013.

[27] Bruno Do Nascimento Teixeira. MTM at Mediaeval 2013 Violent Scenes Detection: Through Acoustic-visual Transform. In *Working Notes Proceedings of the MediaEval 2013 Workshop*, Barcelona, Spain, Oct. 2013.

[28] Hanna M. Wallach. Conditional Random Fields: An Introduction. In *Technical Report MS-CIS-04-21*, 2004.

[29] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Action Recognition by Dense Trajectories. In *IEEE Conference on Computer Vision & Pattern Recognition*, pages 3169–3176, Colorado Springs, United States, June 2011.

Computer Vision Winter Workshop 2014, Zuzana Kúkelová and Jan Heller (eds.)
Křtiny, Czech Republic, February 3–5
Czech Society for Cybernetics and Informatics

Over-Segmentation of 3D Medical Image Volumes based on Monogenic Cues

Markus Holzer and Rene Donner

Computational Imaging Research Lab,
Department of Radiology and Nuclear Medicine,
Medical University of Vienna, Austria
markus.holzer, rene.donner@meduniwien.ac.at

Abstract *In this paper, we propose a novel approach to compute 3D supervoxels for radiological image datasets. It allows to cope with the high levels of noise and low contrast encountered in clinical data such as Computed Tomography (CT), Optical Coherence Tomography (OCT) and Magnetic Resonance (MR) images.*

The method, monoSLIC*, employs the transformation of the image content to its monogenic signal as primal representation of the image. The phase of the monogenic signal is invariant to contrast and brightness and by selecting a kernel size matched to the estimated average size of the superpixels it highlights the locally most dominant image edge. Employing an agglomeration step similar to the one used in SLIC superpixels yields superpixels/-voxels with high fidelity to local edge information while being of regular size and shape.*

The proposed approach is compared to state of the art superpixel methods on the real-world images of the 2D Berkley Segmentation Dataset[1] *(BSD) converted to gray-scale, as well as challenging 3D CT and MR volumes of the Visceral*[2] *dataset. It yields a highly regular, robust, homogeneous and edge-preserving over-segmentation of the image / volume while being the fastest approach.*

1 Introduction

The goal of over-segmenting an image is to merge pixels into homogeneous groups of superpixels while preserving boundaries of objects in an image. This allows to perform image analysis tasks on the greatly reduced number of superpixels as opposed to every pixel in the volume. While several approaches to computing superpixels have been proposed in recent years, most were developed with the application to typical 2D color photographs in mind. Radiological image data, on the other hand, has quite different characteristics: monochrome data, high levels of noise and low contrast, both finely detailed structures (e.g. the internal structure of bones) and large objects with only subtle texture differences (liver in a CT). Our aim is to provide a method specifically adapted to these characteristics which is also computationally attractive.

[1] http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/
[2] http://www.visceral.eu/

All but one (Mori [13]) of the existing methods work directly on the image pixel intensities (see Section 2), causing them to be depended on contrast and brightness of the image. The method of [13] extracts texture and contour information, but with high computational costs [9]. In contrast the proposed method extracts structure information using the computationally attractive approach of the monogenic signal. The phase of the monogenic signal contains the local structural information of the image, from which edge cues are extracted and then used as input for k-means clustering. The number of k-means cluster centers is equal to the number of desired superpixels, resulting in a fast and parameter-free (additional to the desired number of superpixels) over-segmentation method that creates regular and smooth superpixels.

In Section 2 the state of the art is summarized and the new method is described in Section 3, showing the definition and incorporation of the monogenic signal into the proposed approach, as well as its extension to 3D data. Section 4 presents the experimental setup and results, with Section 5 providing a conclusion.

2 State of the Art

The state of the art approaches of the over-segmentation problem can be categorized into *graph-partition-based* and *gradient-ascent-based* [2]. Visual examples for each algorithm are presented in Figure 1 for a cropped part of an abdominal CT slice.

Graph-Partition-Based
The methods listed in this category use a graph to represent the similarity between pixels, where each node refers to a pixel and the weight between nodes refers to the similarity of the pixels. The graph is then cut at weights where the difference between nodes is significant, creating a superpixel segmentation.

Felzenswalb [7] proposed an efficient graph-based image segmentation approach, where the image is represented in a 5-D feature space containing spatial information (*x,y*) and color information (*r,g,b*). A nearest neighbor graph is created, with the weights corresponding to the distance in the feature space. The graph is then cut into components representing a minimum spanning tree of associated pixels. The resulting superpixels are perceptually meaningful but there

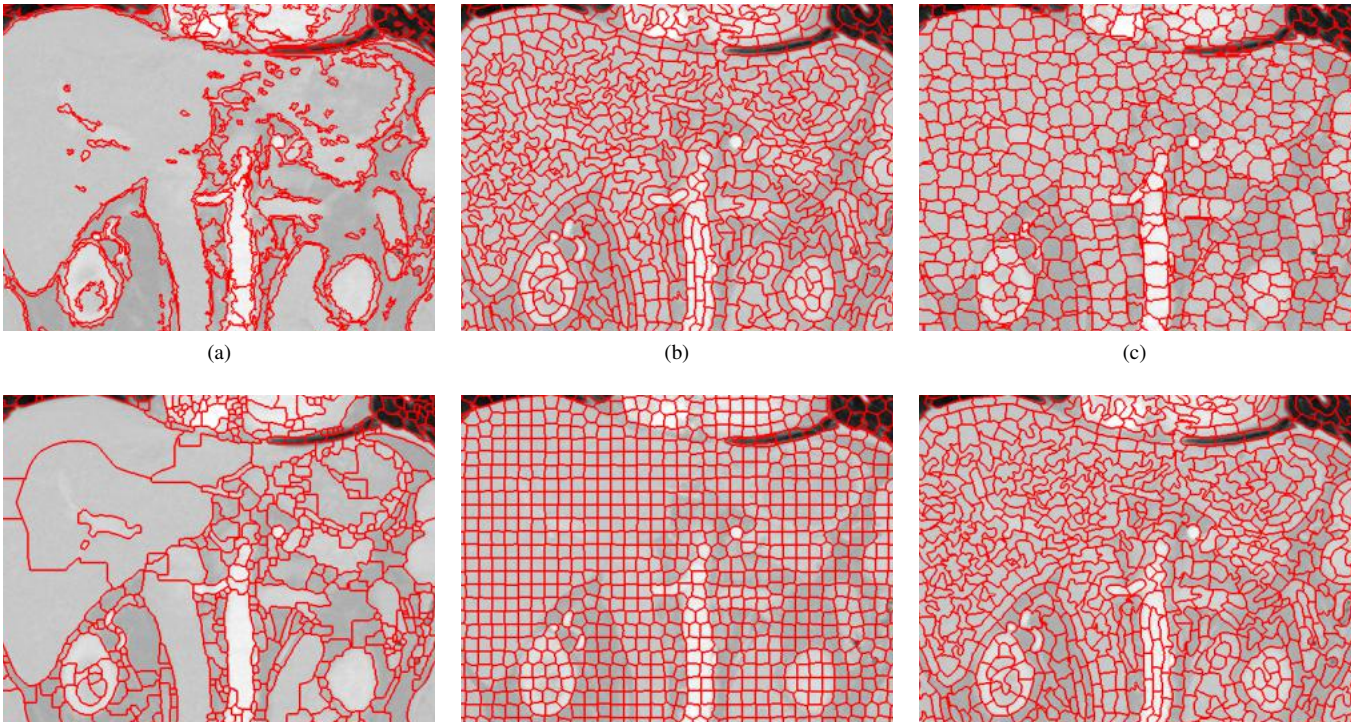


Figure 1: Comparison of the superpixel segmentation on a 2D Computed Tomograph slice of the abdominal region with about 200 pixels per superpixel in average for (a) Felzenswalb [7], (b) Mori [13], (c) Veksler [14], (d) Engel [5], (e) Achanta [1] and (f) MonoSLIC.

are no constraints in terms of superpixel number or size, thus creating irregular ones.

The approach of **Mori** et. al. [13] on the other hand creates regular and visually pleasing superpixels. The quality of its segmentation performance is based on computational intensive extraction of texture and contour features. These are then combined to a final weight matrix creating a graph of size $O(N^2)$ where the normalized graph cut method, proposed by Shi and Malik [10] with a high computational cost of $O(N^{\frac{3}{2}})$ [9], is used to calculate the segmentation.

Similar to the previous method **Veksler** [14] also creates and cuts a graph, but this time an extended grid-graph is created. In this graph each pixel is connected to its neighbors as well as to a number of terminal nodes, each representing a possible superpixel label. A multi-way-graph cut approach is then used to calculate the final segmentation of the image. Two versions of the algorithm are proposed, one for more regular and compact superpixels and another for irregular but more precise superpixels. The latter is named *constant intensity superpixels* and is used for the evalutation in this paper.

Gradient-Ascent-Based In this category methods are used that start by an initial segmentation, which is then improved based on the gradient of the feature space. The boundaries are moved to, or created where, the gradient magnitude of the image is a local maximum.

For the method of **Engel** [5] the initial segmentation is the distribution of seed-points of a watershed transformation. The seed-points and the height-map are calculated by exploiting the properties of the *Gradient Vector Flow* (GVF). Based on the GVF a flux flow field is calculated which serves as the height-map and this height-map is thresholded to generate the initial seed-points. The properties of the superpixel output is similar to [7], with irregular shape and size.

For **Achanta** [2] the initial segmentation is a rectangular grid, which is then iteratively updated to align with local high gradients using the k-means clustering method. The features space for the *Simple Linear Iterative Clustering* (SLIC) method is a 5-D space, which is similar to [7], but instead of the *r,g,b* the *L,a,b* values of the CIELAB color space is used. The distance function for k-means is non-euclidean with a parameter that regulates the weight between *x,y* and *L,a,b* values, giving the user the option to create regular superpixel at the cost of edge precision.

Summary and Additional Approaches A compact summary of the analyzed methods is given in Figure 2. Additional approaches not compared in this paper were proposed, among them the gradient-ascent based approach of Vincent [15], the TurboPixel approach of [9] and the graph-partition-based Lattices approach of [12]. A comparison of these can be found in [2].

3 Methods

The proposed method is designed to over-segment 2D and 3D image data, with a special focus on medical data. Such medical data typically exhibits high levels of noise and low

State of the Art						
Algorithm	Category	Parameter	Preprocessing	Information Representation		Segmentation
Veksler 2010 [13]	graph partition	superpixel count	-	Intensity	Energy Function represented as extended Grid Graph	Energy Minimization as Multi-Way-Graph Cut optimized by α-Expansion
Felzenswalb 2004 [6]	graph partition	significance value	-	Intensity	Grid Graph represented as Boundary Decision Rule	Disjoint-Set Forest optimized by Inverse Ackermann Function
Mori 2005 [12]	graph partition	superpixel count	texture and contour feature extraction	Features	Energy Function represented as Full Graph	Energy minimization as solving a General Eigenvalue System
Engel 2009 [4]	gradient ascent	seed points threshold	edge detection	Edge Information	Gradient Vector Flow Field to compute a Flux Flow Image	Watershed with Seedpoints and Height Map from Flux Flow Field
Achanta 2010 [1]	gradient ascent	superpixel count and regularity	-	Color Information	CIELab Color Space combined with Spatial Information	optimized k-means
Presented Approach						
MonoSLIC 2014	gradient ascent	superpixel count	Monogenic Signal	Local Monogenic Phase	Monogenic Phase combined with Spatial Information	optimized k-means

Figure 2: Overview of the discussed methods.

contrast, as shown on an abdominal CT in Figure 1 and an abdominal MR in Figure 3 (a), as well as the noisy OCT shown in Figure 3 (b). Using the contrast sensitive original gray-level information forces the user to choose a parameter that decides about the level of detail that should be captured. This parameter needs to be tweaked and set appropriately for the methods of [5], [7] and [2]. With the use of the structural information contained in the phase of the monogenic signal, which is brightness and contrast invariant, the proposed method does not require such an parameter. Extracting the structural information also makes the method robust to noise and creates smooth superpixel boundaries.

Our method is similar to the approach of [2], but in our case we perform k-means clustering in a feature space that is spanned by the spatial coordinates *(x,y,z)* and the monogenic phase Λ, which results in 4 dimensional feature space for the 3D case. The properties of the monogenic phase allow for an adaption of the k-means algorithm, that reduces the number of calculations necessary and therefore making our method the fastest 3D approach.

In the following we recapitulate the definition of the monogenic signal, detail how it is employed as additional feature for the k-means clustering and present the changes to the k-means method compared to [2]. In the remainder of this paper the proposed method is called **MonoSLIC**.

The Monogenic Signal The *monogenic signal* proposed by Felsberg [6] originates from the 1D analytic signal [4], which extracts the local phase and local amplitude of a signal. It is based on the Hilbert transform which when applied to $cos(x)$ of a real valued signal x results in $sin(x)$. The Hilbert transform can be therefore understood as a *phase shifter*, shifting every sinosoidal function by -90 degrees. In the frequency domain, with the Fourier transformed signal u the transfer function of the Hilbert transform can be defined as $H(u) = \frac{-iu}{|u|} = -i\,sgn(u)$, where $sgn()$ is the signum function [6]. The kernel can be written as the inverse Fourier transform of the transfer function [17]

$$f_H(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} H(u)e^{iux}du \tag{1}$$

and its combination with the original signal is called the *analytic signal*

$$f_A(x) = f(x) - if_H(x), \tag{2}$$

The Hilbert transform is generalized in [6] using the Riesz transfrom with the transfer function defined in the frequency domain $R(\mathbf{u}) = \frac{-i\mathbf{u}}{|\mathbf{u}|}$, where $\mathbf{u} = (u_1, ..., u_n)^T$, for n dimensional signals [3]. Similar to the analytic signal the monogenic signal is also a combination of the original signal and, this time, its Riesz transformation $f_R(x)$,

$$f_M(x) = f(x) - \mathbf{i}f_R(x), \tag{3}$$

where is $\mathbf{i} = (i_1, ..., i_n)$ dimensional. The monogenic signal is isotropic and also performs a so-called split of identity [6]. This refers to the fact that the signal is decomposed into the local amplitude A_f, the local phase φ and the local orientation θ.

The local amplitude A_f is defined as the norm of the monogenic signal

$$A_f(x) = |f_M(x)| = \sqrt{f^2(x) + |f_R(x)|^2}. \tag{4}$$

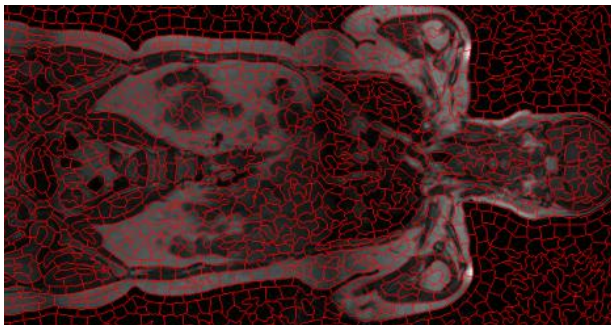
and the local phase

$$\varphi(x) = arg(f_M(x)), \quad \varphi \in [-\pi, \pi] \tag{5}$$

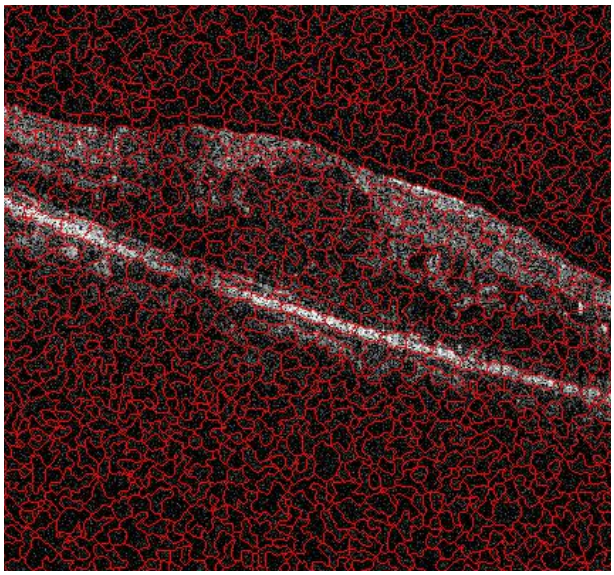
represents the change of local strutural information in the range of $-\pi$ to π [6]. Examples for corresponding structures for a certain φ value are shown in Figure 4 for the 1D case. Finally the geometric information is represented by the orientation in the range of 0 to π

$$\theta(\mathbf{x}) = \arccos(f(\mathbf{x}))/A_f(\mathbf{x}))\,\theta \in [0, \pi] \tag{6}$$

defining the direction of the structure.



(a)



(b)

Figure 3: Example images for 3D medical volumes with the segmentation of MonoSLIC, where (a) shows a Magnetic Resonance Tomography of the thorax and head and (b) an Optical Coherence Tomograph of an eye.

Edge-cues from Monogenic Signal and filter scale selection The monogenic phase $\varphi(x)$ picks up the locally dominant structure in an image regardless of contrast and brightness. Its values are between $-\pi, \pi$ and a value of $\pm\frac{\pi}{2}$ correlates with a strong edge, as previously illustrated in Figure 4. In order to detect this change in structure with the k-means algorithm the values have to be mapped such that there is a high difference between two pixels with different values for $sgn(|\varphi(x)| - \pi/2)$. In other words, to get a representation with similar feature values within superpixels and value changes at their boundaries, the monogenic phase needs to be transformed. We first map φ to $\varphi_{new} = |\varphi| - \frac{pi}{2}$ before calculating the final monogenic phase cue with

$$\Lambda(x) = \frac{sgn(\varphi_{new}) \exp^{(-|\varphi_{new}|)}}{2}, \Lambda \in [-0.5, 0.5]. \quad (7)$$

The wavelength governing the scale of the monogenic signal depends on the number of superpixels and is set to $\lambda = \sqrt[d]{\frac{P}{SP}}$ (for dimension d, the number of pixels P and the number of superpixels SP) which influences the size of the smallest structure that should be detected in the image. λ correlates to the number of superpixels because a small number can only detect large structures and the lower wavelength ensures smoothness balanced with the level of detail, that is desirable at this scale.

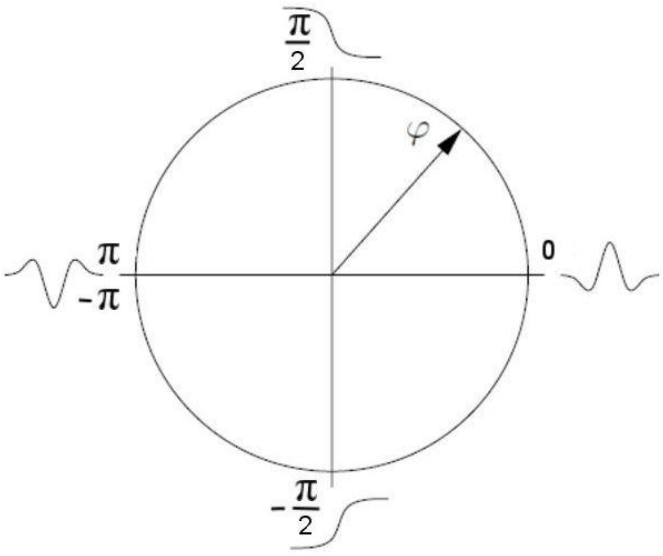


Figure 4: The sketched structure yielding a specific value of φ with a phase wrap from π to $-\pi$. Redrawn from [6].

In the example shown in Figure 5 (c) of the transformed monogenic phase a high contrast can be seen at the change of structure of the original image. Using this as input for k-means will create clusters with are boundaries at the changes of structure. This is the only information about the image employed by our approach.

Super-pixels through clustering Similar to [2], we perform k-means clustering to obtain the superpixels, but instead of using the Lab values color space, we cluster based on the monogenic phase. We initialize the cluster centers (seeds) spatially according to a hexagonal grid with a cluster center distance $d_{cc} = \lambda$ corresponding to the number of superpixels, to avoid imposing too much of a directional preference. Each pixel is represented in an $nD + 1$ space, with the coordinates being the original $n = 2$ pixel or $n = 3$ voxel coordinates. The additional dimension is the monogenic phase, which is scaled by two times the cluster center distance

$$\Lambda(x)_{kmean} = 2d_{cc}\Lambda(x). \quad (8)$$

For the k-means computation we restrict the number of potential pixels for each cluster seed to a $\pm 2d_{cc}$ neighborhood and the maximum number of iterations to $I = 10$, reducing the k-means complexity $O(10SR)$ introduced by [2] to $O(9SN + SR), N \ll R$, where S is the number of seedpoints, R a region around S and N a number of randomly chosen pixels from R. The compact information contained in the monogenic phase makes it possible to choose a random subset of pixels from the region R, reducing the number of computations per iteration.

For the first iteration of the clustering, the initial values for the $nD + 1$'st coordinate of the cluster center have to be

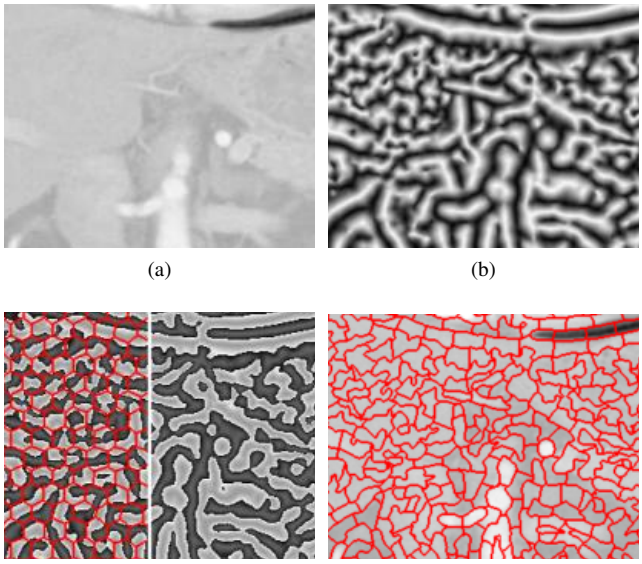


Figure 5: The monogenic phase calculated on an abdominal Computed Tomography image (a) with a filter wavelength of 18.3. In (b) the local phase φ of the image is shown. Panel (c) presents the transformed signal Λ as used by our approach with the overlayed initial cluster centers ($d_{cc} = 18.3$) on the left. In panel (d) the final segmentation is overlayed with the original image.

estimated. These are set to the average of $\Lambda(x)_{kmean}$ of all pixels that are closest (distance $< \frac{d_{cc}}{2}$) to that cluster center.

4 Experiments

In Section 4.1 the setup of the experiments is presented, followed by the results in Section 4.2.

4.1 Setup

The methods presented in this paper are run using their respective reference implementations. The parameters governing the number of superpixels was carefully selected for each case to yield the same number of superpixels, where possible. Parameters specifying edge fidelity versus homogeneity were set to a value such that edge responses are captured, while still maintain regularity. The parameter for [1] was set to 15, which provides a trade-off in terms of regularity and recall and was used throughout this paper. The algorithms are compared in terms of recall, the regularity of their superpixels, their robustness to noise and their runtime.

Two datasets are used for evaluation. The first is the BSD [11] with 500, 0.15 Mega Pixels (MP) real-world images, with objects annotated by five different annotators. The second is the Visceral [8] dataset, which consists of 14 CT and 14 MR volumes. In each of the volumes 20 different anatomies are annotated by medical experts. From the 3D volumes also a 2D dataset is created by extracting the 28 coronal center slices of the dataset with an average size of 0.18 MegaPixel (MP). For the 3D test all the 14 CT and but only 5 abdominal MR volumes are taken, as the method of [1] failed to compute on the other MR volumes.

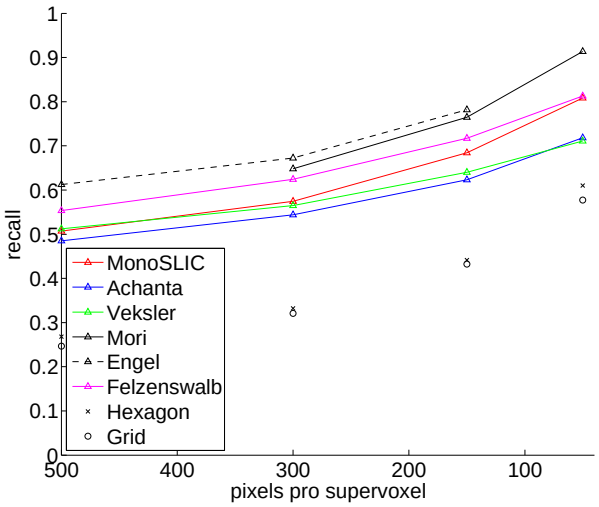


Figure 6: Recall rate for the 2D Visceral image dataset.

4.2 Results

In this section first the results for the 2D Visceral and BSD are presented. The recall rate on the Visceral dataset and the regularity on the BSD dataset are analyzed, before looking at the summary of all acquired statistics. In the following the 3D Visceral results are presented and at the end the run-time performance of the algorithms is evaluated.

2D - Visceral and Berkley Segmentation Dataset The first comparison of the algorithms is the **recall** rate, which describes the percentage of how many of the annotated pixels were detected by the algorithms segmentation. A higher recall value indicates a better detection of the annotated boundaries. The recall rate for the 2D Visceral dataset is measured at defined superpixel sizes shown in Figure 6. The reference over-segmentation using a rectangular and hexagon grid show how a blind over-segmentation would perform. The slightly better results for the hexagon are due to the naturally more meaningful shape compared to a rectangle.

The approaches of Mori [13], Felzenswalb [7] and Engel [5] have the highest recall accuracy followed by MonoSLIC, Veksler [14] and Achanta [1]. The accuarcy of [13] comes at the cost of computation, which is also the reason for the missing first data-point. The methods of [5] and [7] do not allow direct control over the superpixel number. For [5] it was not possible to calculate values for 100 pixels per superpixel due to the properties of the medical images.

The **regularity** of the superpixels is evaluated because smooth superpixel boundaries lower the computation cost of algorithms that compute on the segmentation itself. Another benefit of creating smooth boundaries is that they are visually more pleasing [16], which could be important when presented to physicians. The results are calculated on the BSD, as for regularity comparison this dataset is more representative. That is because medical images contain a high percentage of smooth and homogenous background that would benefit the methods of [1] and [14]. The regluarity is measured

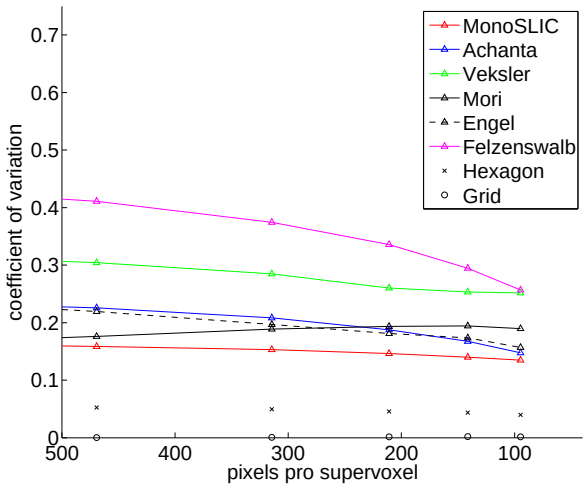


Figure 7: Coefficient of variation for the perimeter over area value on the 2D Berkley Segmentation Dataset images.

with the relation of perimeter *peri* and area *area*, where the final value is $\frac{peri}{\sqrt{area}}$. The mean over all the superpixels of this value is an indicator for the roughness of the boundary, where a lower value means smoother superpixel boundaries. The corresponding standard deviation expresses the consistency of the smoothness, e.g. if there is an high or low variation in superpixel shape. Both measurements are combined in the coefficient of variation, where the mean is divided by the standard deviation. The results are shown in Figure 7 where lower values indicate smoother superpixel boundaries. The MonoSLIC approach has the lowest coefficient of variation and therefor creates the smoothest superpixels for the BSD dataset.

To get a compact view of all the results for the BSD dataset they are **summarized** in Figure 8, where green indicates good performance and desired parameters and red otherwise. The presented columns are recall, recall on images with added Gaussian noise $(mean = 0, std = 0.22)$, standard deviation of the superpixel area in pixels, mean and standard deviation for perimeter over area, the runtime in seconds per MP, if there is a parameter P to set (0 equals no parameter) and if the number of superpixels SP can directly be specified. The values are taken for 70 pixels per superpixel.

Going through the algorithms from bottom to top [7] shows very good runtime and recall rates but the user cannot control the number of superpixel, which also causes a high variation in the area and the regularity of the superpixels. The approach of [5] has the same properties, although a lower recall rate and and higher computation time. The method of [13] achieves the highest recall rate with 84% and 79% for noise, but this comes at a high computational cost, taking about 455s to compute one image. The approach of [14] has a low recall rate and is influenced by noise, the superpixels are of irregular shape and also a relative high runtime of 38s. The recall rate of [1] is a moderate 73% for the normal images but is highly influenced by noise, where the recall rate drops to 60%. The average regularity is the highest of the tested methods but also has a low variation. It has

pixel/SP	recall (%)		area (pixel)	perimeter over area		runtime			
90	normal	noise	std	mean	std	s/MP	P	SP	3D
MonoSLIC	0.72	0.72	796	4.56	0.52	1.6	0	1	1
Achanta	0.73	0.60	922	6.03	0.57	1.6	1	1	1
Veksler	0.71	0.63	4316	5.29	0.87	38	0	1	0
Mori	0.84	0.79	497	4.05	0.72	455.4	0	1	0
Engel	0.71	0.75	7618	5.17	0.57	4.8	1	0	0
Felzenswalb	0.77	0.80	8098	5.29	1.24	0.9	1	0	0
Hexagon	0.54	0.54	314	3.50	0.13	-	-	-	-
Grid	0.50	0.50	36	3.92	0.01	-	-	-	-

Figure 8: Summary of the results for the 2D Berkley Segmentation Dataset images and an over-segmentation of 90 pixels per superpixels, where green indicates desired results.

pixel/SP	recall	area (pixel)	perimeter over area		runtime			
70	normal	std	mean	std	s/MP	P	SP	3D
MonogSLIC	0.68	178	4.57	0.63	1.6	0	1	1
Achanta	0.62	136	4.57	0.38	1.6	1	1	1
Veksler	0.64	401	3.90	0.83	38	0	1	0
Mori	0.76	62	3.95	0.76	455.4	0	1	0
Engel	0.78	4364	4.91	0.67	4.8	1	0	0
Felzenswalb	0.72	6072	5.99	1.76	0.9	1	0	0
Hexagon	0.44	56	3.40	0.13	-	-	-	-
Grid	0.43	10	3.82	0.01	-	-	-	-

Figure 9: Summary of the results for the 2D Visceral images and an over-segmentation of 70 pixels per superpixels, where green indicates desired results.

one of the best run-times and is available in 3D. On the other side it has a parameter that has to be tweaked for a trade-off between compactness. The proposed method MonoSLIC has is very robust to noise with a recall rate of 72% for both the normal and noisy images. It creates very regular superpixels and does not require a parameter to be set.

The summary is also presented for the 2D Visceral dataset in Figure 9. There are some differences to the BSD results. The methods of Achanta [1] and Veksler [14] have a better regularity which is due to the smooth background of medical images. For medical images the recall rate of MonoSLIC is higher when compared to the other methods on the BSD.

3D - Visceral Dataset For the 3D case again the **recall** rate is shown for the Visceral dataset. This time the supervoxel size is given in mm^3 as the medical recordings have a defined real-world size for one voxel. The results are shown for a range of about 10^5 to $10^3 mm^3$ in Figure 10. Our method outperforms Achanta [1] for the broader range of superpixel size, for a over-segmentation in the range of $1 - 10cm^3$.

The 3D results are again **summarized** in Figure 11 for a supervoxel size of about $7 * 10^3 mm^3$. The recall rate is 10% higher for MonoSLIC. With a runtime of $0.7s/MP$ the performance of MonoSLIC is 3 times faster than Achanta [1] and it requires no parameter to be set apart of the desired number of superpixels.

A visual comparison in terms of recall rate is presented for the Visceral dataset. In the following two figures the segmentation of the algorithm is shown in red, the annotated ground truth in green and blue corresponds to a matching segmentation of the algorithm to the ground truth. Each figure is divided by a white line, where the left shows the re-

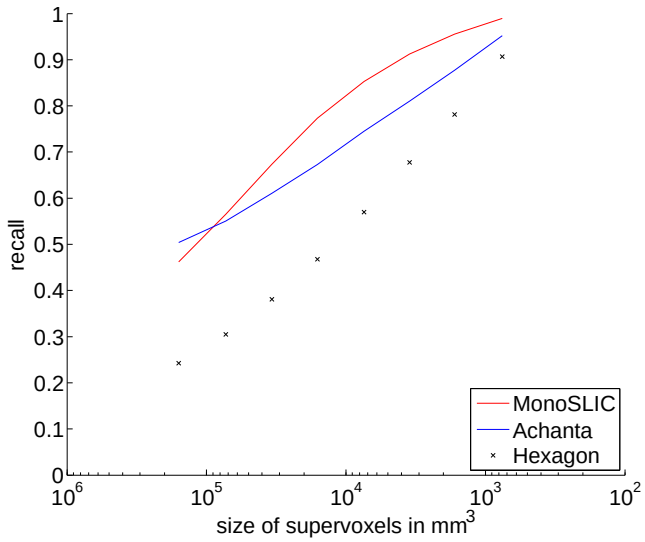


Figure 10: Recall for the 3D Visceral dataset.

SP (mm^3)		runtime		
7*10^3	recall	s/MP	P	SP
MonoSLIC	0.77	0.7	0	1
Achanta	0.67	2.1	1	1

Figure 11: Summary of the results for an over-segmentation into $15 * 10^3 mm^3$ sized supervoxels, where green indicates desired results.

sults of Achanta [1] and the right of MonoSLIC. In Figure 12 a 3D Visceral CT example is presented. If the object boundary has a high contrast it is very well segmented by [1] as seen for the lung and the silhouette of the body, while in the Abdomen region the contrast is very low and the structure is not captured. Tweaking the parameter could increase the structure detected at the cost of regularity. MonoSLIC on the other hand reacts to all the structures which is the reason for the better recall performance and it does so without the need of an additional parameter.

This can also be observed for a 2D Visceral MR example shown in Figure 13 where the uniqueness of the recording technique creates very low contrast volumes. The results for the MR are shown for the 2D case because [1] did not compute on the 3D volume.

Run-Times The run times of the algorithms are displayed in Figure 14. Computations were performed on a 12-core Intel Xeon with the publicly available implementations. Mori [13] not only is the slowest algorithm, but due to the large weight matrix generated for the *NCut* it also scales badly in terms of memory usage. Veksler [14] segments images in about 40 seconds. While Engel [5] has a faster computation time of 4.7 s/MP, Felzenswalb [7] manages to process 1 MP in under 1 second. For images of size 0.15 MP Achanta [1] and monoSLIC take 1.6 s/MP. While [1] has a similar performance for larger images and volumes MonoSLIC segments these in 0.7 s/MP, which is the fastest of all algorithms.

With 12 cores the CPU allows parallel processing, but only a few parts of the methods make use of all the cores.

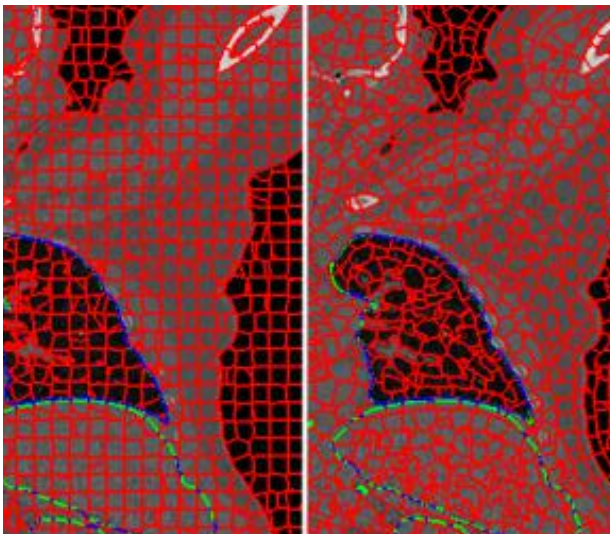


Figure 12: Comparison of Achanta [1] on the left and MonoSLIC on the right on an 3D Visceral Computed Tomography slice example with a supervoxel size of $7 * 10^3 mm^3$.

The features extracted by [13] as well as the NCut segmentation technique use multiple cores. The only other method that has an advantage of the multi-core CPU is the monogenic signal calculation of monoSLIC, as the FFT used for convolution is multi-threaded.

There is more potential for speed improvements. The method of [14] is designed for running in parallel but is only implemented in *C++* for single core. The code of [13] and [5] also have improvement potential as the code is partially implemented in *Matlab* and *C* and only some parts are parallel. The *FFT* used by the monogenic signal could also adapted to make use of the GPU, which for 3D volumes is currently limited by the memory.

5 Conclusion

We have presented *monoSLIC*, a novel method for the computation of superpixels / -voxels, which incorporates the monogenic signal's unique characteristics of being invariant to contrast and brightness as well as robust to noise. It represent the dominant edge information in any given image patch corresponding to a selected scale. The resulting superpixels / -voxels provide a nice balance of of compactness, homogeneity and fidelity to edge information. Combined with a fast runtime in 3D the algorithm outperforms state of the art methods in terms of runtime, recall, regularity and robustness to noise and is therefor particularly well suited for 3D radiological image data.

Acknowledgement

This work was partly supported by the European Union FP7 (KHRESMOI FP7-257528, VISCERAL FP7-318068), by the Austrian National Bank Anniversary Fund (BIOBONE 13468, AORTAMOTION 13497, FETALMORPHO 14812), and the Austrian Science Fund FWF (PULMARCH P 22578-B19).

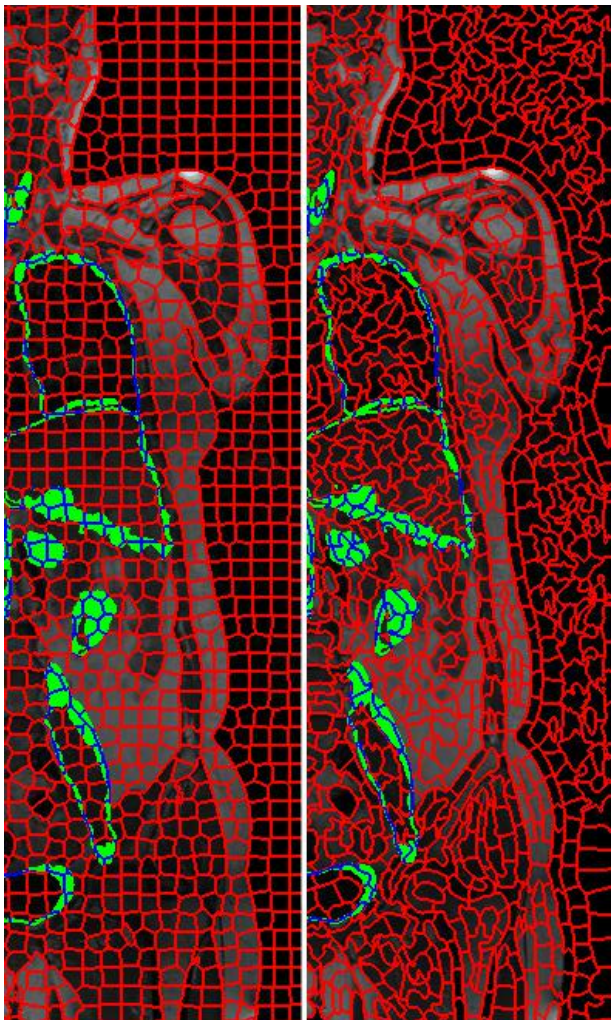


Figure 13: Comparison of the Achanta [1] on the left and MonoSLIC on the right on an 2D Visceral Magnetic Resonance slice example with about 200 pixels per supervoxel.

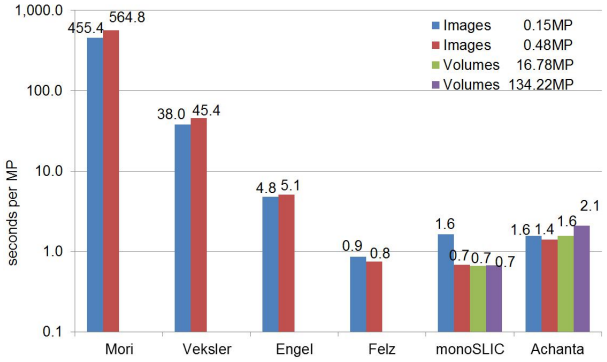


Figure 14: Runtime performance per MP in logarithmic scale on different image and volume sizes.

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. SLIC Superpixels. Technical Report 11, EPFL, Nov. 2010.

[2] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. SLIC Superpixels Compared to State-of-the-art Superpixel Methods. *IEEE PAMI*, 34(11):2274–82, Nov. 2012.

[3] T. Bulow, D. Pallek, and G. Sommer. Riesz transforms for the isotropic estimation of the local phase of moire interferograms. In *22. Symposium fuer Mustererkennung*, pages 333–340. Springer-Verlag, 2000.

[4] L. Cohen. Time-frequency distributions-a review. *Proceedings of the IEEE*, 77(7):941–981, 1989.

[5] David Engel and Cristobal Curio. Scale-Invariant Medial Features based on Gradient Vector Flow Fields. In *19th ICPR*, pages 1–4. IEEE, Dec. 2008.

[6] M. Felsberg and G. Sommer. The Monogenic Signal. *IEEE Transactions on Signal Processing*, 49(12):3136–3144, 2001.

[7] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision*, 59(2):167–181, Sept. 2004.

[8] G. Langs, H. Müller, B. H. Menze, and A. Hanbury. VISCERAL: Towards Large Data in Medical Imaging - Challenges and Directions. In *Medical Content-Based Retrieval for Clinical Decision Support*, pages 92–98, Nice, France, 2013.

[9] A. Levinshtein, A. Stere, K. Kutulakos, D. Fleet, S. Dickinson, and K. Siddiqi. TurboPixels: fast superpixels using geometric flows. *PAMI*, 31(12):2290–2297, 2009.

[10] J. Malik. Normalized Cuts and Image Segmentation. *IEEE PAMI*, 22(8):888–905, 2000.

[11] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics. In *Proceedings of the 8th ICCV*, volume 2, pages 416–423, 2001.

[12] A. P. Moore, S. J. D. Prince, J. Warrell, U. Mohammed, and G. Jones. Superpixel Lattices. In *CVPR*, pages 1–8. IEEE, June 2008.

[13] G Mori. Guiding model search using segmentation. In *10th ICCV*, volume 2, pages 1417–1423. IEEE, 2005.

[14] O. Veksler, Y. Boykov, and P. Mehrani. Superpixels and Supervoxels in an Energy Optimization Framework. In *Proceedings of the 11th European Conference on Computer Vision: Part V*, pages 211–224, 2010.

[15] L. Vincent and P. Soille. Watersheds in Digital Spaces: An Efficient Algorithm based on Immersion Simulations. *IEEE Transactions on PAMI*, 13(6):583–598, June 1991.

[16] M. Wertheimer. Untersuchungen zur Lehre von der Gestalt II. *Psychologische Forschung*, 4(1):301–350, 1923.

[17] L. Yi-Wen. Hilbert transform and applications. In *Fourier Transform Applications*, pages 291–300. InTech, 2012.

Computer Vision Winter Workshop 2014, Zuzana Kúkelová and Jan Heller (eds.)
Křtiny, Czech Republic, February 3–5
Czech Society for Cybernetics and Informatics

Reeb graph based examination of root development

Ines Janusch[1], Walter G. Kropatsch[1], and Wolfgang Busch[2]

[1] Vienna University of Technology
Institute of Computer Graphics and Algorithms
Pattern Recognition and Image Processing Group
Vienna, Austria

[2] Gregor Mendel Institute of Molecular Plant Biology
Austrian Academy of Sciences
Vienna, Austria

ines@prip.tuwien.ac.at, krw@prip.tuwien.ac.at, wolfgang.busch@gmi.oeaw.ac.at

Abstract *This paper presents an approach to analyze plant root development by means of topological image analysis. For phenotyping of plants their root development, the architecture of their root systems and thereby root characteristics such as branches and branch endings are analyzed. In order to simplify the examination of root characteristics and enable an efficient comparison of roots, a representation of imaged root data by Reeb graphs is introduced. Reeb graphs capture the topology of the represented structure - in this case the locations of branches and branch endings of the roots - and form a skeletal representation of the underlying image data in this way. As the roots are pictured as 2D image data, the projection of a 3D structure to a 2D space might result in an overlap of branches in the image. One major advantage when analyzing roots based on Reeb graphs is posed by the ability to immediately distinguish between branching points and overlaps in the root structure. This is not as easily possible by an analysis solely based on contours.*

1 Introduction

Reeb graphs are widely used as shape descriptors for 3D structures. [2] gives a general overview on the use of Reeb graphs for shape analysis. [10] uses Reeb graphs for a pose independent segmentation of 3D data of human body scans, while [8] provides a skeletal representation of point clouds based on Reeb graphs. As a representation of 2D data, Reeb graphs are for example used in [5] to provide a data skeletonization of the image content. However, Reeb graphs have not been applied to branched structures like roots or blood vessels although they pose a well suited representation. An analysis of branching patterns of roots based on a 3D reconstruction of the root architecture of rice plants is provided in [11].

One of the ultimate challenges of biology is posed by the question how genotypes translate into phenotypes. There, the major bottleneck lies in the ability to phenotype a large number of individuals and genotypes with high accuracy. This is particularly lagging in complex multicellular organisms such as plants, in which specific biological processes often occur only temporarily and are restricted to specific organs, tissues or even individual cells. Efficient and unsupervised image segmentation and the extraction of certain characteristics are a key in approaching this goal. The root of the small plant *Arabidopsis thaliana* is excellently suitable for large-scale non-invasive phenotyping because it can be grown on transparent media in large numbers and its projections of the young root essentially capture all the important biological features at the organ level.

When analyzing roots (for e.g. phenotyping), characteristics such as the number of branches or the position and number of branch-endings, are studied. These characteristics can be efficiently described by (Reeb) graphs. Reeb graphs describe changes in topology in the represented structure. Reeb graphs are based on Morse theory and analyze the (here) image content according to a function (Morse function).

When growing, roots change their shape, branches are formed - their topology changes. Moreover the projection of the 3D root structure to the 2D image data might cause overlaps of branches in the image. In a Reeb graph a distinction between a branch and an overlap is immediately possible as these changes in topology are captured by the graph.

The Reeb graph is used as a simplified, skeletal representation of the image data that captures the intrinsic topological structure of the data and allows for a comparison of the image content. Especially for the root dataset these comparisons allow for a description of the growth process: the roots are imaged on consecutive days through their growth period. In comparison with a simple standard skeletonization approach as, for example, the Medial axis transform, the skeleton derived by a Reeb graph not only describes characteristics of the image content (here

branches of the roots) but captures the actual positions of these characteristics as well.

The paper is structured as follows: Section 2 gives an introduction to Reeb graphs, Section 3 describes the dataset used and Section 4 shows the computation of a Reeb graph on the root dataset. The need for modifications of the Reeb graphs and the types of modifications are discussed in Section 5, Section 6 shows evaluation results on the root dataset while a conclusion and a perspective to future work are given in Section 7.

2 Reeb Graphs and Morse Theory

Based on critical points according to a scalar function a Reeb graph describes the topological structure that is the connectivity of level sets of e.g. 2D or 3D content [4]. In order to build a Reeb graph, critical points, of the structure to be represented, need to be computed.
A point (a, b) of a function $f(x, y)$ is called a critical point if both derivatives $f_x(a, b)$ and $f_y(a, b)$ are equal 0 or if one of these partial derivatives does not exist [9].
Such a critical point either be degenerate or non-degenerate. These two cases can be distinguished via the Hessian matrix. The determinant of the Hessian matrix at a critical point x is then called the discriminant. If this determinant is zero then x is called a degenerate critical point of f (or non-Morse critical point of f). Otherwise it is non-degenerate (or Morse critical point of f).

A smooth, real-valued function $f : M_d \to \mathbb{R}$ is called a Morse function if it satisfies the following conditions for a d manifold M_d with or without boundary:

- all critical points of f are non-degenerate and lie inside M_d,
- all critical points of f restricted to the boundary of M_d are non-degenerate,
- for all pairs of distinct critical points p and q, $f(p) \neq f(q)$ must hold [3].

Critical points of such a real-valued function are those points where the gradient becomes zero. The topological information of a shape described by a Reeb graph based on a function is related to the level sets of this function on the shape [2]. A change in topology appears with a change in the number of connected components in a level set. At regular points no topology changes occur. Topological changes occur at critical points only.
Reeb graphs are compact shape descriptors that preserve the topological characteristics of the described shape [2]. Vertices of the Reeb graph correspond to critical points of the function (points where the topology of M changes), edges describe topological persistence [2]. In other words: All nodes having the same function value are represented by one node in the graph, connections between nodes describe connections between segments of the underlying structure.
Reeb graphs are originally defined for the continuous space, but have been extended to the discrete domain: Here the

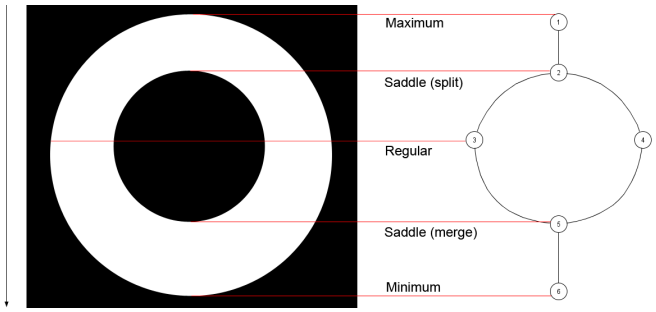


Figure 1: Critical points computed based on the height function and corresponding Reeb graph. The white image region shows the foreground region described by the Reeb graph, black parts are background.

Reeb graph is defined on a piecewise linear Morse function [4]. As the approach presented in this paper provides an analysis of 2D image content, it is based in the discrete domain (image pixels). The Reeb graphs that are built on the root images are therefore discrete Reeb graphs and are based on the following definitions. In order to define a discrete Reeb graph, connective point sets and level-set curves are defined first:

- Two point sets are connected if there exists a pair of points (one point of each point sets) with a distance between these two points below a fixed threshold.
- If all non-empty subsets of a point set, as well as its complements, are connected, such a point set is called connective.
- A group of points that have the same Morse function value and that form a connective point set, is called a level-set curve [10].

The nodes in a discrete Reeb graph represent level-set curves, the edges connect two adjacent level-set curves, therefore the underlying point sets are connected [10].

In 2D critical points and corresponding nodes in the Reeb graph are minima, maxima or saddles [3]. The saddle nodes can be further distinguished: a saddle node that appears with a reduction in the number of connected components is further called merge (saddle) node, a split (saddle) node describes an increase in the number of connected components. When considering these two different types of saddle nodes that might appear in a Reeb graph, four different types of critical points and according nodes in the graph can be distinguished: maximum node, minimum node, split (saddle) node, merge (saddle) node. Besides these nodes corresponding to critical points, regular nodes can be added at any position and along any edge in the Reeb graph as they do not describe a change in topology. Nevertheless regular nodes can, for example, be used to describe changes in the color of the foreground region (see [1]).

The approach described in the following sections uses the height function as Morse function. In 2D the height

function is the function f that associates for each point $P = (x, y)$ the value y as the height of this point: $f(x, y) \mapsto y$.
Figure 1 shows an example for a Reeb graph based on a height function, containing all five types of nodes and the actual image the graph was computed on. Each edge in the Reeb graph describes a connected component. Therefore the edges of a Reeb graph are formed by connecting the node representing the birth of a connected component to the corresponding node representing the death of this component.

3 Root dataset

For the root dataset images of the plant *Arabidopsis thaliana* were taken. This plant is a model organism, which is widely used in plant sciences, due to the small size of its genome, the small size of the plant itself and its rapid life-cycle [6]. The plants are grown on a nutrient containing agar gel surface in plastic petri dishes that are vertically oriented. All plants in one plate belong to one dataset. One dataset/plate consists of 2 rows of 12 plants. The plates are placed in a growth chamber that allows for controlled conditions as constant temperature or humidity.
The images are taken using an image scanner. A special fixture allows for two datasets to be placed in an exact known position inside the scanner. The images are acquired with a scan at 1200 dpi resolution with 8bit color depth, therefore one image is of approximately 6000x6000 pixels in size. The images are stored as bmp files of about 150MB. Along time several successive images are acquired this way, as each plate is scanned at several successive days of the growth process. A 3D stack of 2D images over time is thus created for each root.
In a preprocessing step the 24 plants per plate are cropped to single images: one image per plant with an image size in the range of 500x1300 to 800x1300 pixels resolution and a file size of 1,5-2,5Mbyte. Example images of this dataset are shown in Figure 2.

The whole set of plant images used here consists of 9 sets of time series. Each set holds 6 images of one plant taken over time (day 1, day 4, day 8, day 12, day 16 and day 20 of the growth period). Of these 54 images, 34 images are analyzed, the other images are too early in the growth process and therefore to small in structure to be represented by a non-trivial Reeb graph.
All images analyzed are segmented in a preprocessing step and consist of 2 foreground regions (leaves and roots, only the roots are analyzed for this approach) and up to 2 holes in the foreground structure. For reasons of the needed preceded segmentation, the dataset is restricted in its size, as the segmentation approach was done semi-automatically and required a lot of time (up to 1.5h for one image).

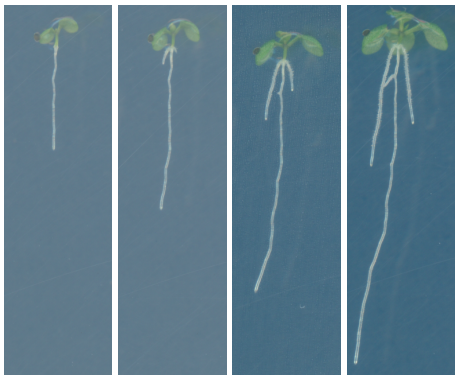


Figure 2: Example images of the root dataset: root004 - (a) day 8; (b) day 12; (c) day 16; (d) day 20.

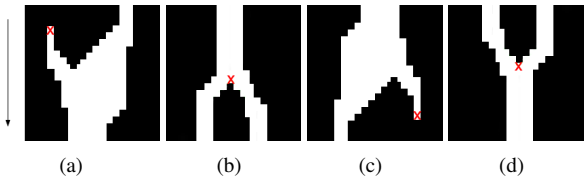


Figure 3: Four different types of critical points computed according to the height function: (a) maximum / birth; (b) saddle (split); (c) minimum / death; (d) saddle (merge).

4 Computation of Reeb graphs

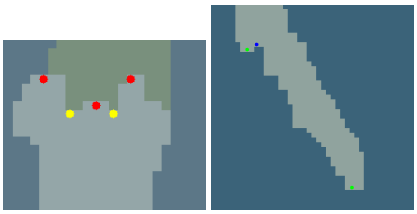
As the roots are imaged in their natural direction of growth (leaves in the top part of the image, roots growing downwards in a vertical direction) and branches occur mostly in this direction of growth, the height function is a suitable measuring function. Critical points indicate a change in topology, therefore they might only appear on the border of a region but not within the region. The borders of flat-regions in the image are analyzed to locate these critical points.
Figure 3 shows the four different types of critical points that are computed for the image content using a height function.

To compute the critical points a segmentation of the image needs to be done during a preprocessing step. As the height function is used to compute the critical points, the foreground region borders are analyzed with regard to horizontal borders as these might describe a change in the number of components. The so found critical points are located at the center of such a horizontal border.

There are two main problems encountered using this approach:

4.1 Critical points at same height

Due to the resolution of the image, the discretization of the root and further distortions during the segmentation process, it is possible that several critical points at different horizontal positions in the image are at the same vertical position (same height) in the image (see Figure 4(a) for an example). In this case the third criteria of Morse theory (see Section 2) is not met. A Reeb graph cannot be built, as a decision on how to connect the nodes in order to build the graph can-



(a) (b)

Figure 4: Problems encountered on the root dataset: (a) several critical points on same height; (b) frayed borders due to segmentation artefacts.

not be taken. Figure 5 shows an example: The solid lines illustrate the only two fixed connections in this example, the dashed lines indicate all possible connections. In this Reeb graph four edges are needed: one from each black (maximum) node to a red (saddle) node and one edge from a red (saddle) node to the green (minimum) node. A decision concerning these connections needs to be taken for the black center node as well as for the two red nodes. A solution to build a Reeb graph, despite several critical points at the same height, is discussed in Section 5.1

4.2 Additional critical points

Because of the segmentation prior to the computation of the critical points, segmentation artefacts appear in the images. The most common problem are frayed borders of image regions (see Figure 4(b) for an example). Especially for images of day 16 the segmentation creates noise and distorted region borders. When analyzing the images of day 16 one notices a high humidity between the plates in the form of water drops, which creates a highly texturized background that complicates the segmentation.
These frayed borders in the segmented images result in additional critical points that describe no actual split or merge of the root structure. These artefacts alter the Reeb graph and complicate a comparison or matching of graphs. One possibility on how to deal with these additional critical points is described in Section 5.2.

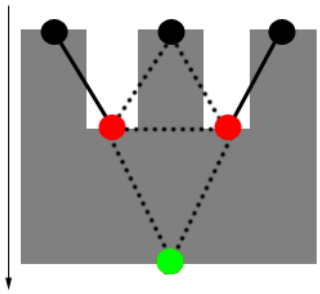


Figure 5: Critical points at same height: the solid lines show connections that are fixed, dashed lines indicate all possible connections - a decision needs to be taken for these.

5 Modifications on the graphs

To overcome the problems discussed in Section 4.1 and Section 4.2 the following techniques were used:

5.1 Controlled shift of critical point coordinates

Due to the discrete pixel-space, the coordinates x and y of a pixel $p = (x, y)$ are integers. Critical points at the same height (same y-coordinate) occur for 35% of all images in the root dataset and are shifted. The height of such critical points is changed by an added factor f, $0 \leq f < 1$. A critical point $p = (x, y)$ is shifted to $p' = (x, y + f)$, f is computed using the following formula: $f = \frac{1}{w} \cdot (x-1)$, with w giving the width of the image. The y-coordinate is thereby changed from an integer to a floating-point number. Critical points at the same height are moved downwards in a left-to-right order, thus for two critical points $p_1 = (x_1, y)$ and $p_2 = (x_2, y)$ with $x_1 < x_2$, it is valid that, after shifting the points to $p'_1 = (x_1, y_1)$ and $p'_2 = (x_2, y_2)$, $y_1 < y_2$ holds. The actual order of heights is preserved by this correction procedure as only critical points that were primarily at the same height are changed. All critical points are at different heights, although when rounding down the y-coordinate of the critical points to an integer, they stay in the actual pixel line. A Reeb graph can therefore be built.
It is important to shift the heights in a fixed approach. A random decision choosing one of two critical points at the same height when building the Reeb graph cannot be used, as the results may vary with repeated tests. Reeb graphs built on such random decisions are not unique and therefore useless for e.g. comparison of two images.
Figure 6 shows a Reeb graph built on the marked critical points / nodes. Compared to Figure 5 where there are several critical points at the same height, Figure 6 shows critical points on different heights. The connections in this graph are unique. By shifting the critical points in Figure 5 according to the approach described in this section, the critical points are shifted to a configuration similar to the one shown in Figure 6.

5.2 Graph pruning

Due to the segmentation done as a preprocessing step, segmentation artefacts falsify the number of critical points and therefore the number of nodes and edges in the Reeb graph.

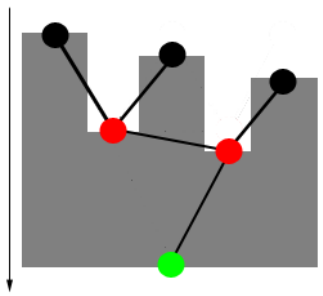
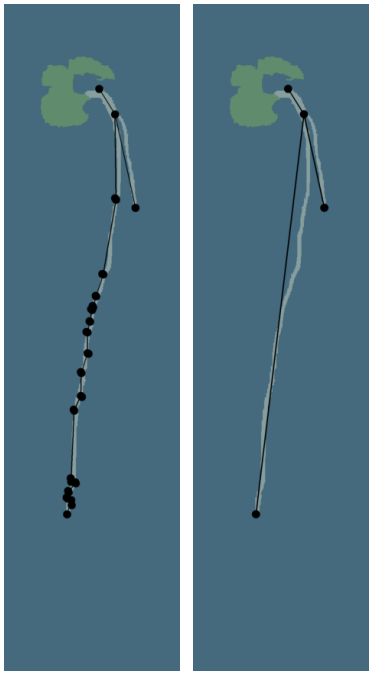


Figure 6: Critical points at different heights, the connections in this Reeb graph are unique.

number of nodes in graph					
type of node	birth	split	merge	death	sum
no graph pruning	111	129	84	156	480
graph pruning	38	54	13	79	184

Table 1: Total number of each type of nodes in the Reeb graphs of the root dataset with and without graph pruning.

In order to use the extracted graphs as a skeletal representation, branches that arise with artefacts need to be removed from the Reeb graph.
For each pair of adjacent nodes in the graph the Euclidean distance between these two nodes is computed. If this distance is less than 1,5% of the image height such connections are discarded and nodes are relinked if needed. This threshold proved to be the best choice in the experiments.
Regular nodes may be introduced by this approach. As these regular nodes do not contain any needed information, they are removed after relinking. This graph pruning results in a reduction of the overall number of nodes in the Reeb graphs of the root dataset by 62%. Table 1 shows the numbers of nodes for all Reeb graphs in the root dataset with and without graph pruning and Figure 7 shows an example of the Reeb graph and the modified Reeb graph for root 05, day 16. All the nodes in the lower part of the root for the Reeb graph without graph pruning indicate spurious branches detected due to noise in the segmented image. These spurious branches are correctly discarded by the graph pruning approach.



(a) (b)

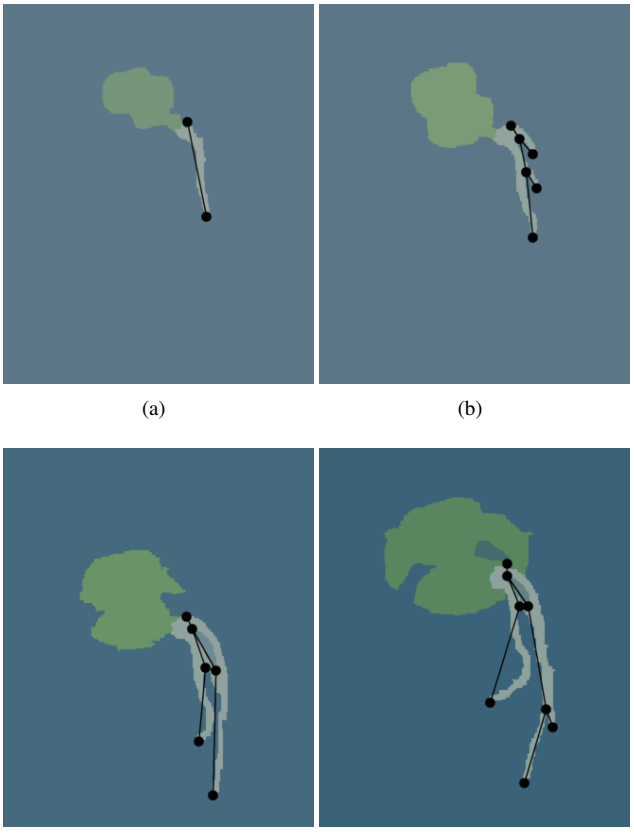
Figure 7: Reeb graph for root 05 day 16. (a) without graph pruning; (b) with graph pruning.

6 Results and evaluation on the root dataset

Figure 8 shows the resulting Reeb graph for root 07 of the dataset with both modifications implemented, drawn as an overlay. There is a cycle in the Reeb graph for day 16 and day 20 (Figure 8(c) and 8(d)). In the image of day 12 (Figure 8(b)) there are three branches: The first and the second branch overlap at some time during the growth process between day 12 and day 16. Because of this overlap in the 3D space, these two branches appear merged in the 2D projection of the image, therefore a cycle is formed in the Reeb graph.

Figure 9 shows the Reeb graphs for root 12 of the dataset (both modifications are used). Some small branches are not represented in the Reeb graphs of day 12, 16 and 20 as they resembled branches due to noise and were discarded during the graph pruning process (see Section 6.1). Again a cycle appears in the Reeb graph for day 20 as two branches overlap.

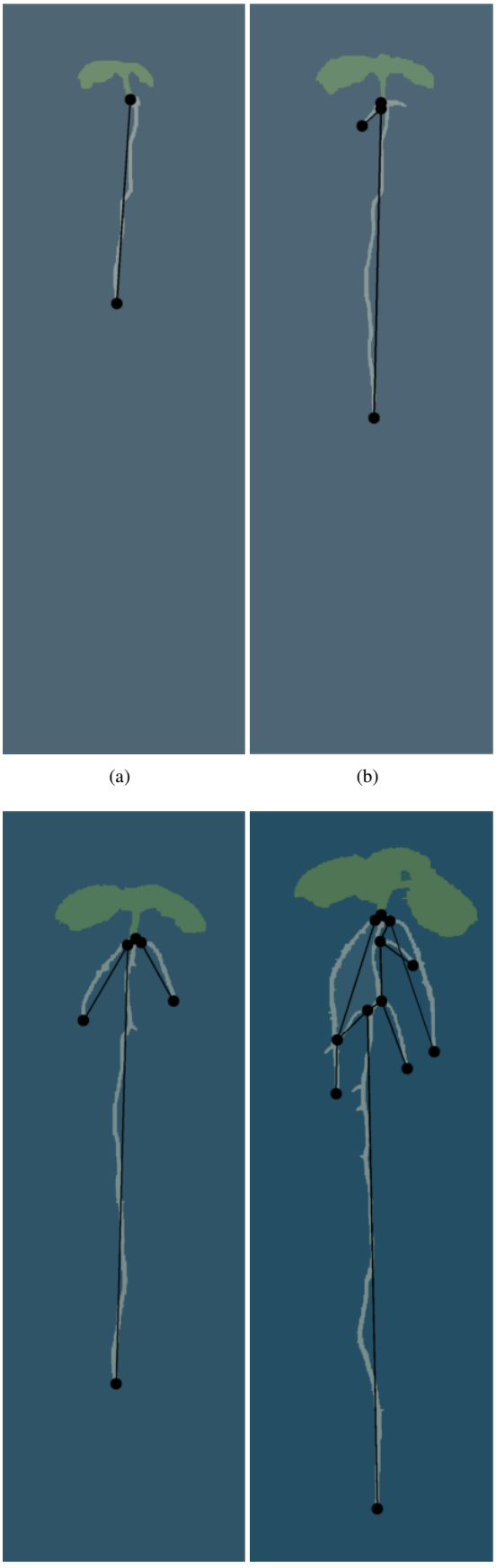
For the 34 single images of the root dataset the following criteria have been evaluated:



(a) (b)

(c) (d)

Figure 8: Resulting Reeb graph for root 07 (a) day 8; (b) day 12; (c) day 16; (d) day 20.



(a) (b) (c) (d)

Figure 9: Resulting Reeb graph for root 12 (a) day 8; (b) day 12; (c) day 16; (d) day 20.

wrong decisions on graph pruning			
	images	false negatives	false positives
graph pruning	8	10	0
extension 1	18	4	36
extension 2	13	10	9

Table 2: Branches wrongly discarded (false negative) and wrongly accepted (false positive) in the graph pruning approach with two different corrections based on pixel-color.

6.1 Are all branches correctly detected and represented by the Reeb graph?

All major branches were correctly detected. Table 2 shows the number of images for which branches were wrongly discarded (false negatives) or wrongly accepted (false positive). For 23,5% of the images smaller branches were discarded due to the graph pruning as they resembled the frayed border artefacts caused by the segmentation. To keep these small branches that describe actual root structures, their angle could be taken into account, as true branches seem to inscribe a larger angle than branches due to noise. However, this assumption is based on the dataset presented and may not be true for other datasets. Therefore another approach was tested: for a small branch with a critical point of type split, the color values at three pixels: at the critical point (a), one row below the critical point (b) and two rows below the critical point (c) were compared:

1. the color of (a) and (c) were taken from the segmented image, while the color value of (b) was taken from the unsegmented image

2. all three color values were taken from the unsegmented image

Branches are kept if the color value of (b) is closer to (a) than to (c). Table 2 shows the results for these two tests. While the first option discards less true branches (false negatives) it keeps spurious branches for more than 50% of all images. The second option keeps less spurious branches, but does not reduce the number of false negatives compared to the graph pruning approach without these color comparisons. Taking into account not only the color values of these three pixels but of several neighbors, as it is done with Local Binary Patterns, might present an option for future work.

6.2 Are additional branches (due to e.g. noise) detected?

As shown in Table 2 all additional branches (due to segmentation artefacts) are correctly discarded by the implemented graph pruning approach.

For a series of images of one plant during the growth process the following factors have been analyzed:

number of nodes / edges / cycles				
	day 8	day 12	day 16	day 20
root 04	2 / 1 / 0	4 / 3 / 0	6 / 5 / 0	8 / 8 / 1
root 05	2 / 1 / 0	4 / 3 / 0	4 / 3 / 0	8 / 8 / 1
root 07	2 / 1 / 0	6 / 5 / 0	6 / 6 / 1	8 / 8 / 1
root 09	2 / 1 / 0	6 / 5 / 0	6 / 5 / 0	6 / 5 / 0
root 12	2 / 1 / 0	4 / 3 / 0	6 / 5 / 0	12 / 12 / 1
root 17	-	2 / 1 / 0	6 / 5 / 0	6 / 5 / 0
root 19	2 / 1 / 0	4 / 3 / 0	8 / 8 / 1	14 / 15 / 2
root 20	2 / 1 / 0	4 / 3 / 0	4 / 3 / 0	12 / 12 / 1
root 24	-	2 / 1 / 0	4 / 3 / 0	10 / 9 / 0

Table 3: Total number of nodes, edges and cycles in the modified graph (graph pruning without corrections) of each root image in the defined dataset.

6.3 Is an automatic grouping of images of one plant from different days possible?

As the roots grow downwards in a vertical direction, there are only minor changes in the position of the starting point of the actual root (transition between leaves an roots) - not accounting for actual movement of the plant (e.g. sliding down the plate). Therefore the starting point was used for this comparison. The average minimal Euclidean distance between all starting points is 14,4 pixels. Using this distance measurement to group one image of a root with earlier or later images of the same root, the grouping is correct for 71% of all images. However, images of day 16 falsify these numbers, as the plate of day 16 appears slightly enlarged in the image compared to the images of other days. As the images were automatically cut into single plant images in a preprocessing step, this scaling is not corrected. Excluding the images of day 16, the average minimal Euclidean distance decreases to 11,6 pixels and one image is grouped correctly with earlier or later images of the same root in 88%.

6.4 General assumption: „Parts of a plant that appear in an early image of a plant do not disappear for a later image of the same plant.“

This assumption proved to be correct for the images in the root dataset. The topology of a root only changes with the creation of new components (e.g. branches) over time. Table 3 shows the number of all nodes, edges and cycles in each (modified) graph of the root images.
However, there is one exception to this assumption, which is based on the projection of a 3D structure to a 2D space. A branch in an early image of a plant might stay in the image of a later day, it may branch again but its ending may also disappear in the 2D image as it is merged with another branch due to an overlap of these two branches in the 3D space.

7 Conclusion and future work

Reeb graphs proved to be suitable descriptors for root structures as they capture the main characteristics of roots, namely branches and branch endings that are used in the phenotyping of plants, well. A Reeb graph provides a skeletal representation of a root that allows for fast analysis of root characteristics and efficient comparison of images and the contained root structure. Overlaps in 3D that appear as a merge of two branches in a 2D image are hard to distinguish from a branching point when analyzing only contours of image regions. Exploiting the topology of the root, actual branching points and overlaps in 3D can be immediately distinguished, as an overlap forms a cycle in the corresponding Reeb graph.
A future application in plant phenotyping is possible. However, for future work the segmentation approach needs to be changed to a less time-consuming (or even automatic) approach in order to allow for a larger dataset to be analyzed.
Moreover, in future work, different functions will be used as Morse functions. Functions that should be taken into consideration are for example a medial axis as in [7] or distance functions: for example the distance to a fixed point in a structure, the sum of geodesic distance (both are used in [10]) or the distance to an existing graph (as for example a medial axis).
Open questions for future work (on the root dataset) are: How does the chosen Morse function influence the correct detection of branches in the root structure? Is the detection of all branches, respectively the detection of additional branches due to noise, dependent on the Morse function used? Which Morse functions are able to correctly represent roots with a complex pattern of growth (e.g.: change in the main direction of growth)?

Acknowledgement

We thank the anonymous reviewers for their constructive comments.

References

[1] Stefano Berretti, Alberto Del Bimbo, and Pietro Pala. 3D mesh decomposition using Reeb graphs. *Image and Vision Computing*, 27(10):1540–1554, Sept. 2009.

[2] S. Biasotti, D. Giorgi, M. Spagnuolo, and B. Falcidieno. Reeb graphs for shape analysis and applications. *Theoretical Computer Science*, 392(13):5–22, Feb. 2008.

[3] Harish Doraiswamy and Vijay Natarajan. Efficient algorithms for computing Reeb graphs. *Computational Geometry*, 42(67):606–616, Aug. 2009.

[4] Rachid EL Khoury, Jean-Philippe Vandeborre, and Mohamed Daoudi. 3D mesh Reeb graph computation using commute-time and diffusion distances. volume 8290, pages 82900H–82900H–10, 2012.

[5] Xiaoyin Ge, Issam I. Safa, Mikhail Belkin, and Yusu Wang. Data skeletonization via Reeb graphs. In J. Shawe-Taylor, R. S. Zemel, P. Bartlett, F. C. N. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 837–845. 2011.

[6] Makoto Hayashi and Mikio Nishimura. Arabidopsis thaliana - a model organism to study plant

peroxisomes. *Biochimica et Biophysica Acta (BBA) - Molecular Cell Research*, 1763(12):1382–1391, Dec. 2006.

[7] Francis Lazarus and Anne Verroust. Level set diagrams of polyhedral objects. In *Proceedings of the fifth ACM symposium on Solid modeling and applications*, SMA '99, page 130140, New York, NY, USA, 1999. ACM.

[8] Mattia Natali, Silvia Biasotti, Giuseppe Patan, and Bianca Falcidieno. Graph-based representations of point clouds. *Graph. Models*, 73(5):151164, Sept. 2011.

[9] James Stewart. *Calculus*. Cengage Learning Emea, 6th edition. international met edition, Feb. 2008.

[10] N. Werghi, Yijun Xiao, and J.P. Siebert. A functional-based segmentation of human body scans in arbitrary postures. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 36(1):153–165, 2006.

[11] Ying Zheng, S. Gu, H. Edelsbrunner, C. Tomasi, and P. Benfey. Detailed reconstruction of 3D plant root shape. In *2011 IEEE International Conference on Computer Vision (ICCV)*, pages 2026–2033, 2011.

Computer Vision Winter Workshop 2014, Zuzana Kúkelová and Jan Heller (eds.)
Křtiny, Czech Republic, February 3–5
Czech Society for Cybernetics and Informatics

Pedestrian Recognition and Localisation in Image Sequences as Bayesian Inference

Tobias Klinger, Franz Rottensteiner, and Christian Heipke

Institute of Photogrammetry and GeoInformation,
Leibniz Universität Hannover, Germany
{klinger, rottensteiner, heipke}@ipi.uni-hannover.de

Abstract *An exhaustive-search people detector such as the HOG/SVM has at least two drawbacks w.r.t. the accuracy of recognition and geometric precision: first, it achieves high recall rates only among several false positive detections and second, the geometric precision of the positioning of the underlying object is poor due to the non-rigid body shape of people and background structures. However, the fact that the HOG/SVM does potentially provide high recall rates makes it a fair basis for hypothesise-and-validate-frameworks. We build upon the outcome of the HOG-detector and improve the recognition performance and geometric precision of the same using Bayesian Networks and apply statistical knowledge that we learn from training data for the definition of the probability functions. The approach is evaluated on two real image sequences and achieves results that can compare with the state of the art.*

1 Introduction

Automatic object detection is a key discipline in photogrammetry and computer vision. The term detection involves the recognition, i.e. the decision that an object of a specific object class is present and at least a coarse localisation of the object. Most state-of-the-art pedestrian detectors like the HOG/SVM [23] or the AdaBoost based detector [7] scan the entire image at different scales with a sliding window and classify its content as either person or background. Though these approaches give a solution to the recognition and localisation problem at the same time, the results are not particularly reliable and precise. In a comparative study of 16 different people detection systems [8] the authors point out that acceptable recognition rates are only achieved, if many false positive detections are accepted as well. Such systems, if applied permissively, have a high chance for false positive detections, because they usually rely on a single type of low-level features, which is typically not discriminative enough against similar object classes. It is hence reasonable to classify also against other similar object classes like in [16], or to evaluate additional information like foreground information [12], [24] or shape [13], [17] prior to further processing. Sequential processing has the drawback that false decisions taken at a single step cannot be recovered later. Other approaches involve context information, e.g. [22], which constrains detections to plausible regions in the image. [9], [11], [19] constrain the detections only to the ground plane, requiring a holistic understanding of the scene, which is a formidable task in itself on one hand, and which is very restrictive, because they disregard all objects that do not stand on the ground plane on the other.

However, there has been considerable success in the improvement of people recognition in [2], [11], [19], all of which use Bayesian Networks for the inference about the presence or absence of people and their positions. Bayesian Networks are directed graphical models in which observations and hidden parameters are treated as random variables in a generative Bayesian manner. The random variables are represented by nodes and the conditional independence properties of their joint distribution are represented by directed edges, see, e.g. [3] for details.

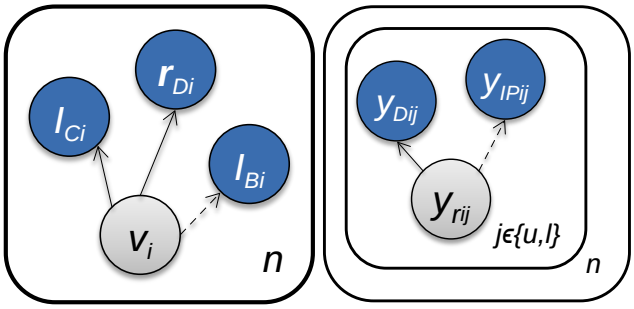
Though there is a lot of work related to the recognition of people, only few papers address the geometric accuracy of the detections. The positions of the detected persons are usually broken down to the location of the classification window, which does not always align well to the actual extents of people in the image and thus only gives an approximate position. For the evaluation of automatic detection results with reference data from manual annotations, the PASCAL VOC challenge, for instance, requires that the ratio between intersection and union area of the two rectangles is larger than 50% [10]. This criterion should highlight the object recognition performance and does not address the geometric accuracy of the detections. For many realistic applications like visual odometry with landmarks, collision avoidance in driver assistance systems or the analysis of motion and interactions of people in sport science or video surveillance, the geometric accuracy is crucial. In such applications, the 2D position of an object, usually its highest or lowest point in the image, is projected into 3D space. In [15] the importance of a correct segmentation of objects in the image for the geometric accuracy in 3D is pointed out. Comaniciou's Mean-Shift tracker [5], for instance, progressively finds the best fitting position of a tracked target by iteratively moving it to the region that best coincides with the colour histogram of the target, but only estimates the centroids of the objects, which makes the actual positioning in 3D difficult. In [19] the authors use stereo-image pairs as input and jointly estimate the object position on and the parameters of the ground plane in the scene. The locations of the pedestrians, given by a HOG-detector, are then optimised in 3D by joint prob-

abilistic modeling with the ground plane parameters. Here, it is indirectly assumed that the detector already delivers the correct bounding boxes in the image. In [6] pixel-wise segmentation of approximately positioned objects is conducted by integrating edge and colour cues. As the location estimated by the detector is only used as initialisation, the segmentation is prone to drift away from the underlying object.

In this work, we stick to the Bayesian probability theory and evaluate various sources of information about the presence or absence and the positions of people in a probabilistic model. In contrast to the related work, we do not require a holistic scene model and we restrict detections only to parts of the scene that are accessed by people during a training sequence. We aim to achieve state-of-the-art recognition results in challenging indoor and outdoor scenarios and to improve the geometric accuracy of the localisation of the detected objects at the same time. The validity of a detection hypothesis and the location in the image are treated as hidden parameters in two different Bayesian Networks. Like [6], we break with the assumption of unbiased results of the detector, but go further and learn the uncertainty of positioning people from image sequences. The prior and conditional probabilities for the Bayesian Networks are all learnt from training data. The recognition performance and the geometric accuracy are evaluated on two common benchmark datasets.

2 Method

The central building block for our work on people recognition is the HOG/SVM-detector, which is capable of achieving relatively high recall rates, but is also prone to false positive detections. Convenient detection results can only be achieved when the false positives are distinguished from the true positives. In this paper people recognition is stated as a binary classification problem in which the results of the HOG-detector as well as additional information are regarded as input for a joint probabilistic model. In order to achieve the highest possible recall rate, no thresholding is applied in the HOG/SVM framework. In the remainder of this section, we introduce two different Bayesian Networks, one for the solution of the recognition problem (Sec. 2.1) and one for the improvement of the localisation accuracy of people (Sec. 2.2). For the positioning we consider the bounding box which results from the application of the HOG-detector only as an approximate position and observe a second source of information about the object position that we derive from the analysis of optical flow points. The positions of people in the image are represented by minimal spanning rectangles around the visible parts of the persons. We model the highest and the lowest row coordinate of the persons as random variables and estimate the posterior position by applying Bayesian inference. The position estimated by the second graphical model is used as observed variable in the first graphical model. Using a refined position of a detection hypothesis has the advantage that a detection candidate will only be discarded, if even at the refined position the classifier does not strike. This gives rise to the possibility that misplaced detection windows - which is often a problem, e.g. when people are observed from the side - are corrected before they are further evaluated.



(a) (b)

Figure 1: Graphical models used (a) for people recognition and (b) for people localisation. Observable variables are shown in blue, the hidden parameters in gray. The dashed edges denote the possibility of the associated observations to be omitted, see text.

2.1 Model for People Recognition

As mentioned above, people recognition is regarded as a binary classification problem and hence we determine a binary label v_i that indicates whether the ith of n detection candidates observed in each frame by the HOG/SVM-detector really corresponds to a person or not. For simplicity of notation, we omit the indexes i in the remainder of this paper.

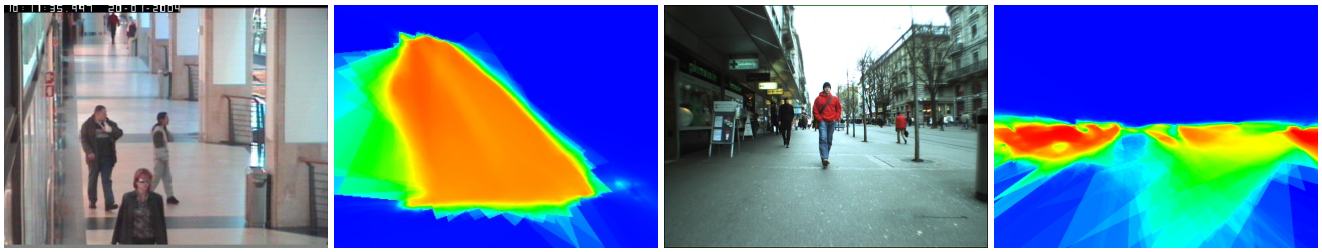
For the determination of v we apply Bayesian inference in the context of a directed graphical model, depicted in Fig. 1a. The observed variables, depicted as blue nodes in the model, are

- The surrounding rectangle $\mathbf{r}_D = [x_{Dl}, y_{Du}, x_{Dr}, y_{Dl}]^T$ around a person given by the HOG-detector, defined by its upper left (x_{Dl}, y_{Du}) and its lower right point (x_{Dr}, y_{Dl}) with their row (y) and column (x) coordinates
- The confidence value I_C that is proportional to the certainty about the binary classification (person vs. not person) of the SVM classifier used in the HOG framework
- An observation obtained from background subtraction, i.e. the fraction of foreground pixels I_B inside $\mathbf{r}_D$.

Following the standard notation for graphical models (see, e.g. [3]), each directed edge represents a conditional probability function for the child node given the parent node. The joint probability density of the involved variables can be written in accordance with the network design in Fig. 1a as Eq. (1):

$$\begin{aligned} P(v|I_B, I_C, \mathbf{r}_D) &\propto P(v, I_B, I_C, \mathbf{r}_D) \\ &= P(v)P(I_B|v)P(I_C|v)P(\mathbf{r}_D|v) \end{aligned} \quad (1)$$

For each edge in Fig. 1a we train an individual Random Forest (RF) classifier [4] using the according observation I_C, $\mathbf{r}_D$ or I_B as features. For training, we apply the HOG-detector on image sequences with available annotations of the bounding rectangles around the visible people in the scene. The detections are then divided into sets of positive



(a) ThreePastShop1 image sample (b) $P(v_i = 1|\mathbf{r}_{Di})$ (c) ETHZ-Bahnhof image sample (d) $P(v_i = 1|\mathbf{r}_{Di})$

Figure 2: Posterior probability of a detection to be correct given the center of the rectangle around the detected object in the image. For processing, the upper left and lower right points of the rectangle is used. Red: $P(v_i = true|\mathbf{r}_{Di})$ is high, blue: $P(v_i = true|\mathbf{r}_{Di})$ is low.

training samples (for v =true) and negative training samples (for v =false) by validation with reference data, using the bounding box intersection-over-union score [10]. The priors $P(v)$ are defined as the ratios of true positive and false positive detections in the training data.

The edge related to the conditional probability density function (pdf) $P(\mathbf{r}_D|v)$ represents the probability, that $\mathbf{r}_D$ is the surrounding rectangle if a person is present or absent. Given the posterior probability $P(v|\mathbf{r}_D)$ by classification with the RF and the priors $P(v)$, the likelihood $P(\mathbf{r}_D|v)$ as required for the factorisation of the joint pdf (Eq. (1)) can be written as

$$P(\mathbf{r}_D|v) \propto \frac{P(v|\mathbf{r}_D)}{P(v)}.$$

In Fig. 2 example images for both datasets used in this work are shown together with the posterior probabilities that are generated by a classifier (the RF used for visualisation is only trained with the 2D center point of the rectangle as feature) for each possible position of $\mathbf{r}_D$ in the image. The incorporation of the pdf $P(\mathbf{r}_D|v)$ is beneficial for two reasons; first, we achieve a very fine differentiation of likely vs. unlikely regions for detections and second, we do not require to interpret the scene geometry prior to the detection.

The pdf $P(I_C|v)$ is the probability density for a confidence value being observed given the presence or absence of a person. Related to posterior probability and the priors, the likelihood $P(I_C|v)$ can be written as

$$P(I_C|v) \propto \frac{P(v|I_C)}{P(v)}.$$

The pdf $P(I_B|v)$ related to the results of background subtractions is integrated into our model due to the assumption that people differ from the background because of their motion, hence the observations I_B is a strong indicator for the presence or absence of a person. The likelihood can be written as

$$P(I_B|v) \propto \frac{P(v|I_B)}{P(v)}.$$

We derive I_B only from images captured by a camera with constant exterior orientation (w.r.t. 6 degrees of freedom), i.e. where an algorithm for background subtraction such as [21] can be applied without adjustments. Therefore, the edge connected with I_B is drawn as a dashed line in Fig. 1a. For image sequences from moving camera platforms we do not apply background subtraction and the variable I_B and the according likelihood $P(I_B|v)$ is excluded from Eq. (1). For image sequences captured by a static camera, we apply the algorithm of [21] for background subtraction.

The unknown parameter v is determined to be the label that achieves the maximum a posteriori (MAP) probability among the two possible states (true and false) given the observations. The decision rule is formalised in Eq. (2).

$$v = \begin{cases} true, & \text{if } \frac{P(v=true,I_B,I_C,\mathbf{r}_D)}{P(v=false,I_B,I_C,\mathbf{r}_D)} > 1, \\ false, & \text{otherwise.} \end{cases} \quad (2)$$

2.2 Model for People Localisation

In the model described in Sec. 2.1 the minimal spanning rectangle around a person is considered observable. In fact, the location that is given by the HOG/SVM-detector is only an approximation to the true position. In this section, we define a second graphical model, see Fig. 1b, which considers the row coordinates of the highest (y_{riu}) and lowest points (y_{ril}) of a person related to the ith detection as hidden parameters. Again, we omit the indexes i for the sake of simplicity. As observed variables of this model we consider the upper and lower row coordinates y_{Du} and y_{Dl} of the HOG-detection window as well as an additional pair of observations of the row coordinates that we derive from the analysis of optical flow, i.e. y_{IPu} and y_{IPl}.

We define y_{IPj} as the row coordinates of the highest and lowest interest points on the visible parts of a person, that are tracked by an optical flow algorithm. We apply the algorithm of [20] for the selection of interest points and track them by the algorithm of [14]. For each hypothesis about the presence of a person given by the HOG-detection we apply the following strategy for the measurement of y_{IPj} (see Figs. 3 and 4 for an illustration):

1. We establish a search space for the person by expansion of the rectangle given by the HOG-detector (visualised as red rectangles in Fig. 3b-e) in vertical direction by a third of its size, in order to assure that the person is within the search space. We consider the upper 25% of this area as search space for the head point (upper yellow rectangles in Fig. 3b-e) and the lower 25% as search space for the foot point (lower yellow rectangles).

(a) (b) (c) (d) (e)

Figure 3: Examples from the ETHZ-sequence for observations used for the localisation of people. In (a) a schematic representation of the pdfs used for the inference of the posterior locations is given. The red curves represent the pdfs based on observations by the HOG-detector, the blue curves those for the observations based on optical flow. The black rectangle symbolises the true position. In (b)-(e), the yellow rectangles indicate the raw detection as result of the HOG-detector. The blue rectangles indicate the search spaces for the head and foot position, respectively. The blue horizontal lines inside the rectangles are the measured positions by the analysis of optical flow, the green horizontal lines the inferred posterior position. In (e) the position of the person in the background is not estimated by our approach correctly due to partial occlusion by the person in the foreground.

2. In both regions, we take all optical flow vectors ending in the upper and lower search space, respectively, of the current image (indicated by the short lines in Fig. 3b-e).

3. We generate a histogram of magnitudes of optical flow vectors, using 10 histogram bins and consider only flow vectors with a magnitude between 0 and 30 pixels. If the histogram has two or more local maxima, we suppose that the flow vectors related to the maximum with the smallest magnitudes originate from the background and discard the flow vectors from further consideration. Of the remaining flow vectors we also remove those that do not have more than a minimum number of neighbours in a predefined radius (we set the minimum number of neighbours to three and the radius to 20 pixels; the discarded flow vectors are visualised by red, the remaining flow vectors by green lines in Fig. 3b-e). We set the image row coordinates of the highest and lowest interest points that remain as observations of the head point (y_{IPu}) and food point (y_{IPl}), respectively (indicated by the blue horizontal lines in Fig. 3). If the histogram only has one local maximum, we do not evaluate the observation x_{IP} for the according person in the current image, because the interest points cannot be separated into points originating from the foreground and the background by our approach.

We model the likelihoods for the measurements y_{IPj} and y_{Dj} to be observed at a distance Δy from the true position y_{rj}, i.e. $\Delta y_{IPj} = y_{IPj} - y_{rj}$ and $\Delta y_{Dj} = y_{Dj} - y_{rj}$, respectively, by normal distributions:

$$P(y_{IPj}|y_{rj}) \propto e^{-\frac{1}{2}(\frac{\Delta y_{IPj} - \mu_{\Delta yIPj}}{\sigma_{\Delta yIPj}})^2} \tag{3}$$

and

$$P(y_{Dj}|y_{rj}) \propto e^{-\frac{1}{2}(\frac{\Delta y_{Dj} - \mu_{\Delta yDj}}{\sigma_{\Delta yDj}})^2} \tag{4}$$

with mean $\mu_{\Delta yDj}$ and $\mu_{\Delta yIPj}$, respectively, and standard deviation $\sigma_{\Delta yDj}$ and $\sigma_{\Delta yIPj}$, respectively. The parameters of the pdf in Eq. (3) are determined from the distribution of deviations of the measured positions y_{IPj} from the (true) positions given by reference data and those of Eq. (4) from the deviations of y_{Dj} from the reference data. A visualisation of two exemplary distributions together with the fitted Gaussians is given in Fig. 4.

Given the observations y_{Dj} and y_{IPj} measured for each detection in each consecutive frame in the evaluation phase and the parameters $\mu_{\Delta yDj}$, $\sigma_{\Delta yDj}$, $\mu_{\Delta yIPj}$ and $\sigma_{\Delta yIPj}$ of the pdfs (3) and (4) learnt from training data, the posterior position y_{rj} can be inferred for each detection candidate as the expected value

$$E(y_{rj}) = \mu_{rj} = \frac{\sigma_{rj}^2}{\sigma_{\Delta yDj}^2}(y_{Dj} - \mu_{\Delta yDj}) + \frac{\sigma_{rj}^2}{\sigma_{\Delta yIPj}^2}(y_{IPj} - \mu_{\Delta yIPj}) \tag{5}$$

with

$$\sigma_{rj}^2 = (\frac{1}{\sigma_{\Delta yDj}^2} + \frac{1}{\sigma_{\Delta yIPj}^2})^{-1} \tag{6}$$

The first term of Eq. (5) considers the influence of the observed position by the HOG-detector, weighted by the variance of the measurements in the training sequence. The second term refers to the position measured by the analysis of the optical flow vectors, also weighted by the variance of the measurements. The observation with the lower variance hence has the stronger influence on the posterior position

μ_{rj}. The subtrahends in the brackets incorporate the mean deviations of the measurements from the reference data as corrections. If the head or the feet point cannot be measured by the analysis of the optical flow vectors, the second terms of Eq. (5) and (6) are set to zero and only the first term, related to the HOG-detection, influences the posterior.

3 Experiments and Results

Experiments are conducted on two publicly available datasets, one from an indoor sequence with constant exterior orientation of the camera, the CAVIAR dataset [1] and the other from the ETHZ dataset [9] captured from a moving platform in an outdoor scenario. For the experiments involving our method from Sec. 2.2, the posterior row coordinates are used as the row coordinates of the observed rectangle in the graphical model from Sec. 2.1, maintaining the column coordinates of the HOG-detections.

3.1 Datasets

In the CAVIAR scenario, the sequence ThreePastShop1, consisting of 1650 images, is taken for training and the sequence ThreePastShop2 with 1521 images for testing. From the ETHZ dataset we take the Bahnhof-sequence of 1000 images, split the data in two halves and apply cross-validation. The training and test sequences hence follow on from one another. As the camera is mounted on a moving platform in the ETHZ-sequence, the observation I_B is excluded from the graphical model in this case. Though in the ETHZ-sequence the position of the camera changes over time, the tilt angle relative to the ground does not change significantly. We hence assume that the probabilities related to the position in the image are transferable from training to test sequences within an acceptable range of validity. The HOG/SVM-detector is configured without internal threshold, so that the results are as complete as possible. Only people with a minimum height of 48 pixels[1] are considered for processing. The bounding rectangles are shrinked in order to compress the systematic margin around people in the training data.

3.2 Detector Recognition Accuracy

The accuracy of the people detector is evaluated in terms of its recall capability and the number of false positive detections per image (fppi). Experiments with the Bayesian Network (Fig. 1a) are conducted with and without the refinement of the position by inference on the graphical model in Fig. 1b. The results are compared with results achieved by the classification of all observations in a single feature vector $[x_{Dl}, y_{Du}, x_{Dr}, y_{Dl}, I_C, I_B]^T$ (with I_B only evaluated for the static camera) with a Naive Bayes model [3], a Gaussian Mixture model (GMM) [18] and Random Forests. The results are plotted in Fig. 5.

We conclude from the plots, that among the three classification techniques Naive Bayes, GMM and RF, the Random Forest features the highest recall rates, though also the false-positive (FP) rates are higher than those of the rest. The Naive Bayes and the GMM classifier deliver similar recall rates with more false positives per image but also a higher recall rate in case of the CAVIAR-sequence on the side of the former one. Using the Bayesian Network from Fig. 1a, the results for the fppi-rates in the ETHZ-sequence are a few percent higher than those of the RF classifier, but the recall rate could be increased slightly as well. The application of the Bayesian Networks on the CAVIAR-sequence delivers less false positives and similar recall values than the RF classier. The extension of the model with the observation of the optical flow points (Fig. 1b) increases the recall rate in the upper part of the curve only in the ETHZ-sequence, while also the FP-rate increases slightly. W.r.t. the quality ($Q = \frac{TP}{TP+FN+FP}$, according to the measure of true positives (TP), false negatives (FN) and FPs) of object detection, in the CAVIAR-sequence the best results are achieved by the Bayesian Networks, both of which achieve the same score ($Q = 64\%$) superior to the quality achieved with the RF (63%). In the ETHZ-sequence, the quality does not differ considerably between the Random Forests and the Bayesian Networks (all around 59%). Compared to the results one achieves by varying the internal threshold of the HOG/SVM, plotted as red dashed lines, where the maximal recall score is achieved at a fppi rate of 6.6 in the CAVIAR-sequence and 6.3 in the ETHZ-sequence, respectively, all of the applied classifiers reduce the fppi at least by a factor of four, while the recall drops, in the best case, only about two percent in the CAVIAR test case and about six percent in the ETHZ test case, respectively. The achieved recognition accuracy is comparable to that of the single frame but stereo based detector in [19] and outperforms [9], see Fig. 5b.

3.3 Detector Position Accuracy

In this section we evaluate the average error of the measured and inferred positions by comparison with reference data. For each true positive detection, the deviations of the row coordinates of the head and the feet points from the reference data are recorded in a histogram. From the 95%-quantile of this histogram a Gaussian $N(\mu_{0.95}, \sigma^2_{0.95})$ with mean $\mu_{0.95}$ and standard deviation $\sigma_{0.95}$ is fitted. For the evaluation of the position accuracy the mean value of the Gaussian is tested for accordance with reference data using a statistical test of differences between two mean values. For the reference data a labeling uncertainty of one pixel is assumed so that the reference values follow a standard normal distribution $N(\mu_{ref}, \sigma^2_{ref}) = N(0, 1)$. As metric for similarity between the mean of the measurements and the mean of the reference data we apply

$$y_d = \frac{\mu_{0.95} - \mu_{ref}}{\sigma_d} \tag{7}$$

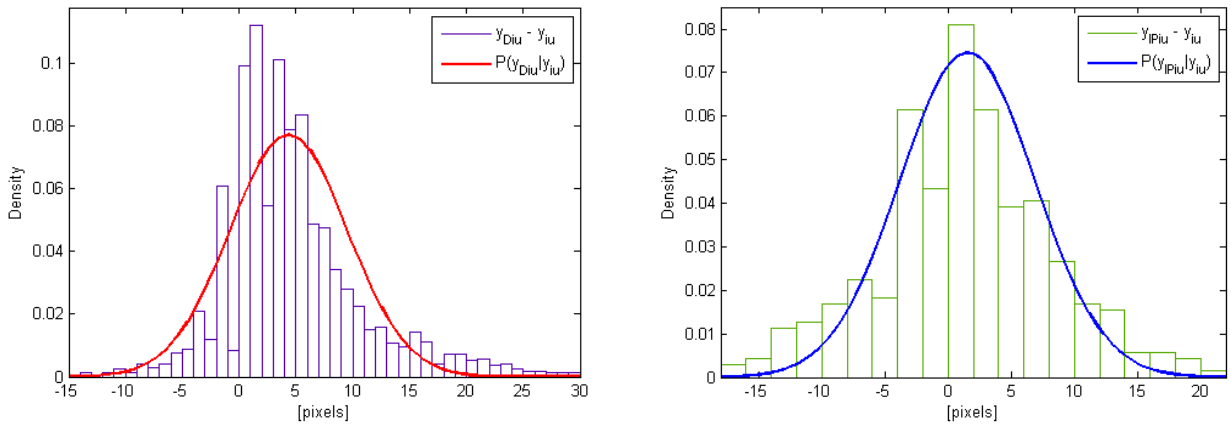
with

$$\sigma_d^2 = \sigma^2_{0.95} + \sigma^2_{ref} \tag{8}$$

resulting from variance propagation of uncorrelated observations.

The average deviation of the observations y_{Dj} and y_{IPj} from the reference data has already been considered as corrections in Sec. 2.2, see also Fig. 4. We calculate the metric

[1] We apply the HOG/SVM-detector of OpenCV, trained with the INRIA person dataset (http://pascal.inrialpes.fr/data/human/) with a height of 96 pixels for the people. We scale the input images by the factor 2 and achieve detections of people appearing with a minimal height of 48 pixels.



(a) (b)

Figure 4: Example histograms of differences in the row coordinates (a) between HOG-detection result and reference and (b) between IP-based locations and reference data. The differences are normalised according to a height of 96 pixel. The distributions are approximated by normal distributions.

	$\mu_{0.95}[\%]$	$\mu_{0.95}[px]$	$\sigma_{0.95}[px]$	y_d
HOG-Head	0.8	0.8	1.6	0.42
HOG-Feet	2.6	2.5	3.6	0.67
IP-Head	5.9	-5.7	5.1	-1.10
IP-Feet	2.0	1.9	4.3	0.43
Inferred Head	1.4	-1.3	2.2	-0.54
Inferred Feet	0.5	0.5	2.7	0.17

Table 1: CAVIAR-sequence: Difference between measured and inferred row coordinates from reference values in percent of the object height and in pixels.

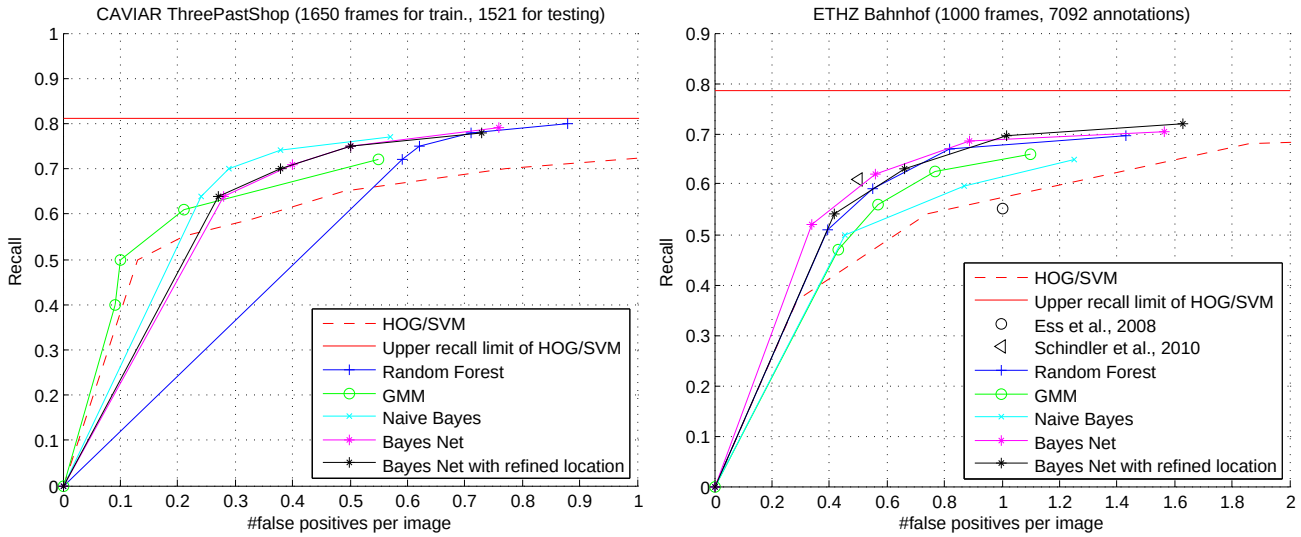
	$\mu_{0.95}[\%]$	$\mu_{0.95}[px]$	$\sigma_{0.95}[px]$	y_d
HOG-Head	5.2	5.0	5.4	0.91
HOG-Feet	0.4	0.4	3.3	0.12
IP-Head	0.9	0.9	6.4	0.14
IP-Feet	1.0	-1.0	5.3	-0.19
Inferred Head	0.6	-0.6	4.9	-0.12
Inferred Feet	0.4	-0.4	2.7	-0.14

Table 2: ETHZ-sequence: Difference between measured and inferred row coordinates from reference values in percent of the object height and in pixels.

y_d for the observed positions and the inferred positions as measure for accordance with the reference data and compare the results in Tab. 1 and 2. The deviations from the reference data are normalised w.r.t. a standard height of 96 pixels. From Eq. (5) it can be concluded, that among the mean values of the positions given by the HOG-detector (referred to as HOG-head and HOG-feet in the tables) and the ones given by the analysis of optical flow (IP-head and -feet) the one with the lower standard deviation has the stronger influence on the posterior, which in any of the test cases (Inferred Head and Feet in the CAVIAR- and ETHZ-sequence) is the position given by the detector. From the tables we conclude that the applied approach does not improve the position accuracy, if the position given by the detector already lies in the sub-pixel domain. In turn, when the error lies in the magnitude of some pixels, the inference of the posterior does enhance the alignment of the posterior positions with the reference. In either case, where the mean deviation of the position given by the detector lies around two and five pixels (see HOG-Feet in Tab. 1 and HOG-Head in Tab. 2), the posterior positions coincide much better with the reference data, which is reflected in the smaller values for y_d.

4 Conclusions

We conclude from this work that by the joint evaluation of the available information in the image that is linked to hidden parameters by a graphical model, the detection performance can be improved, even without the understanding of the 3D scene geometry and with a single camera as measuring device. Our approach leads to recognition results that are comparable with the state-of-the-art and at the same time improves the geometric accuracy of the results. The proposed method is a good starting point for tracking-by-detection systems, because the number of false positive detections from the underlying people detector that would lead to spurious trajectories are reduced significantly. The average geometric accuracy of the estimated location of pedestrians in the image is in any case around one pixel. We estimated the unknown parameters, i.e. the validity flag of a detection and its refined position, in two different graphical models. We plan to combine these models and estimate the parameters in a joint probabilistic model in future work, which opens the possibility for that position to be assigned to the detection, which most supports the joint probability.



(a) Recall and fppi values achieved in the CAVIAR-sequence

(b) Recall and fppi values achieved in the ETHZ-sequence

Figure 5: Recall and fppi values of the investigated classifiers and our proposed methods plotted in (a) for the CAVIAR-sequence and in (b) for the ETHZ-sequence. Also the HOG/SVM applied with internal thresholding (red dashed line) is drawn as baseline. The red horizontal lines indicate the maximum recall values reached by the HOG/SVM, towards which the red dashed line converges at fppi=6.6 in (a) and at fppi=6.3 in (b), respectively. In (b) also two samples from the results of related work are depicted.

References

[1] Caviar test case scenarios. http://homepages.inf.ed.ac.uk/rbf/CAVIAR/, 2004.

[2] M. Andriluka, S. Roth, and B. Schiele. People-tracking-by-detection and people-detection-by-tracking. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[3] C.M. Bishop. *Pattern recognition and machine learning*, volume 4. springer New York, 2006.

[4] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[5] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564 – 577, 2003.

[6] Q. Dai and D. Hoiem. Learning to localize detected objects. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3322–3329. IEEE, 2012.

[7] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In Carlo Tomasi Cordelia Schmid, Stefano Soatto, editor, *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886 –893 vol. 1. IEEE Computer Society, Los Alamitos, june 2005.

[8] P. Dollar, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 34 (4).*, pages 743–761, 2011.

[9] A. Ess, B. Leibe, K. Schindler, , and L. van Gool. A mobile vision system for robust multi-person tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08)*, pages 1–8. IEEE Press, June 2008.

[10] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

[11] D. Hoiem, A. A. Efros, and M. Hebert. Putting objects in perspective. *International Journal of Computer Vision*, 80(1):3–15, 2008.

[12] S. Khan and M. Shah. A multiview approach to tracking people in crowded scenes using a planar homography constraint. In *European Conference on Computer Vision, 2006*, volume 3954 of *LNCS*, pages 133–146, 2006.

[13] B. Leibe, E. Seemann, and B. Schiele. Pedestrian detection in crowded scenes. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 878–885. IEEE, 2005.

[14] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence*, volume 81, pages 674–679, 1981.

[15] M. Menze, T. Klinger, D. Muhle, J. Metzler, and C. Heipke. A stereoscopic approach for the association of people tracks in video surveillance systems. *PFG Photogrammetrie, Fernerkundung, Geoinformation*, 2013(2):83–92, 05 2013.

[16] B. Ommer, T. Mader, and J. M. Buhmann. Seeing the objects behind the dots: Recognition in videos from a moving camera. *International Journal of Computer Vision*, 83(1):57–71, 2009.

[17] D. Ramanan. Using segmentation to verify object hypotheses. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[18] D. A. Reynolds. Gaussian mixture models. in: Encyclopedia of biometrics, pp. 659-663. springer us,

2009.

[19] K. Schindler, A. Ess, B. Leibe, and L. Van Gool. Automatic detection and tracking of pedestrians from a moving stereo rig. *ISPRS Journal of Photogrammetry and Remote Sensing*, 65(6):523–537, 2010.

[20] J. Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1994.

[21] C. Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 246–252, 1999.

[22] A. Torralba and P. Sinha. Statistical context priming for object detection. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 763–770. IEEE, 2001.

[23] P. Viola, M.J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision*, 63(2):153–161, 2005.

[24] T. Zhao and R. Nevatia. Tracking multiple humans in complex situations. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(9):1208–1221, 2004.

Computer Vision Winter Workshop 2014, Zuzana Kúkelová and Jan Heller (eds.)
Křtiny, Czech Republic, February 3–5
Czech Society for Cybernetics and Informatics

The VOT2013 challenge: overview and additional results

M. Kristan[1], R. Pflugfelder[2], A. Leonardis[3], J. Matas[4], F. Porikli[5], L. Čehovin[1], G. Nebehay[2], G. Fernandez[2], and T. Vojir[4]

[1] University of Ljubljana, Slovenia [2] Austrian Institute of Technology, Austria
[3] University of Birmingham, United Kingdom [4] Czech Technical University in Prague, Czech Republic
[5] NICTA and Australian National University, Australia

Abstract *Visual tracking has attracted a significant attention in the last few decades. The recent surge in the number of publications on tracking-related problems have made it almost impossible to follow the developments in the field. One of the reasons is that there is a lack of commonly accepted annotated data-sets and standardized evaluation protocols that would allow objective comparison of different tracking methods. To address this issue, the Visual Object Tracking (VOT) challenge and workshop was organized in conjunction with ICCV2013. Researchers from academia as well as industry were invited to participate in the first VOT2013 challenge which aimed at single-object visual trackers that do not apply pre-learned models of object appearance (model-free). In this paper we provide an overview of the VOT2013 challenge, point out its main results and document the additional previously unpublished experiments and results.*

1 Introduction

Visual tracking is a rapidly evolving field of computer vision that has been increasingly attracting attention of the vision community. One reason is that it offers many challenges as a scientific problem. Second, it is a part of many higher-level problems of computer vision, such as motion analysis, event detection and activity understanding. Furthermore, the steady advance of technology in terms of computational power, form factor and price, opens vast application potential for tracking algorithms. Applications include surveillance systems, transport, sports analytics, medical imaging, mobile robotics, film post-production and human-computer interfaces.

Single-object trackers that do not apply pre-learned models of object appearance (model-free) are of particular interest due to their large application domain. The activity in the field is reflected by the abundance of new tracking algorithms presented and evaluated in journals and at conferences, and summarized in the many survey papers, e.g., [13, 29, 11, 17, 30, 43, 26]. Despite the efforts invested in proposing new trackers, these have not been accompanied with established evaluation methodology.

One of the most influential performance analysis efforts for object tracking is PETS (Performance Evaluation of Tracking and Surveillance) [44]. The first PETS workshop that took place in 2000, aimed at evaluation of visual tracking algorithms for surveillance applications. However, its focus gradually shifted to high-level event interpretation algorithms. Other frameworks and datasets have been presented since, but these focussed on evaluation of surveillance systems and event detection, e.g., CAVIAR[1], i-LIDS [2], ETISEO[3], change detection [15], sports analytics (e.g., CVBASE[4]), or specialized on tracking of specific objects like faces, e.g. FERET [33] and [20]. In general, the evaluation of new tracking algorithms, and their comparison to the state-of-the-art, depends on three essential components: (1) an evaluation system, (2) a dataset, (3) performance evaluation measures.

The Evaluation system. For objective and rigorous evaluation, an evaluation system that performs the same experiment on different trackers using the same dataset is required. Ideally, the system should support multiple OS and programming languages and allow easy integration of new trackers. Furthermore, a certain level of interaction with the tracker is desirable, for instance to allow for detection of tracking failures. Currently, the most notable and general systems are the ODViS [18], VIVID [4] and ViPER [8] toolkits. These, however, do not allow for interaction with the tracker. Recently, Wu et al. [41] have performed a large-scale benchmark of several trackers and developed an evaluation kit that allows integration of third-party trackers as well. However, in our experience, the integration is not straightforward due to a lack of standardization of the input/output communication between the tracker and the evaluation kit.

Dataset A trend has emerged in the single-object model-free tracking community to test newly proposed trackers on larger datasets that include different real-life visual phenomena like occlusion, clutter and illumination change. As a consequence, various authors nowadays compare their trackers on many publicly-available sequences, of which some have became a de-facto standard in evaluation of new

[1] http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1
[2] http://www.homeoffice.gov.uk/science-research/hosdb/i-lids
[3] http://www-sop.inria.fr/orion/ETISEO
[4] http://vision.fe.uni-lj.si/cvbase06/

trackers. However, many of these sequences lack a standard ground truth labeling, which makes comparison of proposed algorithms difficult. Furthermore, authors usually do not use datasets with various visual phenomena equally represented. In fact, many popular sequences exhibit the same visual phenomenon, which makes the results biased toward some particular types of the phenomena. To address this issue, Wu et al. [41] annotated each sequence with several visual attributes and report tracker performance with respect to each attribute separately. For example, a sequence is annotated as "occlusion" if the target is occluded anywhere in the sequence, etc. However, visual phenomena like occlusion do not usually last throughout the entire sequence. For example, an occlusion might occur at the end of the sequence, while a tracker might fail due to some other effects occurring at the beginning of the sequence. In this case, the failure would be falsely attributed to occlusion. Thus a per-frame dataset labeling is required to facilitate a more precise analysis.

Performance measures. A wealth of performance measures have been proposed for single-object tracker evaluation. These range from basic measures like center error [34], region overlap [25], tracking length [24] and failure rate [22, 21] to more sophisticated measures, such as CoTPS [31], which combine several measures into a single measure. A nice property of the combined measures is that they provide a single score to rank the trackers. A downside is that they offer little insight into the tracker performance. In this respect the basic measures, or their simple derivatives, are preferred as they usually offer a straight-forward interpretation. While some authors choose several basic measures to compare their trackers, the recent study [37] has shown that many measures are correlated and do not reflect different aspects of tracking performance. In this respect, choosing a large number of measures may in fact again bias results toward some particular aspects of tracking performance.

VOT2013. In order to address the above stated issues, the Visual Object Tracking (VOT2013) challenge was organized. Its aim was to provide an evaluation platform that goes beyond the current state-of-the-art. In particular, the authors of the challenge have compiled a labeled dataset collected from widely used sequences showing a balanced set of various objects and scenes. All the sequences are labeled per-frame with different visual attributes to aid a less biased analysis of the tracking results. An evaluation kit[5] was developed in *Matlab/Octave* that automatically performs experiments on a tracker using the provided dataset. A new tracker performance comparison protocol based on basic performance measures was also proposed. A significant novelty of the proposed evaluation protocol was that it explicitly addresses the statistical significance of the results and addresses the equivalence of trackers. A dedicated VOT2013 homepage[6] has been set up, from which the dataset, the evaluation kit and the results are publicly available. The authors of tracking algorithms have an opportunity to publish their source code at the VOT homepage as well, thus pushing the field of visual tracking towards reproducible research. The results of the challenge have been presented at the VOT2013 workshop in conjuction with the ICCV2013 and documented in the supporting paper [23]. In this paper we provide an overview of the VOT2013 challenge with a particular focus on the evaluation methodology and provide additional results that have not been published in [23].

[5] https://github.com/vicoslab/vot-toolkit
[6] http://www.votchallenge.net/

2 Summary of the tracking experiments

The VOT2013 benchmark is designed for single-object, single-camera, short-term causal trackers. The tracker is initialized at the beginning of a sequence using the ground truth bounding box and is required to predict a single bounding box of the target for each frame of the sequence. Causality requires the tracker to solely process the frames from the initialization up to the current frame without using any information from the future frames. Since we are evaluating short-term tracking, whenever the tracker fails, a complete reinitialization is performed so that any previously learned information (such as appearance and dynamics) is discarded. The challenge consists of three experiments:

- Baseline: Ground truth bounding boxes are used for initialization.
- Noise: Randomly perturbed bounding boxes are used for initialization, where the perturbation is in order of ten percent of the ground truth bounding box size.
- Grayscale: Color information is removed from the sequences.

The evaluation kit runs each tracker 15 times on each experiment to obtain a reliable estimate of the performance.

3 The dataset

We collected a large pool of sequences that have been used by various authors in the tracking community. Many sequences may be visually similar and would not contribute to diversification of the dataset, while significantly prolong the execution of the experiments. We have therefore reduced the set to 16 sequences, while keeping the dataset rich in visual phenomena. We represented each sequence in the pool of sequences as a 6-dimensional vector of global visual features and clustered them into 16 clusters by affinity propagation [10]. From each cluster a single sequence was manually selected.

The six global visual features were defined as follows: The *illumination change* as the maximal difference in average intensities computed from the bounding boxes; the *object size change* as the average of sequential differences in the ground-truth bounding box size; the *object motion* as the average of changes in bounding box center over the frames; *clutter* as the histogram difference within and outside the ground truth bounding box; *camera motion* as the per-pixel average difference outside the bounding box; *blur* was measured by a camera focus measure.

Since the bounding boxes were annotated by various authors, there was no common guideline for the process of manually annotating the sequences. It seemed that most authors followed the strategy of maintaining a high fore-

ground/background ratio within the bounding box (at least $> 60\%$). In most cases, this ratio is quite high since the upright bounding box tightly fits the target. But in some cases, (e.g., the *gymnastics* sequence) where an elongated target is rotating significantly, the bounding box contains a large portion of the background at some frames as well. After inspecting all the bounding box annotations, we have re-annotated those sequences in which the original annotations were out of place.

Additionally, we manually or semi-manually labeled each frame in each selected sequence with five visual attributes that reflect a particular challenge in appearance degradation: occlusion, illumination change, motion change, size change and camera motion. In case a particular frame did not correspond to any of the five degradations, we denoted it as non-degraded.

4 Evaluation methodology

There exists an abundance of performance measures in the field of visual tracking (e.g., [40, 32, 15, 20, 41]). Our approach to choosing the performance measures was the interpretability of the measures while selecting as few measures as possible to provide a clear comparison among trackers. Based on the recent analysis of widely-used performance measures [37] we have chosen two weakly-correlated measures: (i) accuracy and (ii) robustness.

The accuracy measures how well the bounding box predicted by the tracker overlaps with the ground truth bounding box. The tracking accuracy at time-step t is defined as the overlap between the tracker predicted bounding box A_t^T and the ground truth bounding box A_t^G

$$\phi_t = \frac{A_t^G \cap A_t^T}{A_t^G \cup A_t^T}. \tag{1}$$

On the other hand, the robustness was measured by the failure rate measure, which counts the number of times the tracker drifted from the target and had to be reinitialized. A failure is indicated when the overlap measure (Eq. 1) drops to zero.

The reinitialization of trackers might introduce a bias into the performance measures. Typically, if a tracker fails at a particular frame it will likely fail again immediately after re-initialization. To reduce this bias, we re-initialize the tracker five frames after the failure and denote the skipped frames as invalid for accuracy computation. This number was determined experimentally on a separate dataset. A similar bias occurs in the accuracy measure, as the overlap measure in the frames right after the initialization are biased towards higher values for several frames (burn-in period, Figure 1). In a preliminary study we have determined by a large-scale experiment that the burn-in period is approximately ten frames. The burn-in frames are also labeled invalid and are not used in the computation of accuracy.

Let $\Phi_t(i,k)$ denote the accuracy of i-th tracker at frame t at experiment repetition k. The per frame accuracy is obtained by taking the average over these, i.e., $\Phi_t(i) = \frac{1}{N_{\text{rep}}} \sum_{k=1}^{N_{\text{rep}}} \Phi_t(i,k)$. The average accuracy of the i-th tracker, $\rho_{\text{A}}(i)$, over some set of N_{valid} valid frames is then

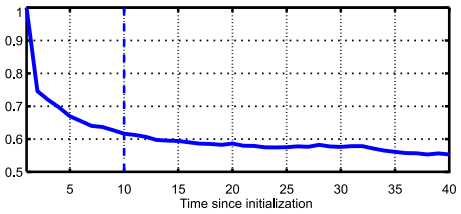


Figure 1: Overlaps after reinitialization averaged over a large number of trackers and many reinitializations.

calculated as the average of per-frame accuracies

$$\rho_{\text{A}}(i) = \frac{1}{N_{\text{valid}}} \sum\nolimits_{j=1}^{N_{\text{valid}}} \Phi_j(i). \tag{2}$$

In contrast to accuracy measurements, we obtain a single measure of robustness per experiment repetition. Let $F(i,k)$ be the number of times the i-th tracker failed in the experiment repetition k over a set of frames. The average robustness of the i-th tracker is then

$$\rho_R(i) = \frac{1}{N_{\text{rep}}} \sum_{k=1}^{N_{\text{rep}}} F(i,k). \tag{3}$$

Note that in the dataset some attributes are more frequently presented than the others, which would introduce a bias into the results. To address this, we calculate the accuracy (2) and robustness (3) separately for each attribute. For a particular attribute we calculate the two measures only on the subset of frames in the dataset that contain that attribute (attribute subset). To compare different trackers one might average the accuracy and robustness over all the attribute subset frames. However, these will likely be at a different scale across the attribute sequences in which case direct averaging of performance measures is not appropriate. Instead, we have developed a ranking-based methodology akin to [6, 9, 15]. We start by ranking all the trackers with respect to each measure on each attribute subset separately. Let $r(i,a,m)$ be the rank of the i-th tracker on the attribute subset a using the performance measure m, which can either be accuracy (A) or robustness (R). Now we can calculate the average rank for the i-th tracker by averaging over the attributes $r(i,m) = \frac{1}{N_{\text{att}}} \sum_{a=1}^{N_{\text{att}}} r(i,a,m)$. Giving an equal weight to each performance measure, we average the two corresponding rankings as

$$r(i) = \frac{1}{2} \sum_{m \in \{A,R\}} r(i,m). \tag{4}$$

The averaging over attribute subsets assures that every attribute contributes equally to the final ranking. Since the frequency of the attributes is uneven and some frames contain several attributes, it means that some frames contribute more than the other to the final rank. This is a subtlety that might not be immediately apparent, but has to be kept in mind when interpreting the results.

A group of trackers may perform equally well on a given attribute subset, in which case they should be assigned an equal rank. In particular, after ranking trackers on an attribute set, we calculate for each i-th tracker its corrected

rank as follows. We determine for each tracker, indexed by i, a group of equivalent trackers, which contains the i-th tracker as well as any tracker that performed equally well as the selected tracker. The corrected rank of the i-th tracker is then calculated as the average of the ranks in the group of equivalent trackers. Note that this equality is not transitive. For example, consider trackers T_1, T_2 and T_3. It may happen that a tracker T_2 performs equally well as T_1 and T_3, but this does not necessarily mean that T_1 performs equally well as both, T_2 and T_3 – the equivalence groups should be established for each tracker separately.

To determine for each tracker the group of equivalent trackers, we require an objective measure of equivalence on a given sequence. In case of accuracy measure, a per-frame accuracy is available for each tracker. One way to gauge equivalence in this case is to apply a paired test to determine whether the difference in accuracies is statistically significant. In case the differences are distributed normally, the Student's t-test, which is often used in the aeronautic tracking research [3], is the appropriate choice. However, in a preliminary study we have applied Anderson-Darling tests of normality [1] and have observed that the accuracies in frames are not always distributed normally, which might render the t-test inappropriate. As an alternative, we apply the Wilcoxon Signed-Rank test as in [6]. In case of robustness, we obtain several measurements of the number of times the tracker failed over the entire sequence in different runs. However, these cannot be paired, and we use the Wilcoxon Rank-Sum (also known as Mann-Whitney U-test) [6] instead to test the difference in the average number of failures.

When establishing equivalence, we have to keep in mind that statistical significance of performance differences does not directly imply a practical difference [7]. One would have to define a maximal difference in performance of two trackers at which both trackers are said to perform practically equally well. By varying the practical difference from zero to 0.1 we have not observed significant changes in ranking. However, since we could not find clear means to objectively define this difference, we reserve our methodology only to testing the statistical significance of the differences.

5 Result analysis

In the following section we overview the results of the VOT2013 challenge. We briefly overview the results from [23] and focus on results that were not yet published.

5.1 Submitted trackers

In total 27 trackers were evaluated for the challenge, 19 original submissions and 8 baseline well-known trackers that were contributed by the VOT committee. In interest of space, we cite [23] for all trackers submitted and/or presented at VOT2013. We also refer the reader to the appendix of the VOT supporting paper [23] for short descriptions. The set of trackers was very diverse, ranging from trackers that use background-subtraction (MORP [23], STMT [23]), are based on optical-flow or motion cues (FoT [39], TLD [19], SwATrack [27]), key-points (SCTT [23], Matrioska [28]), use complex generative (IVT [34], MS [5], CCMS [23], DFT [35], ORIA [42], EDFT [23], AIF [23], CACTuS-FL [12], PJS-S [45], SwATrack) or discriminative (MIL[2], STRUCK [16], PLT [23], CT [46], RDET [23], ASAM [23], GSDT [23]) models to trackers that use geometrical constellation of parts (HT [14], LGT [38], LGT++ [23], LT-FLO [23]).

5.2 Conclusions for the challenge experiments

We overview only the major conclusions for experiments 1, 2 and 3 and refer the reader to [23] for further details. For reference, we show the accuracy-robustness rank plots in Figure 2. Averaging ranks across all three basic experiments, the top performing trackers were PLT, FoT, EDFT, LGT++ and LT-FLO. The top performing PLT is a single-scale, detection-based tracker that applies online structural SVM on color and grayscale pixels and grayscale derivatives. In all experiments the PLT achieved best robustness, however in the Baseline and Noise experiment, the part-based LGT++ and the original LGT tightly followed. The three trackers (PLT, LGT++ and LGT) perform well even in noisy initializations, but in accuracy, the top performing tracker on average was FoT, followed by SCTT and a RANSAC-based edge tracker LT-FLO. We have noticed that the top ranked trackers in the averaged ranks remain at the top also with respect to each attribute separately, with two exceptions. When considering the size change, the best robustness is still achieved by PLT, however, the trackers that yield best trade-off between the robustness and accuracy are the LGT++ and the size-adaptive mean shift tracker CCMS. When considering occlusion, the PLT and STRUCK seem to share the first place in the best trade-off. Note that the evaluation kit was also measuring the tracker speed during the experiments. Since the trackers were coded in different programming languages and run on different machines, the measurements are not directly comparable. However, two trackers stood out in performance. These were the PLT and FoT, whose speed was higher than 150fps.

We ranked the individual types of visual degradation according to the tracking difficulty they present to the tested trackers. Our results imply that the subsequences that do not contain any specific degradation present little difficulty for the trackers in general. Most trackers do not fail on such intervals and achieve best average overlap. On the other hand, camera motion is the hardest degradation in this respect. One way to explain this is that most trackers focus primarily on appearance changes of the target and do not explicitly account for changing background. Note that camera motion does not necessarily imply that the object is significantly changing position in the image frame. For accuracy, the hardest degradation is the change of object size. This is plausible as many trackers do not adapt in this respect and sacrifice their accuracy for a more stable visual model that is more accurate in situations where the size of the target does not change.

In summary, the sparse discriminative tracker PLT seems to address the robustness quite well, despite that it does not adapt the target size, which reduces its accuracy when the size of the tracked object is significantly changing. On the other hand, the part-based trackers with a rigid part constel-

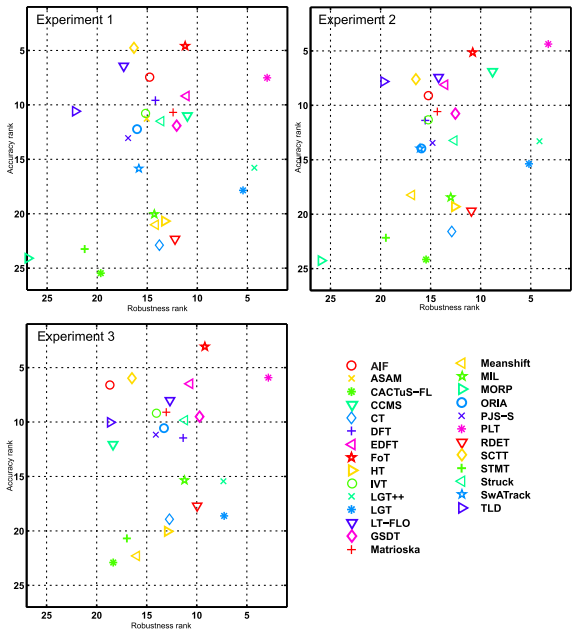


Figure 2: The accuracy-robustness ranking plots with respect to the three experiments. An optimal tracker would reside in the top-right corner of the plot.

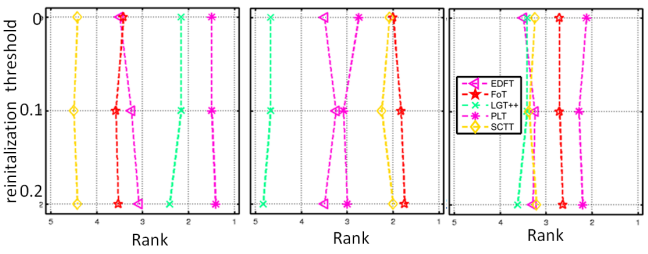


Figure 3: Effects of failure threshold on ranking.

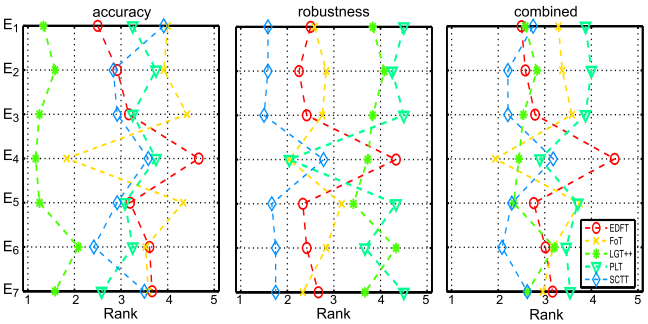


Figure 4: Results of top performing trackers on Baseline (E_1), Noise (E_2), Gray (E_3), Empty frames (E_4), Frame skipping (E_5), Frame Resize (E_6) and Reverse order (E_7) experiments.

lation yield a better accuracy at reduced robustness. The robustness is increased with part-based models that relax the constellation, but this on average comes at a cost of significant drop in accuracy.

5.3 Additional experiments

In addition to the official challenge experiments, the VOT committee has also performed four additional experiments on the top five submitted entries that had been attached a working executable version of the tracker. The aim of the first experiment was to evaluate the effect of the failure threshold on the overall ranking outcome. The remaining four experiments were designed to offer further insights into the tracker performance.

5.3.1 Effects of the failure threshold Recall that the evaluation kit proclaimed a failure if the overlap between the predicted and ground-truth bounding box became zero. To study how increasing the threshold affects the ranking of the trackers, we have repeated the baseline experiment with thresholds 0.1 and 0.2. The results are shown in Figure 3. We have observed that the failure rate increased with the threshold, however, the increase is approximately the same for all five trackers. From Figure 3 we see that the two top performing trackers do not change rank, but there is a slight change in ranking of the last three trackers. This change is due to ranking change in the accuracy rankings, since the practical difference in accuracy is in fact small for these trackers. We can conclude that the applied ranking scheme is sufficiently stable across reasonable values of failure thresholds.

5.3.2 Sequence degradation We have considered four diverse challenging scenarios of sequence degradation:

- **Empty frames**: The adaptability of the employed visual models is tested by replacing every fifth frame in the sequence by a black image.
- **Skipping frames**: To simulate frame-drops that can occur in video transmission, the original sequences were modified by removing every third frame from the sequence.
- **Reduction of image size**: To study how the size of the target affects the tracking, the size of the images is reduced by 60%.
- **Reversed sequence**: To test the importance of the specific sequence of events in the sequence, the order of the frames is reversed.

The overall results for the four additional results above are shown in Figure 4. In all but one experiment the ranking results do not change a lot, meaning that the trackers are equally well adapted to these degradations. In the experiment with black frames, the performance of the FoT and PLT significantly decreased relative to other trackers, while the performance of EDFT relatively increased. Note that the absolute performance decreased for all trackers, but this reduction was greater for FoT and PLT than it was for EDFT. The significant jump in ranking for the FoT can be explained by the way this tracker adapts its visual model. In particular, the FoT performs full adaptation in each frame. Once a black frame occurs the visual model becomes completely corrupted, which leads to failure. In case of PLT the decrease is most likely a result of fixed color model that is initialized at the first frame and is used to determine regions that most likely belong to the object. Once a black frame arrives, the discriminative power of model is rendered useless, which, may lead to unrepairable false adaptations of the visual model. EDFT on the other hand is better suited for this

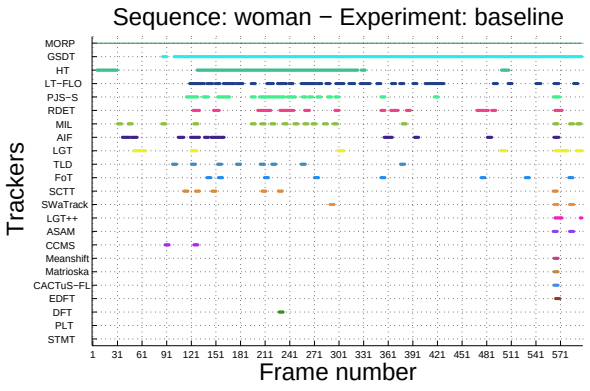


Figure 5: Scatter plot for the *woman* sequence shows the failures for each tracker w.r.t. frame number.

kind of changes, likely because of lazy adaptation of the visual model and a well designed motion model, which help it to survive short-term image degradations.

5.4 Sequence analysis

The second part of our analysis focused on the selected sequences. In particular, we have first analyzed the difficulty of each sequence presented to the trackers. Namely, for each sequence we have recorded if a particular tracker fails at least once at a particular frame. Using this approach we have constructed for each sequence a scatterplot of failures over each tracker (see the example in Figure 5). We visualize the level of difficulty for each sequence by summing the scatterplots vertically. This yields the failure curve (e.g., Figure 6a,b,c,d) which shows how many trackers failed at each frame. From these curves we derive two objective measures of sequence difficulty: *area* and *max*. The *area* is just the sum of frame-wise values from the failure curve normalized by the number of frames, while the *max* is the maximum on this curve, i.e. the maximal number of failed trackers in a frame. While the *area* indicates the average level of difficulty of a sequence, the *max* is localized and focuses on a particular frame that presented most difficult part of the sequence. For example, the *area* in the failure curve for the David sequence (Figure 6a) is smaller than the *area* for the *woman* sequence (Figure 6b), which suggests that *david* sequence is less challenging that the *woman* sequence. Furthermore, a significant peak in the *woman* sequence (frame 565) suggests that this sequence contains a subsequence which is challenging to most of the trackers. Table 1 summarizes the *area* and *max* values for all sequences.

Given the measures of *area*, we intuitively clustered the remaining 15 sequences by hand into three clusters according to their level of difficulty: Hard, Intermediate, Easy. The results are summarized in Table 1. *david*, *face*, *bicycle*, *juice*, *jump*, *car* and *cup* sequences do not present a significant challenge to most of the trackers; on average, less than a single tracker fails. Surprisingly the *david* sequence (Figure 6a) shows a small *area* in this study, although the sequence is usually considered in literature to be challenging. The reason might be that this sequence is so frequently used for tracker evaluation that the authors might have over-

sequence	area	max	frame	difficulty
bolt	4.28	13	242	hard
diving	4.23	9	105	hard
hand	4.22	14	51	hard
gymnastics	3.13	12	98	interm.
woman	2.86	15	565	interm.
sunshade	2.79	11	85	interm.
torus	2.67	8	189	interm.
iceskater	2.38	6	227	interm.
singer	1.68	4	268	interm./easy
david	1.36	4	337	easy
face	1.22	3	140	easy
bicycle	1.22	11	178	easy
juice	1.12	4	242	easy
jump	0.93	4	203	easy
car	0.92	5	253	easy
cup	0.22	2	232	easy

Table 1: The sequence analysis results. The area under the failure curve (area), the maximal number of simultaneously failed trackers (max), the frame number with maximum number of failures (frame), and the difficulty label (difficulty).

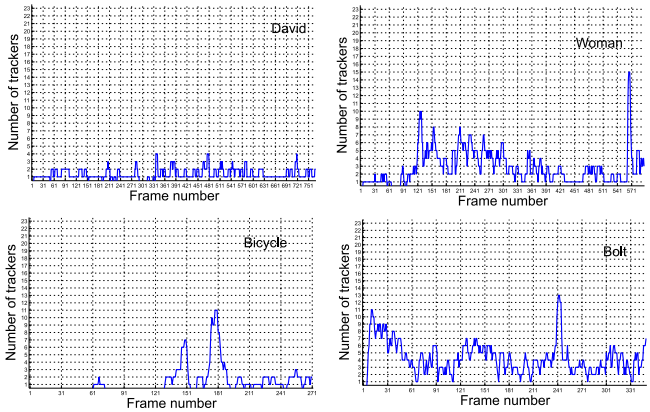


Figure 6: Failure curves for *david*, *woman*, *bicycle* and *bolt* sequences.

fitted to this sequence. The analysis also shows that the *bolt*, *diving* and *hand* sequences are the most challenging ones, followed by sequences of intermediate difficulty, in particular, the *gymnastics*, *woman*, *sunshade*, *torus*, and *iceskater* sequences and the *singer* sequence which seems to be easy-to-intermediate.

Most of the difficulties in hard and intermediate sequences arise from changes in camera and object motion as well as from rapid changes in object size. For example, *bolt* is hard, as all three aforementioned nuisances occur simultaneously in the sequence. The *diving* sequence shows significant changes in object size while the *hand* sequence shows challenging pose variations of the person's hand.

Easy to intermediate sequences might remain valuable for tracker comparison as these sequences still conceal challenges in particular frames. We can identify those sequences by considering *max* in Table 1. For example, the *woman* sequence at frame 565 (Figure 6b) shows camera zooming which lets 15 out of 23 trackers fail. Similarily, the *bicycle* sequence at frame 178 (Figure 6c) shows a peak in the failure curve. Here, an object occlusion happens and immediately after that, the tracked object is partially covered by a

shadow. A large peak is also present in the challenging Bolt sequence (Figure 6d) at frame 242. Almost half of the trackers fail here. A closer look at the frame and its neighboring frames shows significant object motion between frames as a cause of failures.

6 Conclusions and Future Work

This paper reviewed the VOT2013 challenge and its results. The challenge provides an evaluation kit comprising an evaluation protocol and dataset of 16 sequences which allows simple and objective tracker comparison. VOT2013 also provides attributes such as illumination change, occlusion, etc. on frame level for each sequence. First results show that trackers tend to specialize either for robustness or accuracy. None of the trackers consistently outperformed the others by all measures at all sequence attributes. It is currently impossible to conclusively say what kind of tracker design works best in general, however, there is some evidence showing that accuracy tends to be better for the trackers that do not apply global models, but rather split their visual models into parts. On the other hand, robustness is pretty much achieved by discriminative learning where variants of Structured SVM, e.g. PLT, seems very promising. The analysis of our dataset showed that some sequences are challenging on average, other sequences are very challenging at particular frames and some of them were well tackled by all the trackers. While we believe that it is difficult to overfit a tracker to a visually diverse dataset, tuning parameters may very likely contribute to higher ranks. Because of this unavoidable dependence on implementation and parameter tuning, care has to be taken when deciding for or against a new tracker based on performance scores. Rather than waging decision on absolute scores, comparative evaluation should be used to position trackers against baseline implementations and further focus on detailed analysis per visual properties. Our future work will focus on revising and carefully enriching the dataset with new sequences, e.g. including sequences from related datasets like the recent [36] with the aim of significantly increasing diversity while keeping the number of sequences on a useful level. We also intend to improve the evaluation kit, allowing faster execution of more complex experiments. Our work will focus on organizing further VOT challenges and pushing towards a standard for tracker comparison.

Acknowledgement

This work was supported in part by the following research programs and projects: Slovenian research agency research programs P2-0214, P2-0094, Slovenian research agency projects J2-4284, J2-3607, J2-2221 and the EPiCS project from European Union seventh framework programme under grant agreement no 257906. Jiri Matas and Tomas Vojir were supported by CTU Project SGS13/142/OHK3/2T/13 and by the Technology Agency of the Czech Republic project TE01020415 (V3C – Visual Computing Competence Center)

References

[1] T. W. Anderson and D. A. Darling. Asymptotic theory of certain "goodness of fit" criteria based on stochastic processes. *The Annals of Mathematical Statistics*, 23(2):193–212, 1952.

[2] B. Babenko, M. H. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(8):1619–1632, 2011.

[3] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation*, chapter 11, pages 438–440. John Wiley & Sons, Inc., 2001.

[4] R. Collins, X. Zhou, and S. K. Teh. An open source tracking testbed and evaluation web site. In *Perf. Eval. Track. and Surveillance*, 2005.

[5] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(5):564–577, 2003.

[6] J. Demšar. Statistical comparisons of classifiers over multiple datasets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[7] J. Demšar. On the appropriateness of statistical tests in machine learning. In *Workshop on Evaluation Methods for Machine Learning in conjunction with ICML*, 2008.

[8] D. Doermann and D. Mihalcik. Tools and techniques for video performance evaluation. In *Proc. Int. Conf. Pattern Recognition*, page 167170, 2000.

[9] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision*, 88(2):303–338, June 2010.

[10] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315:972976, 2007.

[11] P. Gabriel, J. Verly, J. Piater, and A. Genon. The state of the art in multiple object tracking under occlusion in video sequences. In *Proc. Advanced Concepts for Intelligent Vision Systems*, page 166173, 2003.

[12] A. Gatt, S. Wong, and D. Kearney. Combining online feature selection with adaptive shape estimation. In *Image and Vision Computing New Zealand (IVCNZ), 2010 25th International Conference of*, pages 1–8. IEEE, 2010.

[13] D. M. Gavrila. The visual analysis of human movement: A survey. *Comp. Vis. Image Understanding*, 73(1):82–98, 1999.

[14] M. Godec, P. M. Roth, and H. Bischof. Hough-based tracking of non-rigid objects. *Comp. Vis. Image Understanding*, 117(10):1245–1256, 2013.

[15] N. Goyette, P. M. Jodoin, F. Porikli, J. Konrad, and P. Ishwar. Changedetection.net: A new change detection benchmark dataset. In *CVPR Workshops*, pages 1–8. IEEE, 2012.

[16] S. Hare, A. Saffari, and P. H. S. Torr. Struck: Structured output tracking with kernels. In Dimitris N. Metaxas, Long Quan, Alberto Sanfeliu, and Luc J. Van Gool, editors, *Int. Conf. Computer Vision*, pages 263–270. IEEE, 2011.

[17] W. Hu, T. Tan, L. Wang, and S. Maybank. A survey

on visual surveillance of object motion and behaviors. *IEEE Trans. Systems, Man and Cybernetics, C*, 34(30):334–352, 2004.

[18] C. Jaynes, S. Webb, R. Steele, and Q. Xiong. An open development environment for evaluation of video surveillance systems. In *PETS*, 2002.

[19] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(7):1409–1422, 2012.

[20] R. Kasturi, D. B. Goldgof, P. Soundararajan, V. Manohar, J. S. Garofolo, R. Bowers, M. Boonstra, V. N. Korzhova, and J. Zhang. Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(2):319–336, 2009.

[21] M. Kristan, S. Kovacic, A. Leonardis, and J. Perš. A two-stage dynamic model for visual tracking. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 40(6):1505–1520, 2010.

[22] M. Kristan, J. Perš, M. Perše, M. Bon, and S. Kovačič. Multiple interacting targets tracking with application to team sports. In *International Symposium on Image and Signal Processing and Analysis*, pages 322–327, September 2005.

[23] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, F. Porikli, L. Cehovin, Georg Nebehay, Fernandez G., T. Vojir, and et al. The visual object tracking vot2013 challenge results. In *ICCV2013 Workshops, Workshop on visual object tracking challenge*, pages 98 –111, 2013.

[24] J. Kwon and K. M. Lee. Tracking of a non-rigid object via patch-based dynamic appearance modeling and adaptive basin hopping monte carlo sampling. In *Comp. Vis. Patt. Recognition*, pages 1208–1215. IEEE, 2009.

[25] H. Li, C. Shen, and Q. Shi. Real-time visual tracking using compressive sensing. In *Comp. Vis. Patt. Recognition*, pages 1305–1312. IEEE, 2011.

[26] X. Li, W. Hu, C. Shen, Z. Zhang, A. R. Dick, and A. Van den Hengel. A survey of appearance models in visual object tracking. *arXiv:1303.4803 [cs.CV]*, 2013.

[27] M. Lim, C. Chan, D. Monekosso, and P. Remagnino. Swatrack: A swarm intelligence-based abrupt motion tracker. In *In proceedings of IAPR MVA*, page pages 3740, 2013.

[28] M. E. Maresca and A. Petrosino. Matrioska: A multi-level approach to fast tracking by learning. In *Proc. Int. Conf. Image Analysis and Processing*, pages 419–428, 2013.

[29] T. B. Moeslund and E. Granum. A survey of computer vision-based human motion capture. *Comp. Vis. Image Understanding*, 81(3):231–268, March 2001.

[30] T. B. Moeslund, A. Hilton, and V. Kruger. A survey of advances in vision-based human motion capture and analysis. *Comp. Vis. Image Understanding*, 103(2-3):90–126, November 2006.

[31] T. Nawaz and A. Cavallaro. A protocol for evaluating video trackers under real-world conditions. *IEEE Trans. Image Proc.*, 22(4):1354–1361, 2013.

[32] Y. Pang and H. Ling. Finding the best from the second bests – inhibiting subjective bias in evaluation of visual tracking algorithms. In *Int. Conf. Computer Vision*, 2013.

[33] P. J. Phillips, H. Moon, S. A. Rizvi, and P. J. Rauss. The feret evaluation methodology for face-recognition algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(10):1090–1104, 2000.

[34] D. A. Ross, J. Lim, R. S. Lin, and M. H. Yang. Incremental learning for robust visual tracking. *Int. J. Comput. Vision*, 77(1-3):125–141, 2008.

[35] L. Sevilla-Lara and E. G. Learned-Miller. Distribution fields for tracking. In *Comp. Vis. Patt. Recognition*, pages 1910–1917. IEEE, 2012.

[36] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. Visual Tracking: an Experimental Survey. *TPAMI*, 2013.

[37] L. Čehovin, M. Kristan, and A. Leonardis. Is my new tracker really better than yours? Technical Report 10, ViCoS Lab, University of Ljubljana, Oct 2013.

[38] L. Čehovin, M. Kristan, and A. Leonardis. Robust visual tracking using an adaptive coupled-layer visual model. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(4):941–953, 2013.

[39] T. Vojir and J. Matas. Robustifying the flock of trackers. In *Comp. Vis. Winter Workshop*, pages 91–97. IEEE, 2011.

[40] H. Wu, A. C. Sankaranarayanan, and R. Chellappa. Online empirical evaluation of tracking algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(8):1443–1458, 2010.

[41] Y. Wu, J. Lim, and M. H. Yang. Online object tracking: A benchmark. In *Comp. Vis. Patt. Recognition*, 2013.

[42] Y. Wu, B. Shen, and H. Ling. Online robust image alignment via iterative convex optimization. In *Comp. Vis. Patt. Recognition*, pages 1808–1814. IEEE, 2012.

[43] A. Yilmaz and M. Shah. Object tracking: A survey. *Journal ACM Computing Surveys*, 38(4), 2006.

[44] D. P. Young and J. M. Ferryman. Pets metrics: On-line performance evaluation service. In *ICCCN '05 Proceedings of the 14th International Conference on Computer Communications and Networks*, pages 317–324, 2005.

[45] A. Zarezade, H. R. Rabiee, A. Soltani-Frani, and A. Khajenezhad. Patchwise joint sparse tracker with occlusion detection using adaptive markov model. *preprint in arXiv*, 2013.

[46] K. Zhang, L. Zhang, and M. H. Yang. Real-time compressive tracking. In *Proc. European Conf. Computer Vision*, Lecture Notes in Computer Science, pages 864–877. Springer, 2012.

Computer Vision Winter Workshop 2014, Zuzana Kúkelová and Jan Heller (eds.)
Křtiny, Czech Republic, February 3–5
Czech Society for Cybernetics and Informatics

A Few Things One Should Know About Feature Extraction, Description and Matching

Karel Lenc, Jiří Matas, and Dmytro Mishkin

CMP CTU in Prague, Czech Republic

Abstract *We explore the computational bottlenecks of the affine feature extraction process and show how this process can be speeded up by 2-3 times with no or very modest loss of performance. With our improvements the speed of the Hessian-Affine and MSER detector is comparable with similarity-invariant SURF and DoG-SIFT detectors.*

The improvements presented include a faster anisotropic patch extraction algorithm which does not depend on the feature scale, a speed up of a feature dominant orientation estimation and SIFT descriptor computation using a look-up table.

In the second part of the paper we explore performance of the recently proposed first geometrically inconsistent nearest neighbour criterion and domination orientation generation process.

1 Introduction

Extraction of local image features is an important part of a wide variety of computer vision algorithms and significant effort has been put into speeding up this process. Speed up efforts have been usually directed at similarity covariant feature detectors or even only translation invariant detectors. Affine covariant detectors have been avoided because they are considered, the paper indicates somewhat unfairly, too computationally expensive (see Table 1).

Method	SIFT	SURF	HesAff	MSER	**HesAff+**	**MSER+**
Avg. NFeats	1719.23	2192.33	3181.81	1028.02	3787.60	1348.71
Avg. Time [s]	0.90	0.72	5.38	2.83	**1.80**	**0.69**

Table 1: Average processing time (without image IO) and number of features per an image of commonly used feature extractors and their variants implementing the proposed improvements. We compare similarity invariant SIFT (VlFeat implementation) and SURF (OpenSURF implementation) and standard implementations of affine invariant Hessian Affine (HesAff) (implementation by [18]) and MSER [11] with the improved affine invariant HesAff+ and MSER+. Values are computed on all images from Mikolajczyk's dataset [15] with average image resolution of 0.7MPx.

In the first part of this work, we show that with careful implementation the processing time of the traditional affine covariant detectors (such as Hessian-Affine and MSER) is comparable to similarity covariant detectors. We show that by loosening the demands on correctness of the patch extraction process from the signal processing point of view, the processing time can be significantly reduced with only a modest loss in the scale invariance of the extracted descriptors.

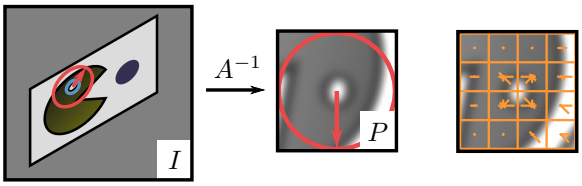


Figure 1: The classical local image feature extraction pipeline. Measurement region (red) of a detected feature (blue) is warped from image I to a patch P normalising the region based on the local affine shape of the feature described by matrix A. The image data in the patch are used to compute the SIFT descriptor.

In the second part, we investigate another bottleneck which is the conversion of gradients from cartesian to polar coordinates and we examine existing and propose a new $\arctan 2$ approximation which speed up this process.

Finally we examine strategies for generating tentative correspondences. The traditional method for SIFT matching is based on the second (to first) nearest neighbour (SNN) distance ratio. We confirm that using the first geometric inconsistent nearest neighbour [16] can improve the performance even on two view matching without view synthesis. We investigate the tentative correspondences between distinguished regions (DRs) and number of their matched dominant orientations. We show that the standard second nearest neighbour criterion is able to remove most of the inconsistent correspondences (e.g. two dominant orientations of a single DR matched to two different DRs in the tested image) and that the knowledge of the multiple orientation correspondences can help to avoid some degenerate hypotheses in RANSAC.

2 Related Work

The feature extraction process consists of several stages that are visualised in Figure 1. In the first step, distinguished regions (DRs) are detected using a feature detector. There are several ways how to perform that: Scale-space detectors (DoG [10], Hessian or Harris-Affine [13]) start with building a scale space pyramid and each layer of the pyramid is anal-

ysed for local maxima of some differential operator; in the Fast Hessian, which is how we call the detector part of the SURF system [2], the pyramid consists of integral images, which are used for computing an approximated Hessian operator with Haar wavelets. In the case of Hessian and Harris Affine detector, the structure tensor is used to estimate an affine shape of each DR. Another affine invariant detector, MSER [11], detects DRs finding maximally stable extremal regions where the affine shape is defined by their interior's second moments.

In order to obtain rotationally invariant descriptions, dominant orientations of each DR are detected. This is usually done by collecting a weighted histogram of finite derivatives in the feature's measurement region, which has m-times bigger measurement scale. This measurement region is used for descriptor computation and is then normalised to a small patch (usually of size 41 pixels) used to form invariant descriptor such as SIFT [10].

The process differs for the SURF descriptor, which uses Haar wavelets for the computation of both the dominant orientations and for the SURF descriptor; therefore, no patch needs to be extracted. The main advantage of the SURF detector is its speed, but like the DoG detector, it does not offer Affine invariance. However, this can be achieved by using ASIFT [17] or MODS [16] matching startegy which synthesise views so that images can be matched across significant viewpoint changes without a substantial increase in the processing time.

Another speed-aware approach to feature detection is the FAST detector [20], which is a classifier learned for corner detection. However, it examines a neighbourhood of constant size; therefore, it is not scale invariant. This restriction was addressed in BRISK detector [9].

The computationally expensive part of estimating dominant orientations or computing SIFT descriptor is the conversion of gradients from Cartesian to Polar coordinates. This operation can be accelerated in several ways, e.g. in [8], a circuit design is proposed to accelerate in hardware this operation. The $\arctan 2$ function can be approximated well by Taylor series expansions, which is often used in speed-aware implementations [19], [12]. Indeed, this is used in practice e.g. in *VLFeat* library [21], where the 3rd order Taylor polynomial is used to approximate the function $\arctan((1-r)/(1+r))$. Similarly, in *OpenCV* library[1], the $\arctan 2$ is approximated with a 7th degree Taylor series expansion. Moreover, it is implemented using SSE instructions.

The extracted local image features are usually used for image based matching, e.g. in wide baseline stereo problems. Feature extraction is followed by generation of tentative correspondences, usually using the Lowe's Second Nearest Neighbour criterion [10]. Then the tentative correspondences are verified against two-view geometry constraints in a RANSAC framework [5].

[1] http://opencv.org/

3 Speeding up patch extraction process

We start with investigation of the main bottlenecks of the feature extraction process. In order to compare existing feature extraction algorithms fairly, we measure time needed for processing hypothetical 1MPx image where a detector finds 2000 Distinguished regions and extract 3600 descriptors as each distinguished region has 1.8 dominant orientations on average. Because the processing time can also depend on the size and shape of the detected features, we compute an average over 16 images of various scenes.

Figure 2 shows the execution time of each stage of the VLFeat[2] DoG detector [21], OpenSURF[3] implementation of SURF detector [2], improved implementation of Hessian-Affine [13] by Perdoch et. al. [18] and MSER [11].

All detectors have been configured in such a way that they detect features within the same range of scales; the OpenSURF algorithm was used in two configurations: FHes+SURF where the initial sampling is set to default 2pxs (i.e. the response is computed for every second pixel) and FHes-1px+SURF where the sampling is set to 1px. All measurements in this article are done with measurement scale $m = 3\sqrt{3}$.

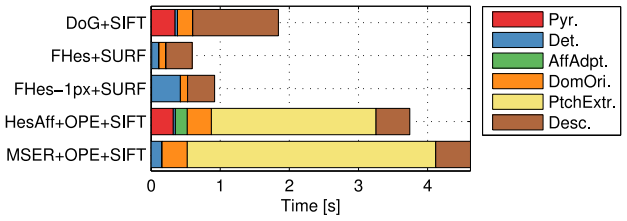


Figure 2: The processing time of feature extraction stages of commonly used algorithms (VLFeat DoG, OpenSURF, HessianAffine by Perdoch [18] and authors' implementation of MSER [11] which uses same SIFT implementation as the HessianAffine). The time was measured on a set of 16 1MPx images. *Pyr.* stands for the time needed for building a pyramid (the scale space or the integral images for the SURF detector), *Det.* is the duration of the feature detection stage. Both of these stages depend mainly on the image size. *AffAdapt.* is the duration of iterative affine adaptation and *DomOri.* is the time needed for detection of dominant orientations. *AffAdapt.* and and *DomOri.* are normalised to 2000 DRs. The last two stages, *PtchExtr.*, the time needed to extract patches used for the description (*Desc.*) are normalised to 3600 features.

Figure 2 clearly shows that the most expensive stage of the existing affine invariant feature detectors is the patch extraction process, note that this stage takes longer for MSER as it generally detects regions with higher scale [15] and uses the same patch extraction algorithm as Hessian-Affine, which means that the patch extraction time depends on the feature size.

In the following, we propose novel algorithms which reduce significantly the dependence of patch extraction time

[2] http://www.vlfeat.org/

[3] http://www.chrisevansdev.com/computer-vision-opensurf.html

on the local image feature scale. Then, several improvements which speed up the feature extraction process are presented.

3.1 The standard patch extraction process

In order to obtain invariant description of a co-variant local image feature, the image region corresponding to a feature is normalised. The local derivatives, used to obtain gradients for the SIFT descriptor, are computed using a Gaussian kernel of the size determined by differentiation scale σ_D [13] and the extracted patches should have $\sigma_D = \text{const}$ in order to gain full scale invariance.

In the case of the original Lowe's [10] SIFT feature detection and description framework, the patch used for feature description is extracted from the scale-space layer where the feature has been located. This means that for a feature which was found in octave o and in a layer l of a scale space with O octaves and L layers, the feature can have a scale in the input image in interval

$$s \in \left[\sigma_i 2^{o+\frac{l-1}{L}}, \sigma_i 2^{o+\frac{l+1}{L}}\right] \quad (1)$$

where σ_i is the initial scale (the prior smoothing used for building the scale-space pyramid) [10]. Then the descriptor is computed from a measurement region with scale $s_{mr} = s \cdot m$ in the original image where m is the magnification factor. This method can be used only for similarity-invariant features as it does not handle anisotropy of affine co-variant features.

In patch extraction implementations [13] and [18], at first, an affine image feature is normalised with $A^{-1} \cdot s$ from the input image where $A \in GL(2)$ is the de-normalisation matrix and $s = \sqrt{A}$ is the feature scale. In the next step, the extracted patch is blurred with an isotropic Gaussian kernel with variance $\sigma_B = 2\sigma_D ms/p$ in order to obtain differentiation scale σ_D in the down-sampled descriptor patch.

The disadvantage of the method ([13], [18]) is its computational complexity as it works with the original image data even for features which may cover the whole image. This can be seen in Figure 3 which clearly shows that even though there is only a fraction of detected features in higher octaves, the patch extraction process still takes a significant amount of time. All the computation times were measured on a machine with Intel® Core™i7-3517U CPU with 4MB cache.

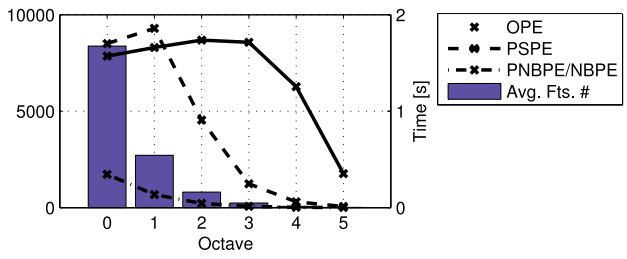


Figure 3: Time needed for extraction of patches found at different scale space octaves. OPE is the original patch extraction method [13], [18]. Results obtained with the Hessian Affine detector. Average on 18 various 3MPx images.

3.2 Speeding up the affine patch extraction process

We propose a novel algorithm for patch extraction which uses the pre-blurred layers of the isotropic Gaussian scale space pyramid. We will refer this algorithm as Pyramid-Smoothing Patch Extraction (PSPE). Based on the required σ_D it operates on the Gaussian scale space pyramid layer where the differentiation scale is small enough that after the extraction it would not exceed σ_D in any direction.

Then, the measurement region of the feature is extracted to the intermediate patch in its original scale in such a way that the eigenvectors of the affine transformation are aligned with the image axes. This transformation is found using the singular value decomposition. Then, equal differentiation scale in all directions can be achieved by a convolution with an anisotropic, separable Gaussian kernel, which has the same computational complexity as isotropic Gaussian blurring. Finally, the patch is downscaled and rotated to the final patch of size p. Details are given in Algorithm 1.

Algorithm 1 PSPE Pyramid-Smoothing Patch Extraction

Require: I – Input image; L_{o_i,s_j} – Gaussian Scale-Space pyramid with initial image scale σ_0, octaves $0 < o_i < O$ and octaves' layers $0 \leq s_j < S$; A – Local affine feature; σ_D – required diff. scale, p – a patch size, m - a measurement region multiplier.

Ensure: P – Extracted patch.

Get feature scale $s = \sqrt{A}$
Get patch to extr. feature scale $\rho = \frac{2\,ms}{p}$
Compute Singular Value Decomposition $A = U\ D\ V^T$,
Set $l_1 = D_{1,1}/s,\ \ l_2 = D_{2,2}/s$
Differentiation blur in l_2 direction in I is $\sigma_{l_2} = \sigma_D\ \rho\ l_1$
if $\sigma_{l_2} < \sigma_0$ **then**
  $\hat{\sigma} = 0.5,\ \ \hat{d} = 1$
  $\hat{P}(\mathbf{x}) = I(U\ \text{diag}(l_1, l_2)\ \mathbf{x})$
else
  $o = \lfloor \log_2(\sigma_{l_2}/\sigma_0) \rfloor,\ \ v = \lfloor \log_{2^{1/S}}(2^{-o}\sigma_{l_2}/\sigma_0) \rfloor$
  $\hat{\sigma} = \sigma_0\ 2^{o+v/S},\ \ \hat{d} = 2^o$
  $\hat{P}(\mathbf{x}) = L_{o,s}(2^{-o}\ U\ \text{diag}(l_1, l_2)\ \mathbf{x})$
end if
Ensure correct diff. scale in both x and y directions
$\hat{\sigma}_x = l_2\ \hat{\sigma}/\hat{d},\ \ \hat{\sigma_y} = l_1\ \hat{\sigma}/\hat{d},\ \ \hat{\sigma}_D = \sigma_D\ \rho/\hat{d}$
$\hat{\sigma}_{dx} = \sqrt{\hat{\sigma}_D^2 - \hat{\sigma}_x^2},\ \ \hat{\sigma}_{dy} = \sqrt{\hat{\sigma}_D^2 - \hat{\sigma}_y^2}$
Blur $\hat{P}_B(\mathbf{x}) = \hat{P}(\mathbf{x}) * g(\mathbf{x}, \Sigma), \Sigma = \text{diag}(\hat{\sigma}_{dx}, \hat{\sigma}_{dy})$
where $g(\mathbf{x}, \Sigma)$ is 2D Gaussian filter
Sub-sample to patch: $P(\mathbf{x}) = \hat{P}_B(\rho\ V^T\ \mathbf{x})$

We propose a faster variant of the PSPE algorithm, PNBPE (Pyramid, No-Blur Patch Extraction), which ignores the anisotropic blurring step. This variant simply warps the selected pyramid layer to the patch without any intermediate steps. Unlike the PSPE, the pyramid layer is not selected according to the affine shape of the feature, but solely based on the feature scale. With this simplification, the PNBPE method gets several times faster than the former PSPE variant. Computation time of this method depends only on the SIFT patch size (usually $p = 41$). Details

Algorithm 2 PNBPE Pyramid, No-Blur Patch Extraction

Require: I – Input image; L_{o_i,s_j} – Gaussian Scale-Space pyramid with initial image scale σ_0, octaves $0 < o_i < O$ and octaves' layers $0 \leq s_j < S$; A – Local affine feature; σ_D – required diff. scale, p – a patch size, m - a measurement region multiplier.
Ensure: P – Extracted patch.

Get feature scale $s = \sqrt{A}$
Get patch to extr. feature scale $\rho = \frac{2\ ms}{p}$
Differentiation blur in I is $\sigma = \sigma_D\ \rho$
if $\sigma < \sigma_0$ **then**
  $P(\mathbf{x}) = I(\rho\ A\ \mathbf{x})$
else
  $o = \lfloor \log_2(\sigma/\sigma_0) \rfloor,\ \ v = \lfloor \log_{2^{1/S}}(2^{-o}\sigma/\sigma_0) \rfloor$
  $P(\mathbf{x}) = L_{o,s}(2^{-o}\ \rho\ A\ \mathbf{x})$
end if

of this variant are given in Algorithm 2.

We also test a method, called NBPE (No-Blur Patch Extraction), which simply warps the feature measurement region to the patch using the *original image*, not the pyramid. The speed of NBPE and PNBPE is equal.

With PSPE, PNBPE and NBPE it generally holds that the time needed for patch extraction is proportional to the number of the features and does not depend on feature size, as can be seen in Figure 3. The independence of patch extraction time on feature size is demonstrated in Figure 4.

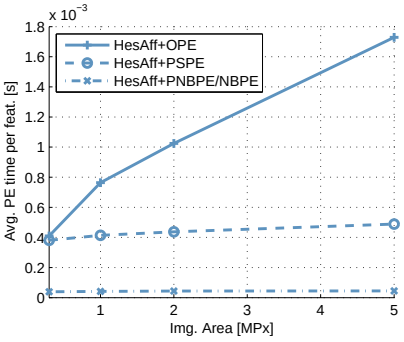


Figure 4: Average time needed for extraction of a single patch. The values are computed as an average over all features from 18 different images of a given resolution and is computed with HessianAffine detector.

3.3 Matching performance of patch extraction variants

In Figure 7, the comparisons of matching scores of different patch extraction methods for the Hessian-Affine and MSER detector are shown. The matching score is measured as defined by [15], with a difference that one-to-one correspondences are computed for descriptors only. In the original implementation of the matching score benchmark, match is deemed correct when it is a one-to-one match both based on descriptor distances and in ellipse overlaps. Though, this is not usable for DRs with multiple orientations as the ellipse overlap does not take into consideration the dominant orientations. We use matching score instead of [14] as we want to measure performance under various geometric transformations. Matching score has been computed with RootSIFT [1] descriptor normalisation.

Tests have been performed on datasets from [15] or their variants from [4] with more precise ground truth where available. The GRAF and WALL dataset test invariance to viewpoint change, BOAT and BARK to zoom and rotation and BIKES are used to test invariance to image blur. In order to show the invariance to scale changes we have measured average matching score using 32 images which have been resized to scales in $\langle 1, 0.2 \rangle$ (SYNTH. SCALE). Similarly, to simulate invariance to anisotropic deformations, we have generated datasets of the same images but scaled only in the y-axis direction (SYNTH. ANIS. SCALE).

It can be observed that PSPE method obtains in general the same performance as the original patch extraction method. The PNBPE and NBPE methods have similar performance when the features which need to be matched are relatively small, though they get worse performance on the Bark and Bikes dataset and with MSER detector which detects bigger features. This is additionally confirmed with the tests on synthetic images. On synthetic scale dataset, it can be seen that the PNBPE method has slightly better scale invariance as the NBPE method is ignoring Nyquist–Shannon sampling theorem and in case of bigger scale changes, the aliasing becomes an issue.

The reason why PNBPE and NBPE has got the same number of matches is that these methods detects more dominant orientations (1.9 for PNBPE and 2 for NBPE) as the extracted patch contains higher frequencies. But those orientations are less stable, thus these patch extraction variants have worse matching score.

4 Speeding up the SIFT

HesAff+OPE+SIFT	HesAff+NBPE+SIFT
Convolution (30%)	Interpolation (19%)
Interpolation (20%)	SIFT sampling (16%)
SIFT sampling (10%)	arctan2 (14%)
arctan2 (8%)	Gradient comp. (10%)

Table 2: The most time consuming functions (self-cost, percentage of the whole program runtime) by profiling Wall-1 [15] feature extraction.

As a significant bottleneck of the feature extraction process is the arctan2 function (see Table 2), we have investigated the precision and speed of existing implementations in *VLFeat* and *OpenCV* and proposed a new algorithm which outperforms these approximations in speed.

We have created a method which approximates the arctan2 function using look-up table (LUT). This method divides the interval $\langle 0, 2\pi \rangle$ into octants, and is using a LUT of 256 bins accordingly for each octant as $\arctan(x/y) = \pi/2 - \arctan(y/x)$, if $x > 0$ and similar rule is for $x < 0$.

The comparison of different methods in single precision floating point numbers is given in Table 3 where the error

Impl.	RMS Err. [rad$\times 10^{-3}$]	Max Err. [rad$\times 10^{-3}$]	Avg. time [ns]
ARCTAN2	0	0	139.1
VLFEAT	4.278	6.136	95.3
OPENCV	0.073	0.167	99.5
LUT-256	1.815	3.922	89.9

Table 3: Speed of different arctan2 implementations and approximations in single floating point precision. Values are computed over 2×10^6 measurements and the error is compared against the arctan2 , standard reference.

is compared against the standard implementation[4]. It can be seen that our LUT-256 method outperform the investigated methods in speed and has smaller error than the method used in VLFeat library. The error does not have any influence on descriptor performance. However, as the speed of LUT approximation depends mostly on memory access speed, for processors with smaller CPU cache it may be needed to reduce the number of bins, partially scarifying the precision.

Method	DomOri [μs] (Speed-up)	SIFT [μs] (Speed-Up)
Standard	110.35	159.26
LUT-256	47.12 (2.3)	95.06 (1.7)
SSE-OpenCV	38.75 (2.8)	73.38 (2.2)

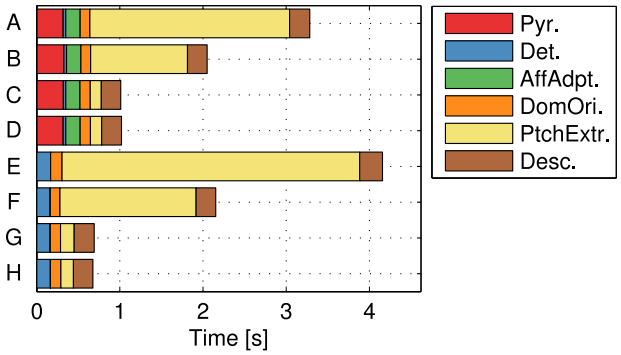
Table 4: The average speed-up per a single MSER feature using the arctan2 approximation and SSE instructions in different patch extraction stages.

However the OPENCV approximation is implemented using SSE instructions, and in addition it uses SSE for gradient magnitude computation where the SSE instruction for the square root (SQRTPS) is already an approximation with rel.err. $\leq 1.5 \times 10^{-12}$ [6] which is more than sufficient for the feature extraction purposes.

The overall speed-up of the feature extraction stages in comparison to the original implementation is shown in Table 4. It can be seen that even speeding up the arctan2 function with a look-up table can bring a significant improvements in the processing time. The advantage of the LUT-256 is that it has tunable precision by varying the number of bins which can be chosen in such a way that it would fit to the CPU cache of the target architecture. However, in the following experiments we use the SSE-OpenCV variant.

The processing time of Hessian-Affine and MSER detectors using the proposed improvements are shown in Figure 5. It can be seen that with the PSPE or NBPE algorithm together with the SSE-OpenCV (referred as SIFT+), the feature extraction process takes around half the time of the original implementation. In Table 5 we show the average processing time per an image from the Mikolajczyk's dataset [15] without normalisation to a constant number of features, i.e. the expected time needed to extract features from a single image. From this table it is clear that for example using the improved MSER+NBPE+SIFT+ for feature extraction is similarly time consuming as using SURF algorithm.

[4]Defined by IEEE Std 1003.1, particularly used GNU C Library 2.15 implementation



Variant	Detector	Patch Extr.	Descriptor
A	HesAff	OPE	SIFT+
B	HesAff	PSPE	SIFT+
C	HesAff	PNBPE	SIFT+
D	HesAff	NBPE	SIFT+
E	MSER	OPE	SIFT+
F	MSER	PSPE	SIFT+
G	MSER	PNBPE	SIFT+
H	MSER	NBPE	SIFT+

Figure 5: Processing time of particular feature extraction stages using the proposed improvements. In all feature extractors here, the SSE-OpenCV method is used for computing arctan2 . The values in the graph are measured in the same way as in Figure 2.

Method	Avg. NFeats	Avg. Time
DoG+SIFT	1719.23	0.90
FHes+SURF	2192.33	0.72
HesAff+OPE+SIFT	3181.81	5.38
HesAff+PSPE+SIFT+	3201.98	3.56
HesAff+PNBPE+SIFT+	3714.40	1.78
HesAff+NBPE+SIFT+	3787.60	1.80
MSER+OPE+SIFT	1028.02	2.83
MSER+PSPE+SIFT+	1035.04	2.15
MSER+PNBPE+SIFT+	1266.79	0.69
MSER+NBPE+SIFT+	1348.71	0.69

Table 5: Average processing time and number of features per an image (without image IO) of commonly used extractors and feature extractors with the proposed speed improvements on all images from Mikolajczyk's dataset [15]. The values are computed in the same way as in Table 1.

5 Matching features in multiple-orientation context

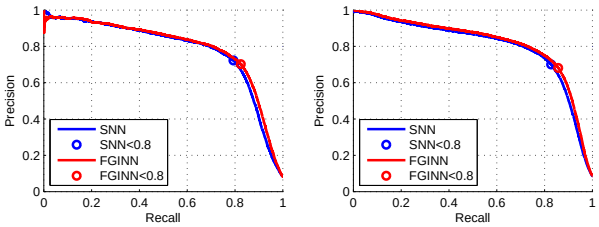
The output of a detection algorithm on input image I is a set of distinguished regions (DR). Afterwards, dominant orientations $\mathcal{A}$ for each region are detected in order to obtain rotation invariance, which creates several local affine features for each DR. Usually, the number of dominant orientations is limited to $1 < |\mathcal{A}_u^I| \leq 4$ and for each dominant orientation, one descriptor is extracted.

In image matching task, the fact that a single DR generates more descriptors is usually ignored and it simply matches all descriptors from a reference image to the matched image. This has several consequences, e.g. for the SNN ratio (SNNr) criterion. This criterion is used to filter correspondences $\mathcal{C}$ of the reference image descriptors to the matched image descriptors and generates a set of tentative correspondences $\mathrm{TC} \subset \mathrm{C}$. It computes the distance ratio

of the first closest to the second closest reference image descriptor and when this ratio is higher than 0.8, the correspondence is rejected (i.e. that the correspondence is not distinguishable enough). On the other hand, in case of symmetric DRs, the second nearest neighbour can be located on the same DR in the matched image as the first nearest neighbour. This issue was tackled by [16] where the SNN criterion has been revisited. The authors added a new condition that the SNN ratio is not computed with the second closest descriptor but with the *First Geometrically Inconsistent Nearest Neighbour* (FGI-NN). Two descriptors are geometrically inconsistent when their centres of gravity are farther than a given threshold (in our case set to $10px$).

In [16], it was used in context of synthesised views, but we have observed that it has some influence on simple two view matching as well. In our tests we have matched various 85 image pairs (image pairs introduced by [7], [15] and 65 pairs selected from clusters generated by [3]) and we estimated a homography between them using LO-RANSAC algorithm [7]. Descriptors are extracted using the Hessian-Affine+OPE+SIFT. Inliers $\mathrm{I} \subset \mathrm{TC}$ are TC with symmetric reprojection error [5] smaller than $4px$, and similarly, valid correspondences $\mathcal{VC} \subset \mathrm{C}$ are all geometrically correct correspondences. We compute precision and recall as:

$$\text{precision} = \frac{|\mathcal{I}|}{|\mathcal{TC}|} \quad \text{recall} = \frac{|\mathcal{I}|}{|\mathcal{VC}|} \tag{2}$$



(a) MSER (b) HessianAffine

Figure 6: Precision and recall of SNN and FGINN for MSER and HessianAffine with QRT-SIFT. Results for 85 image pairs.

Method	AP	Prec [%]	Rec [%]	$\lvert\mathcal{TC}\rvert$	$\lvert\mathcal{I}\rvert$
MSER SNN	78.42	72.25	79.38	9408	6797
MSER FGI-NN	79.60	70.22	82.43	10039	6940
HesAff SNN	80.12	70.04	82.74	29817	20883
HesAff FGI-NN	81.63	68.12	85.73	31919	21397

Table 6: Precision and recall for $\mathrm{SNNr} < 0.8$ with average precision (area under the PR curve) for methods for generating tentative correspondences and two selected detectors which use SQRT-SIFT as descriptor. Results for 85 image pairs

We have measured that using the FGI-NN method improves the performance of the system even in the case of two-view matching without view synthesis. The particular values of precision and recall for $\mathrm{SNNr} < 0.8$ with average precision are shown in Table 6. Though it slightly lowers the precision, it increases the recall and is able to obtain more inliers which are important for the accuracy. The FGI-NN criterion also increases the average precision means that FGI-NN is a better classifier of tentative correspondences than SNN without being dependent on the particular SNNr threshold. The correspondences missed by the SNN method are usually caused by symmetric features where the SNN may be on the same DR but with a different orientation. The precision-recall curve, computed varying the SNNr threshold, is shown in Figure 6.

# dom.orientations $\lvert\mathcal{A}\rvert$	1	2	3	4
% of DRs	44.97	43.09	10.72	1.22

Table 7: Percentage of detected distinguished regions with a certain number of dominant orientations. Values are computed out of 1.8×10^5 DRs.

In the next experiment we investigate how dominant orientations of the DRs are matched across the images. At first, in Table 7, we show the distribution of number of dominant orientations per a DR detected in the reference images. On average, each DR is assigned 1.8 dominant orientations.

	Dominant orientations $\lvert\mathcal{A}\rvert$			
Matched DRs	Matched $\lvert\mathcal{A}_\mathcal{D}\rvert$			
	1	2	3	4
1	62.49%	30.70%	4.63%	0.38%
2	0.00%	1.37%	0.39%	0.03%

Table 8: Distribution of the DRs in tentative correspondences $\mathcal{TC}$ according to their number of matched Dominant orientations (column) and number of matched *unique* DRs from the tested image (row). E.g. a DR which is in the second row and third column has three dominant orientations in $\mathcal{TC}$ where two are matched to the same DR in the tested image. $\mathcal{TC}$ are generated using the FGI-NN criterion.

There is clearly a lot of DRs which have more than one descriptor. But what happens when correspondences are generated? In Table 8 we show the distribution of the DRs in $\mathcal{TC}$ according to the number of their dominant orientations in $\mathcal{TC}$ and number of matched DRs in the second image. It can be seen that for many DRs with multiple dominant orientations, only some of them passed the FGI-NN criterion.

More than 30% of the DRs have 2 dominant orientations where both of them are matched against a single DR in the tested image (row 1 column 2). Those 30% of DRs are actually generating more than 42% of correspondences which are passed to RANSAC algorithm. This also means that in our dataset, more than 22% of the correspondences are duplicates.

This can be exploited by improving the speed of RANSAC algorithm by passing less tentative correspondences as sampling two correspondences of same image regions leads to a degenerate solution. This issue is handled using "duplicate filtering" procedure in [17], [16]. However, if the double correspondence is found as inlier, it may be counted twice as the fact that two dominant orientation have been matched may bear a prior of the correspondence quality. Though, we have not investigated those issues.

From Table 8 it can be observed that most of the incoherent correspondences, i.e. ref. image DRs matched to different DRs in the matched image, does not pass to the list of $\mathcal{TC}$, thus the Lowe's (FGI)SNN ratio works well for the multiple orientation matching by itself.

6 Conclusions

It has been shown shown that a careful implementation of existing affine invariant feature detectors has a speed comparable to existing similarity covariant detectors. Furthermore, using a simplified patch extraction method the speed of the affine feature extraction becomes comparable to their approximations, such as SURF. The speed leads to slight decrease in the scale invariance of the extracted patches.

The process of patch description is made faster by a simple approximation of the arctan2 function. We have created a simple approximations of arctan2 which uses a look-up table and the terms of speed outperforms the approximations used in different computer vision libraries.

We have investigated the Lowe's second nearest neighbour criterion in the context of multiple orientation matches. We confirmed that using the first geometrically inconsistent nearest neighbour increases the number of inliers as it allows to match symmetric features. Furthermore, we have investigated the way how the second nearest neighbour works in case of multiple orientations and proposed some improvements which can speed up RANSAC.

Acknowledgement

The authors were supported by EC project FP7-ICT-270138 DARWIN by the Technology Agency of the Czech Republic program TE01020415 (V3C – Visual Computing Competence Center).

References

[1] R. Arandjelović and A. Zisserman. Three things everyone should know to improve object retrieval. In *IEEE Computer Vision and Pattern Recognition*, 2012.

[2] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[3] Ondrej Chum and Jirí Matas. Large-scale discovery of spatially related images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(2):371–377, 2010.

[4] Kai Cordes, Bodo Rosenhahn, and Jörn Ostermann. Increasing the accuracy of feature evaluation benchmarks using differential evolution. In *IEEE Symposium on Differential Evolution*, pages 1–8. IEEE, 2011.

[5] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*, volume 2. Cambridge University Press, 2000.

[6] Intel Corporation. *Intel® 64 and IA-32 Architectures Software Developer's Manual*, volume 2. June 2013.

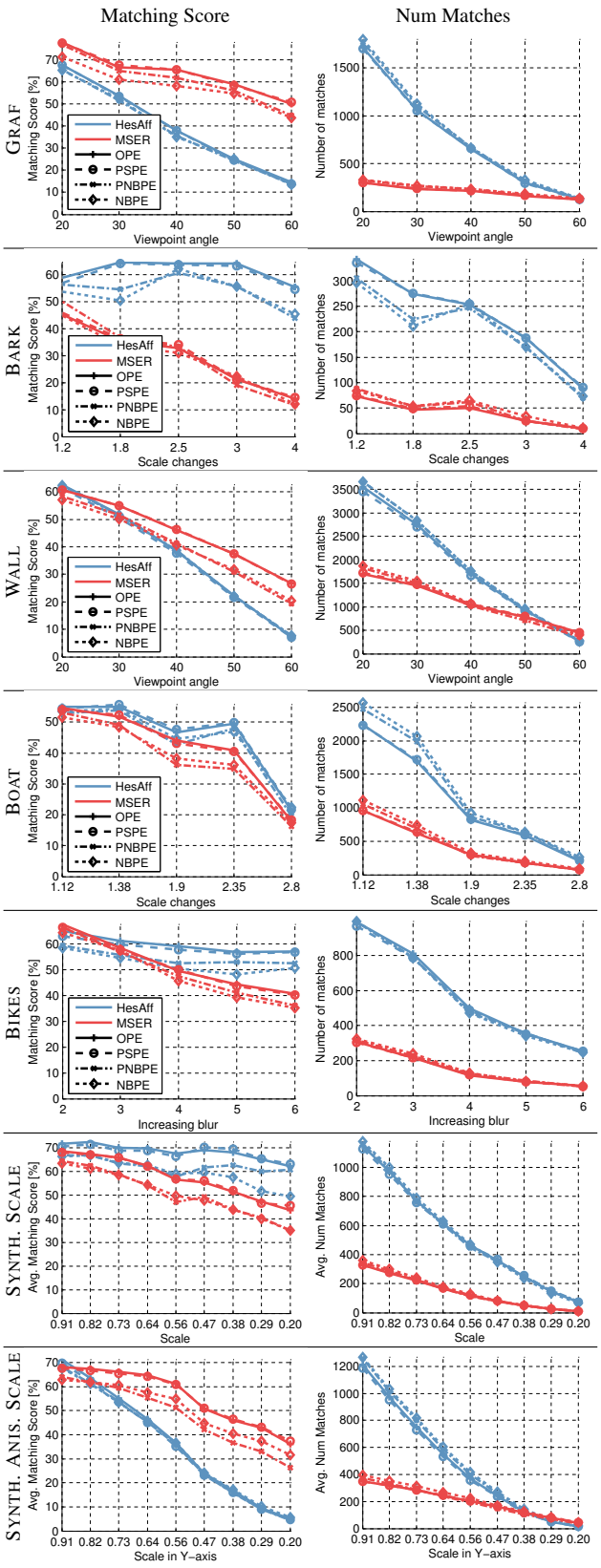


Figure 7: Comparison of the performance of different patch extraction algorithms using Mikolajczyk's matching score protocol [15]. Measurements on synthetic datasets are computed as an average over generated image pairs from 32 various images. Line colour distinguishes different detectors and line style signifies the patch extraction algorithm.

[7] Karel Lebeda, Jiri Matas, and Ondrej Chum. Fixing the locally optimized RANSAC. In *British Machine Vision Conference*, 2012.

[8] Sung-Won Lee, Ki-Seok Kwon, and In-Cheol Park. Pipelined cartesian-to-polar coordinate conversion based on srt division. *Circuits and Systems II: Express Briefs, IEEE Transactions on*, 54(8):680–684, 2007.

[9] Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart. BRISK: Binary robust invariant scalable keypoints. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2548–2555. IEEE, 2011.

[10] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[11] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *British Machine Vision Conference*, pages 384–393, 2002.

[12] Herbert A Medina. A sequence of polynomials for approximating arctangent. *The American Mathematical Monthly*, 113(2):156–161, 2006.

[13] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *Proceedings of the 7th European Conference on Computer Vision*, pages 128–142, 2002.

[14] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(10):1615–1630, 2005.

[15] Krystian Mikolajczyk, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1/2):43–72, 2005.

[16] Dmytro Mishkin, Michal Perdoch, and Jiri Matas. Two-view matching with view synthesis revisited. In *Proceedings of the 28th Conference on Image and Vision Computing New Zealand*, 2013.

[17] Jean-Michel Morel and Guoshen Yu. Asift: A new framework for fully affine invariant image comparison. *SIAM Journal on Imaging Sciences*, 2(2):438–469, 2009.

[18] M. Perďoch, O. Chum, and J. Matas. Efficient representation of local geometry for large scale object retrieval. In *IEEE Computer Vision and Pattern Recognition*, pages 9–16. IEEE, 2009.

[19] Sreeraman Rajan, Sichun Wang, Robert Inkol, and Alain Joyal. Efficient approximations for the arctangent function. *Signal Processing Magazine, IEEE*, 23(3):108–111, 2006.

[20] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, volume 1, pages 430–443, May 2006.

[21] Andrea Vedaldi and Brian Fulkerson. VLFeat: an open and portable library of computer vision algorithms. In *Proceedings of the international conference on Multimedia*, pages 1469–1472, 2010.

Computer Vision Winter Workshop 2014, Zuzana Kúkelová and Jan Heller (eds.)
Křtiny, Czech Republic, February 3–5
Czech Society for Cybernetics and Informatics

Print Localization on Transparent Pharmaceutical Capsules

Andraž Mehle[1], Marko Bukovec[1], Franjo Pernuš[1,2], Boštjan Likar[1,2], and Dejan Tomaževič[1,2]

[1]Sensum, Computer Vision Systems
Ljubljana, Slovenia
[2]Faculty of Electrical Engineering,
University of Ljubljana, Slovenia
andraz.mehle@sensum.eu

Abstract *This paper presents a novel method for real-time print localization on transparent pharmaceutical capsules which is a crucial step for automated visual inspection. The method is based on print segmentation and template matching technique. A print appearance used for template matching is constructed from capsule images without defects during the training phase. Print localization during inspection phase is achieved by combination of phase correlation between template and sample, and additional criterion to compensate for print overlaps and ambiguities due to capsule's transparency. The method was evaluated in terms of robustness, accuracy and speed on a large image database of transparent capsules with radial print. Results were compared to the method for standard opaque capsules. The results indicate that our method shows improved robustness and accuracy. Moreover, computational time of less than 10 milliseconds allows real time visual inspection of pharmaceutical capsules.*

1 Introduction

Nowadays, pharmaceutical industry produces vast amount of different pharmaceutical tablets and capsules. To avoid dangerous drug mix-ups, every type of product has to be quickly, easily and unambiguously identified by doctors and pharmacists as well as by end consumers. Different products should be uniquely characterized by their size, shape, color, texture, prints, etc. [1]. Demands are enforced by national regulators in each country such as a regulation code 21CFR206 [7] issued by the Food and Drug Administration in USA. Besides unambiguous identification, high quality of visual appearance of pharmaceutical products is required by pharmaceutical companies since defected products cause doubts and lower level of trust among consumers.

The most common method for visual quality control of pharmaceutical capsules is manual inspection by various methods. The disadvantage of such methods is that the overall quality of the whole batch of capsules is estimated by inspecting only certain sample of capsules. The required quality of single product can thus not be guaranteed. Furthermore some countries, e.g. Japan, have regulations that enforce every single product to be visually inspected, either manually or automatically. The capsules may be inspected before or after they have been filled with active substance, however the latter is more common. Since manual visual inspection is slow, unreliable, tedious, costly and even harmful to the operators, fully automated visual inspection of every single product in a batch is emerging.

Automated visual inspection of pharmaceutical capsules [10, 12] is very challenging, because capsules come in different sizes, colors and prints and may have various visual defects. Sophisticated high-tech machine vision system with fast mechanical capsule manipulation, proper illumination, fast image acquisition, image analysis, capsule classification and sorting mechanism is thus required [3]. Speed requirements of such systems are from 20 up to 100 products per second. This calls for fast and efficient image processing algorithms with low computational complexity but high reliability and robustness [8, 14].

Beside other visual characteristics, print plays important role in identification of pharmaceutical capsules, because it provides fast identification of manufacturer and active substance [17]. Print can include company logo or name, commercial name of the product, chemical name or even information about dosage. Print on pharmaceutical capsules consists of two parts, one on each half of a capsule. Print can be oriented along the main axis of the capsule, i.e. axial print, or perpendicular to the main axis, i.e. radial print. Combination of axial and radial print is also possible, wherein each part of a capsule has different print orientation [15]. Print legibility is the most important criterion for identification. Despite optimal choice of print size, shape and content, print defects made during printing process or transportation, may affect print legibility. The defects include partly or entirely missing print, multiple prints, blurred print, smudged print, ink spots, color and size variations of the print, etc.

Capsules with transparent shell are commonly used as the dose-holding system for breath-actuated dry powder inhalers [6], devices for delivery of drugs to the lung. Many disorders affecting the lungs such as asthma are treated by inhaling drugs to increase the airflow or reduce inflammation. Transparency of capsule's shell allows the user to see the active substance (powder) and to easily check if the whole dose was properly inhaled. Capsules may be completely clear (natural gelatin color) or colored and may include prints.

There are various automated visual inspection systems for pharmaceutical capsules on the market from companies such as Ackley, Eisai, Ikegami, Mutual, Proditec, Seidenader, Sensum and Viswill but only a few research articles on this subject have been published so far. Karloff et al. [10] and Islam et al. [8, 9] designed a low cost capsule inspection system that can be integrated into an existing mechanical capsule sorters. The system is capable of inspecting 20 capsules per second and is able to inspect uni-color, bi-color as well as transparent capsules. However, their system can only detect larger defects such as cracks, dents, double caps and improper length and can only inspect capsules without print. Visual inspection of printed capsules is even more challenging task. The main problem is cylindrical shape, which allows capsules to freely rotate around their main axis. Thus, the spatial location of a print may vary from image to image, where portions of a print may be hidden. Moreover, due to the cylindrical shape of capsules, spatial distortions occur when 3D capsule surface is projected onto 2D image plane. Distortions are most prominent at the border of a capsule.

Print localization is a crucial element of automated visual inspection of printed capsules, without which further analysis and classification is not possible. The area of capsule's surface containing print is inspected separately from the rest of a capsule. Successful print localization enables inspection of print quality as well as inspection of the rest of a capsule. If the print is not properly localized, portions of print may be recognized as defects. Špiclin et al. [14] proposed a template matching technique for localization of print on opaque capsules. They eliminated spatial distortions by transforming a capsule image into cylindrical coordinate system and used template matching technique [16] to localize print. A print appearance template is constructed from capsule images without defects during the training phase. Because of high speed requirements, registration method incorporates simple two dimensional translation as transformation model between template and sample image. In general, transformation is not linear due to different spatial deformations that can occur during printing process or additional distortions caused by imperfect image acquisition.

Transparent capsules bring additional challenges to all segments of automated visual inspection including print localization. Transparency causes that both the front surface, i.e. capsule surface faced towards the camera (foreground), and the back surface, i.e. capsule surface faced to the opposite direction (background), are captured (Fig. 1). When a print is located on the background, it appears mirrored and has lower contrast. Moreover, portions of the print can be concurrently visible on the foreground and on the background and may overlap. Furthermore, spatial deformations due to printing process and distortions due to imperfect image acquisition are emphasized. Axial capsule movement during printing process causes radial line to manifest as helix (Fig. 1, left), while slightly tilted capsule causes radial line to be seen as ellipse (Fig. 1, right).

Due to problems mentioned above, the template matching method described by Špiclin et al. [14] does not achieve adequate results. In this paper we propose a method for real-time localization of print on transparent capsules that is capable of matching both foreground and background print simultaneously and is robust to spatial deformations and distortions. We validated our method in terms of robustness, accuracy and speed on a large image database of transparent capsules with radial print.

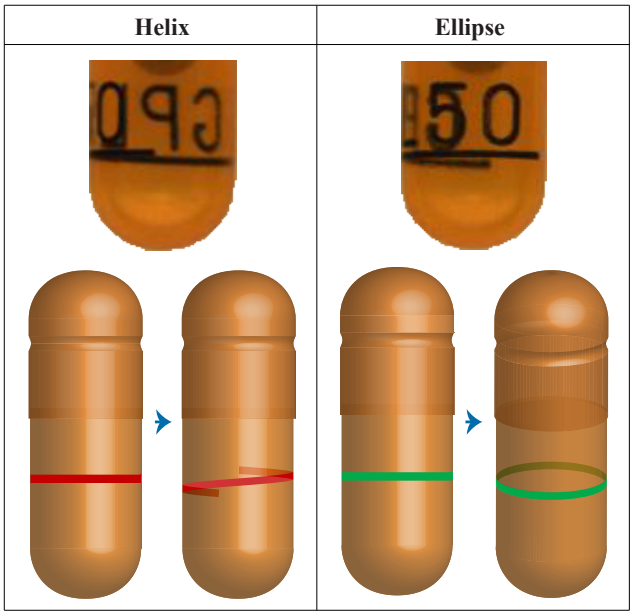


Figure 1: Spatial deformations of print on transparent capsules: altered camera viewing angle causes radial line to manifest as elipse (left), rotated radial line manifests as helix.

2 Materials and Methods

In this section the method for print localization on transparent pharmaceutical capsules is described. The method matches a print template to foreground and background print on sample image where foreground and background print may overlap significantly (Fig. 2). Our method is based on segmented print images.

First the entire capsule is segmented by border tracking algorithm [13] (Fig. 2, left). Then the print is segmented from color image by max shift segmentation algorithm [5] (Fig. 2, right) which is based on mean shift clustering [4]. The task of max shift segmentation algorithm is to separate the modes of the probability distribution in multidimensional histogram, i.e. to separate clusters in feature space of a histogram that represent different regions in an image. Color values of image pixels are mapped into a 3D histogram, a feature space with multivariate and generally multi-modal probability distribution. The regions in histogram with the highest density correspond to clusters centered on the modes of the underlying probability distribution. Max shift algorithm uses a cube search kernel to find the clusters in the histogram. In each iteration the kernel is shifted in the direction of maximum gradient inside the kernel. The center of a cluster is obtained by convergence of the kernel from the initial location. Once the center is obtained, the corresponding feature points that belong to this cluster are determined by applying the max shift algorithm to the neighboring points of the cluster center. The procedure is

repeated until all the feature points are labeled.

Main axis of the capsule is estimated from the capsule's shape obtained from capsule segmentation and is used for transformation into cylindrical coordinate system as described in [14] , where only capsule region where print presence is expected is transformed (Fig. 3a).

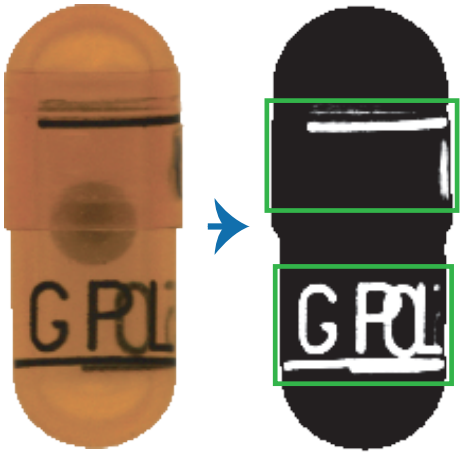


Figure 2: Segmented color image (left) and corresponding print segmentation (right) with print regions (rectangles).

Špiclin et al. [14] performed transformation of print region into cylindrical coordinate system only for visible (foreground) print, i.e. transformation was performed only on the interval of [0°, 180°]. In contrast, for transparent capsules both foreground and background print are visible but inseparable, thus the extended transformation (from 0°to 360°) is performed (Fig. 3a). Transformation of print on interval [0°, 180°] can be interpreted as foreground print with visible background, while transformation on interval [180°, 360°] represents background print occluded by foreground. The goal of the method is to match the entire 360°template to foreground and background print.

The template is matched to the input image by calculating phase correlation [11] between template $h(x,y)$ and input image $f(x,y)$. Phase correlation method is based on Fourier Shift Theorem [2] and can be efficiently calculated in frequency (Fourier) domain. It computes normalized cross-power spectrum $S(\xi,\eta)$ between images:

$$S(\xi,\eta) = \frac{F(\xi,\eta)H^*(\xi,\eta)}{|F(\xi,\eta)H^*(\xi,\eta)|}, \tag{1}$$

$$PC(u,v) = \mathcal{F}^{-1}\{S(\xi,\eta)\}, \tag{2}$$

where $H(\xi,\eta)$ and $F(\xi,\eta)$ are discrete 2D Fourier transforms of template and input image respectively and H^* is complex conjugate of H. $PC(u,v)$ denotes inverse Fourier transform of $S(\xi,\eta)$ which is ideally (according to Fourier Shift Theorem) a Dirac delta function $\delta(x+u_0, y+v_0)$ centered at (u_0,v_0), where u_0 and v_0 represent shift between images. The problem of finding shift (u_0,v_0) thus translates to the problem of locating delta peak in PC:

$$(u_0,v_0) = \underset{(u,v)}{\operatorname{argmax}}(PC(u,v)). \tag{3}$$

Phase correlation shows strong robustness against the narrow band noise and non-uniform illumination changes [11, 18].

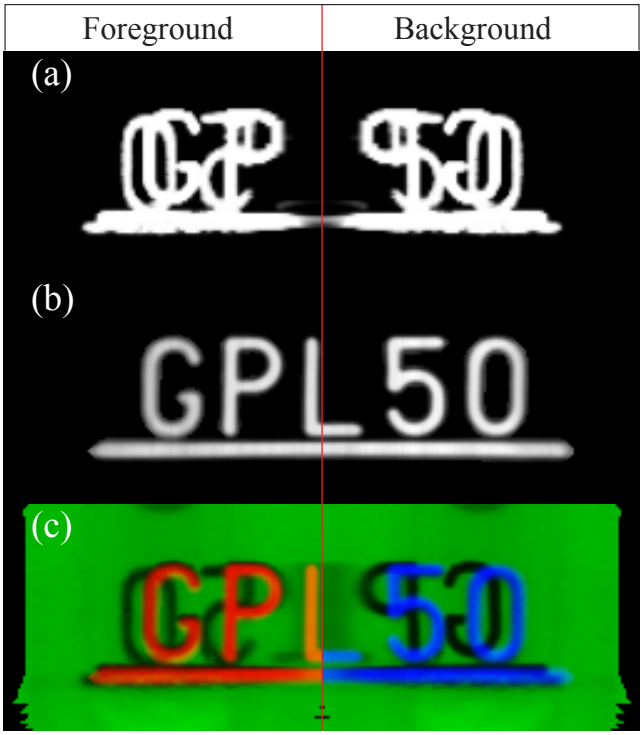


Figure 3: (a) 360°input image, (b) template image, (c) template image matched to input image (red - foreground, blue - background).

Because of foreground and background print overlap, and due to similar appearance of individual symbols or characters, phase correlation often results into more than one distinct peak, where the most prominent one does not necessarily represents the optimal alignment (Fig. 4). Therefore, additional criterion is needed to isolate the optimal peak from N most distinct peaks $(u_i,v_i; i=1\ldots N)$.

Let us define Foreground-Background Overlap

$$FBO_{u_i,v_i} = FO_{u_i,v_i} + BO_{u_i,v_i}, \tag{4}$$

that measures overlap of print area between template and input image separately for foreground (FO) and background (BO) at given shift (u_i,v_i). Each of N most distinct peaks $(u_i,v_i; i=1\ldots N)$ can further be evaluated as

$$C_{u_i,v_i} = PC_{u_i,v_i}\, FBO_{u_i,v_i}, \tag{5}$$

where PC_{u_i,v_i} represents the value of PC (2) at given peak location (u_i,v_i). The optimal alignment, i.e. the shift between the template and input image, is found as

$$(u_0,v_0) = \underset{i}{\operatorname{argmax}}(C_{u_i,v_i}). \tag{6}$$

The goal of FO is to measure the overlap between the template and input image only on foreground (from 0°to 180°). It is defined as:

$$FO_{u_i,v_i} = \left[\frac{\sum_{x,y} f_{fg}\, h_{fg}}{\sum_{x,y} h_{fg}}\right] \sum_{x,y} f_{fg}\, h_{fg} \tag{7}$$

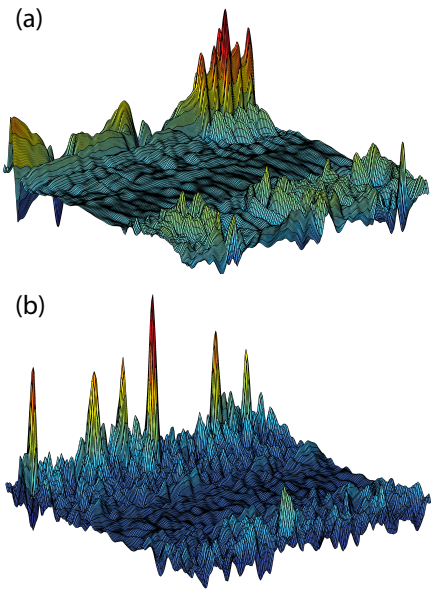


Figure 4: Examples of PC correlation images with several distinct peaks: (a) cap print correlation image, (b) body print correlation image.

Input image f_{fg} represents only foreground half of 360°input image (Fig. 5c):

$$f_{fg} = f(x, y)\, m(x, y), \tag{8}$$

where mask $m(x, y)$ (Fig. 5b) is equal to one at the foreground (interval from 0° to 180°) and zero elsewhere. Similarly, template h_{fg} represents only foreground half of the template at given shift (u_i, v_i) (Fig. 5e):

$$h_{fg} = h(x + u_i, y + v_i)\, m(x, y). \tag{9}$$

The first term of FO (7) represents the overlap factor, i.e. fraction of template h_{fg} overlapped with input f_{fg}. The overlap factor is equal to one when the entire foreground template h_{fg} is overlapped with f_{fg} and zero when they are completely mismatched. The second term in expression (7) stands for size of overlap region. Therefore measure FO has the highest value when the entire foreground template h_{fg} is overlapped (first term of (7)) and it overlaps as much input f_{fg} as possible (second term of (7)). When background print dominates on f_{fg} the FO might be highest at shifts where foreground template overlaps with background print, especially if the print appearance is very symmetrical. Thus an additional complementary measure of background overlap (BO) is needed.

Similarly to FO the BO is defined as:

$$BO_{u_i,v_i} = \left[\frac{\sum_{x,y} f_{bg}\, h_{bg}}{\sum_{x,y} f_{bg}}\right] \sum_{x,y} f_{bg}\, h_{bg} \tag{10}$$

New input images f_{bg} and h_{bg} are defined by erasing the overlapping print on f_{fg} and h respectively (Fig. 5d and 5f):

$$f_{bg} = f_{fg}\,(1 - h_{fg}), \tag{11}$$

$$h_{bg} = h_F\,(1 - h_{fg})\, m, \tag{12}$$

where h_F is vertically flipped background part of the template. At optimal shift (u_0, v_0) f_{bg} represents only the background print since the foreground print has been erased (11). Similarly from h_F, i.e. the part of the template expected on the background, the foreground part of the template (h_{fg}) has been erased (12), because the background print is always occluded with foreground print. While FO measures an overlap between foreground print f_{fg} and foreground template h_{fg}, BO measures an overlap between the rest of visible print (f_{bg}) and expected template on the background h_{bg}.

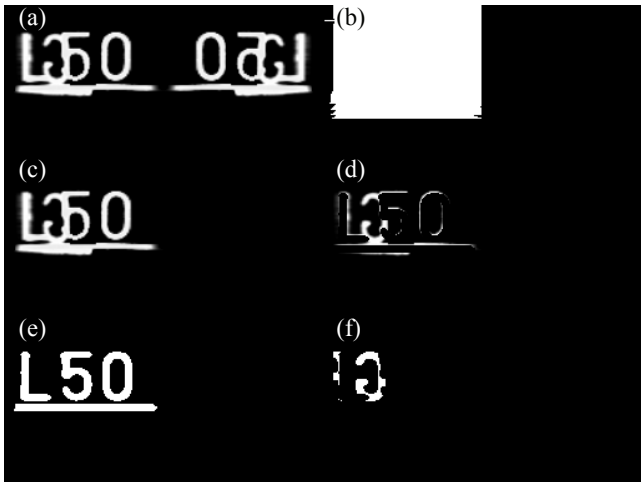


Figure 5: Input images for calculation of FO and BO: (a) 360°input image $f(x, y)$, (b) mask image $m(x, y)$, (c) foreground print f_{fg}, (d) background print f_{bg}, (e) foreground template h_{fg}, (f) background template h_{bg}.

3 Experiments and Results

Performance of the proposed method was evaluated in terms of robustness, accuracy and speed on a database of 516 images of transparent orange capsules with radial print on cap and body. The cap print included radial line with company logo while the body print included radial line and some text. The template and input image size was 256 x 128 pixels. Gold standard of print locations was obtained by manually determining three pairs of corresponding points between template and each input image. The localization error after the matching was defined as RMS of corresponding point distances. The implementation of the method was done in C++ and executed on a 3.4 GHz Intel Core i7 3770 platform. Speed was measured by the mean execution time to assess the feasibility of the method for real-time visual inspection of pharmaceutical capsules. The performance was compared to the print localization method for opaque capsules proposed by Špiclin et al. [14] where template matching technique based on normalized cross-correlation (NCC) was used. The spatial deformations of the print in cylindrical coordinate system can be as large as 5 pixels thus the localization was considered successful if the error was below 5 pixels. The accuracy was defined as mean error of all successful print localizations. The performance of the two methods is presented in Table 1. Additionally,

	Cap print		Body print	
Method	NCC	PC FBO	NCC	PC FBO
Robustness (%)	91.8	99.1	98.6	100
Accuracy (pixel)	1.9	1.4	1.8	1.1
Speed (ms)	4	7	4	7

Table 1: Performance of the proposed method (PC FBO) and method for opaque capsules (NCC) [14] in terms of robustness (percentage of successful localizations), accuracy (mean error of successful localizations in pixels), and speed (mean computation time in milliseconds).

Fig. 6 shows cumulative fraction of matched samples with respect to print localization error. Fig. 7 shows an example of failure of the print localization method for opaque capsules and successful print localization of the same sample with the proposed method.

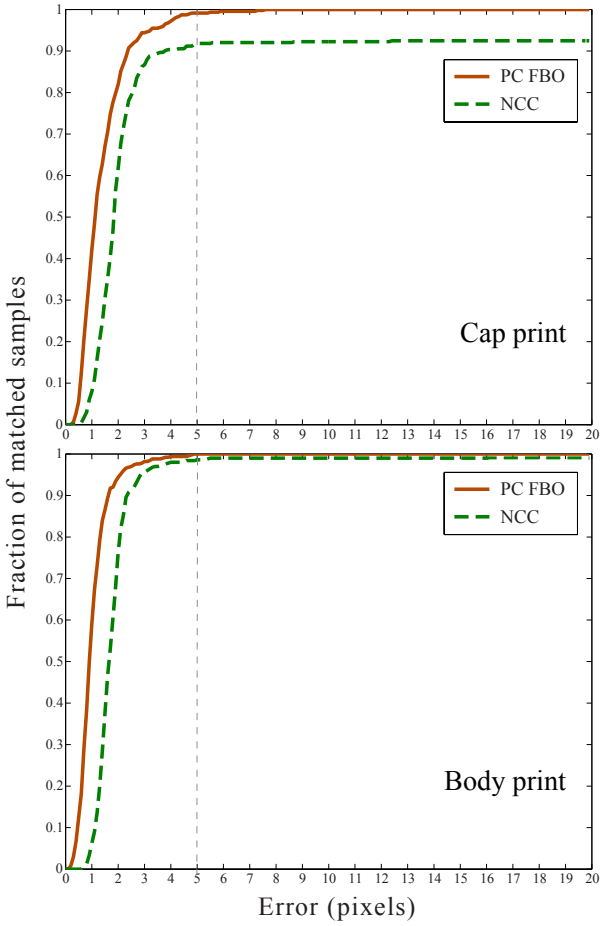


Figure 6: Cumulative fraction of matched samples with respect to print localization error for cap print (top) and body print (bottom). Our method (PC FBO) is compared to the method for opaque capsules (NCC) [14]. Localization with error less than 5 pixels was considered successful.

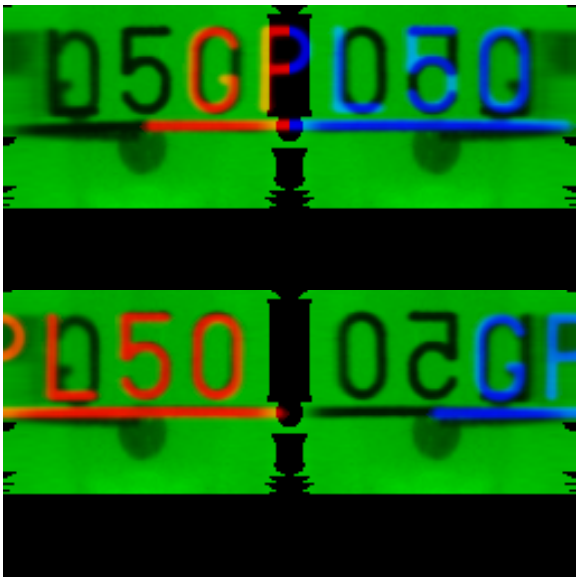


Figure 7: An example of unsuccessful print localization with NCC (top) and successful print localization with PC FBO (bottom). The print template is colored red on the foregorund (from 0° to 180°) and blue on the background (from 180° to 360°). Similarity between characters and overlapping of foreground and background print caused localization with NCC to fail.

4 Discussion and Conclusion

Successful print localization is a crucial element of visual inspection of pharmaceutical capsules with print. It allows proper inspection of print validity as well as detection of defects on the rest of the capsule's surface. A novel method for print localization on transparent capsules was proposed. The method was evaluated on real images and showed sufficient performance for defect detection and print quality inspection. The method shows high robustness to illumination changes, small spatial deformations of the print, and overlapping of foreground and background print. The method was compared to the print localization method used for standard opaque capsules where only foreground print is visible.

The success rate of our method was 99.1 % for the cap print and 100 % for the body print while the standard method with NCC achieved 91.8 % and 98.6 % success rate respectively. Our method shows great improvement of cap print localization. Extremely symmetrical appearance and small size of cap print made its localization much more difficult than that of the body print. Furthermore the standard method often resulted in completely false localization where the foreground template was matched to the background print or vice versa. The maximum error of our method was 7.6 pixels which is only a few pixels above the threshold of successful localization. Our method is computationally almost two times more demanding than the standard method but shows highly improved robustness and accuracy. Furthermore the execution time of 7 milliseconds for one image is sufficient for real time visual inspection of pharmaceutical capsules.

The overall measure C_{u_i,v_i} (5) is only as precise as PC because FBO only selects the optimal peak among N peaks

of PC. Furthermore if N is too low, none of the peaks might represent optimal shift. To achieve better precision and robustness we can calculate the measure FBO at all possible shifts $(u_i = 0 \dots W-1, v_i = 0 \dots H-1)$ where W is the width and H is the height of the input image. Calculating the entire FBO in time domain is computationally very expensive but it turns out that it can be efficiently calculated in Fourier domain in real time.

Because of high speed requirements the assumed transformation between template and input image was simple translation. However spatial deformations of the print caused the localization error to be as large as 5 pixels on some parts of the print. That means that during inspection phase the print template had to be substantially dilated in order to entirely cover the print. Our future work includes the calculation of the entire FBO in Fourier domain, the modeling of the most significant spatial deformations of the print, and the estimation of non-rigid transformation that will eliminate spatial deformations (ellipse and helix) on each sample image.

Acknowledgement

This work was supported by the Ministry of Higher Education, Science and Technology, Republic of Slovenia under grants L2-4072, L2-5472, by Sensum, Computer Vision Systems, and by the European Union, European Social Fund.

References

[1] Adrienne Berman. Reducing medication errors through naming, labeling, and packaging. *Journal of Medical Systems*, 28(1):9–29, Feb. 2004.

[2] Ronald N. Bracewell. *The Fourier Transform and Its Applications*. McGraw-Hill Higher Education, 2000.

[3] Marko Bukovec, Ziga Spiclin, Franjo Pernus, and Bostjan Likar. Automated visual inspection of imprinted pharmaceutical tablets. *Measurement Science & Technology*, 18(9):2921–2930, Sept. 2007.

[4] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799, 1995.

[5] J. Derganc, B. Likar, and F. Pernus. Shading correction and segmentation of color images. In *Proceedings of the 2nd International Symposium on Image and Signal Processing and Analysis, 2001. ISPA 2001*, pages 345–350, 2001.

[6] David Edwards. Applications of capsule dosing techniques for use in dry powder inhalers. *Therapeutic delivery*, 1(1):195–201, July 2010.

[7] FDA. Code of federal regulations (21CFR206) imprinting of solid oral dosage form drug products for human use. `http://www.accessdata.fda.gov/`.

[8] M.J. Islam, M. Ahmadi, and M.A. Sid-Ahmed. Image processing techniques for quality inspection of gelatin capsules in pharmaceutical applications. In *10th International Conference on Control, Automation, Robotics and Vision, 2008. ICARCV 2008*, pages 862 –867, Dec. 2008.

[9] M.J. Islam, S. Basalamah, M. Ahmadi, and M.A. Sid-Ahmed. Capsule image segmentation in pharmaceutical applications using edge-based techniques. In *2011 IEEE International Conference on Electro/Information Technology (EIT)*, pages 1 –5, May 2011.

[10] A.C. Karloff, N.E. Scott, and R. Muscedere. A flexible design for a cost effective, high throughput inspection system for pharmaceutical capsules. In *IEEE International Conference on Industrial Technology, 2008. ICIT 2008*, pages 1–4, Apr. 2008.

[11] CD Kuglin and DC Hines. The phase correlation image alignment method. *IEEE Conference on Cybernetics and Society*, pages 163–165, 1975.

[12] Elias N Malamas, Euripides G.M Petrakis, Michalis Zervakis, Laurent Petit, and Jean-Didier Legat. A survey on industrial vision systems, applications and tools. *Image and Vision Computing*, 21(2):171–188, Feb. 2003.

[13] Miha Možina, Dejan Tomaževič, Franjo Pernuš, and Boštjan Likar. Real-time image segmentation for visual inspection of pharmaceutical tablets. *Machine Vision and Applications*, 22(1):145–156, Jan. 2011.

[14] Z. Špiclin, B. Likar, and F. Pernuš. Real-time print localization on pharmaceutical capsules for automatic visual inspection. In *2010 IEEE International Conference on Industrial Technology (ICIT)*, pages 279 –284, Mar. 2010.

[15] Fridrun Podczeck and Brian E. Jones. *Pharmaceutical Capsules*. Pharmaceutical Press, 2004.

[16] D. M. Tsai and C. T. Lin. Fast normalized cross correlation for defect detection. *Pattern Recognition Letters*, 24(15):2625–2631, Nov. 2003.

[17] P. Vasudevan, T. DelGianni, and W. O. Robertson. Avoiding medication mixups - identifiable imprint codes. *Western Journal of Medicine*, 165(6):352–354, Dec. 1996.

[18] Barbara Zitová and Jan Flusser. Image registration methods: a survey. *Image and Vision Computing*, 21(11):977–1000, Oct. 2003.

Computer Vision Winter Workshop 2014, Zuzana Kúkelová and Jan Heller (eds.)
Křtiny, Czech Republic, February 3–5
Czech Society for Cybernetics and Informatics

Visual Recognition and Fault Detection for Power Line Insulators

Markus Oberweger, Andreas Wendel[1], and Horst Bischof

Institute for Computer Graphics and Vision
Graz University of Technology, Austria
markus.oberweger@student.tugraz.at, {wendel, bischof}@icg.tugraz.at

Abstract *The inspection of high voltage power lines is an important task in order to prevent failure of the transmission system. In this work, we present a novel approach to detect insulators in aerial images and to analyze them automatically for possible faults. Our detection algorithm is based on discriminative training of local gradient-based descriptors and a subsequent voting scheme for localization. Further, we introduce an automatic extraction of the individual insulator caps and check them for faults by using a descriptor with elliptical spatial support.*
We demonstrate our approach on an evaluation set of 400 *real-world insulator images captured from a helicopter and evaluate our results with respect to a manually created ground-truth. The performance of our insulator detector is comparable to other state-of-the-art object detectors and our insulator fault detection outperforms existing methods.*

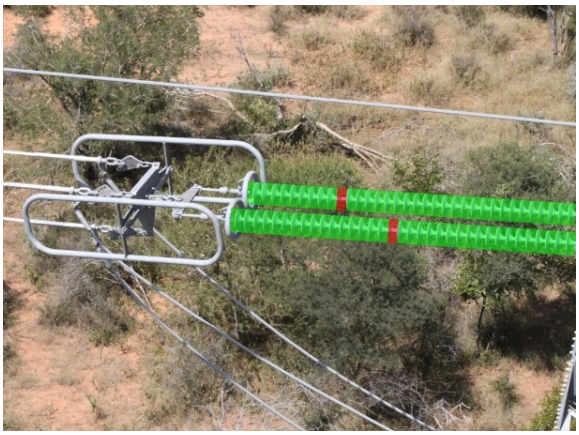


Figure 1: Our method detects insulators in highly cluttered aerial images and performs an automatic fault analysis. The faults are highlighted in red.

1 Introduction

High voltage power lines and transmission systems become more and more important with the raising demand of energy, especially in context of renewable resources. Pre-emptive inspection is an essential maintenance procedure in order to keep the downtime of a power line low, but it is time- and money-consuming, requiring much manual labor. Therefore we propose a machine-aided method for insulator inspection by automatically analyzing the images taken along a power line in order to determine faulty insulators, which are among the most common problems in transmission networks [21]. While there are different types of insulators, we focus on the most common *cap and pin insulator* with its characteristic stacked caps. The insulator can be applied to the pylon in different orientations, different sizes, or combined parallel or serial. We thus do not assume a certain orientation, combination or size in our method.

In this paper we present a novel recognition method for insulators in highly cluttered images (see Fig.1), and we introduce an automatic insulator fault detector. An overview illustrating the key-features of our work is shown in Fig. 2. The main contributions of this work are

1. an insulator detector, which is invariant to insulator orientation, size and combinations, partial overlap, illumination, and background clutter based on a circular descriptor and a noise-tolerant voting scheme, and
2. an automatic insulator fault detector, which automatically partitions each insulator into its individual caps and subsequently analyzes each cap for faults based on an elliptical descriptor.

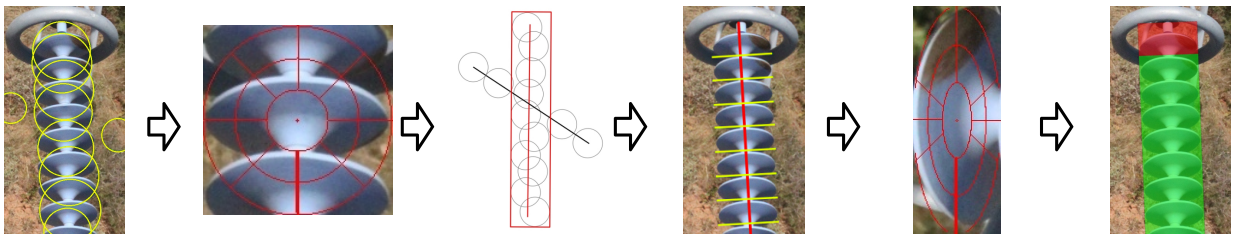
We demonstrate the performance of our approach using an image set taken from a helicopter inspection, and evaluate the quality using a manually created groundtruth, which is, to the best of our knowledge, the first systematic evaluation.

2 Related Work

While different methods for detecting insulator faults exist, e.g. visual inspection or electrotechnical measurement, our method can be used complementary with other methods and especially for identifying mechanical damage and flashover marks. However, there is no inspection method or measurement device that is able to detect all possible insulator faults [14].

For insulator detection there are several works, as e.g. [4, 9], who use the detection as enabling method for further processing, but these methods' localization is too inaccurate for our work and restricted to a certain scenario, e.g. untextured background or a camera facing the sky, and thus not working well for our highly cluttered background. Opposing to these inaccurate methods, Kawamura et al. [11] published an approach based on 3D template matching for accurate 3D localization of insulators for robot interaction. The method

[1]Current affiliation: Google Inc.



(a) (b) (c) (d) (e) (f)

Figure 2: An overview of our method. Based on detected keypoints (a) we extract our proposed descriptor (b) and classify them as insulator cap or background clutter. On the classified keypoints we perform a RANSAC-based voting scheme to locate insulator detections (c). From these detections we compute the insulator partitioning (d) and extract our elliptical descriptor on individual caps (e). From these descriptors we determine faulty caps, which are then highlighted in the original image (f). Best viewed in color.

has high computational costs and is specifically designed for their task of 3D localization, which is not necessary and not possible in our case as we neither have range images nor a suitable model readily available.

A simple method to detect insulators is provided by Zhang et al. [26] who use color thresholding, but their method is only capable of detecting tempered glass insulators due to their characteristic color, and it needs a well adjusted threshold parameter which limits this method considerably.

Two approaches [24, 13] use edge descriptors for insulator detection. Both calculate the descriptors on a dense grid, which creates a high number of false positive detections resulting in a limited applicability in a cluttered environment. Further, edge descriptors are not discriminative enough, which is indicated by a high false positive rate.

A completely different method was proposed by Zhao et al. [27] who use a modified Markov Random Field to model the repetitive geometric structure of an insulator, which is more invariant to clutter. They have only shown their method for combined insulators in groups of two or four. In our case most of the insulators are attached solely to the power line and thus their method cannot initialize the geometric models and fails to detect them.

For the task of insulator fault detection, [16] and [8] proposed methods especially for dirt detection based on high resolution images which cannot be applied in our case, because our images are taken from a greater distance and thus the spatial resolution is too low.

For the detection of missing caps from aerial images Zhang et al. [25] proposed a method predicated on an accurate binary segmentation of the insulator provided by color thresholding, which is limited by the choice of the threshold. Further, they split the insulator into ten parts, but this static partitioning does not incorporate differently sized insulators or partially visible insulators.

Up to now, no work that provides a proper evaluation which could be used as baseline for our evaluation has been published, but only practical demonstrations. Next to the wide applicability of our method, this is another reason to provide a well documented baseline for further work.

3 Insulator Recognition

We first detect the insulators in the image and based on the detections we perform the fault detection. From a recognition point of view, insulators are weakly textured objects and in our case surrounded by clutter, which makes it hard to detect them. In contrast, insulators have a rigid form with repetitive geometric structure and a distinctive circular shape of each cap, which are properties that can be exploited. Therefore we use a part-based model with a tailored circular descriptor, where each insulator cap is one part of the model. The model geometry is a line segment (the major axis), where all caps belonging to the insulator must lie close to it and near other detected caps of the insulator. Therefore we detect Difference of Gaussians (DoG) [15] keypoints in the image and extract a square image patch around the keypoint according to the size of the keypoint, which fits very well to the actual cap size. From the patch we calculate our Circular GLOH-like (CGL) descriptor. It is similar to the GLOH descriptor [17], which is in turn a circular implementation of the prevalent SIFT descriptor [15]. Our descriptor is based on image gradients, which are derived by the Scharr operator [22]. This operator exhibits better rotational invariance than other gradient operators, which is beneficiary in our case as the caps are circular. The gradients are assigned to 17 spatial regions as visualized in Fig. 2b. Note that the central radial bin does not have any angular bins. Each gradient casts a vote according to its gradient magnitude in the 16-bin orientation histogram of its spatial region, resulting in 272 dimensions. For rotation and illumination invariance the methods of [15] are implemented. For scale invariance we enlarge the spatial support according to the size of the keypoint.

We employ Principal Component Analysis (PCA) to reduce the dimensionality of our descriptor for speedup but without loss of classification performance [23]. We calculate the orthogonal eigenspace from $11.3k$ cap descriptors and project the descriptors on the reduced space spanned by the 192 components with the largest eigenvalues.

We train a k-Nearest Neighbors (kNN) classifier with the descriptors of detected DoG keypoints from the training set. Keypoints within the ground-truth mask are positive sam-

ples and randomly selected keypoints from the background negative samples. For recognition we query the classifier with the descriptors of the detected keypoints in order to distinguish between insulator caps and clutter.

From the classified keypoints we determine the bounding boxes for the insulators. We group the keypoints by their scale and apply an adapted RANSAC [7] approach on all keypoints of each scale to robustly fit the insulator model to the detected keypoints. We have to modify the original algorithm to handle multiple insulator instances in an image [19]. Therefore we determine two random initial points $\mathbf{p_1}, \mathbf{p_2}$ from all available keypoints of one scale which satisfy the proximity constraint $\|\mathbf{p_1} - \mathbf{p_2}\| < 4 \cdot sz(\mathbf{p_1})$, where $sz(\mathbf{p})$ is the keypoint size. If no point combination satisfies the proximity constraint, the algorithm terminates as no model can be initialized. From these two points we create an initial model, i.e. the line segment connecting them. We add keypoints as inliers to the model, whose distance from point to line is smaller than half the keypoint size. All inliers have to be in relative proximity to already added inliers. This ensures that we do not add false positive detections located on the line. Each valid model must contain at least five inliers. We use the best model, thus having most inliers, as detection. The bounding box is created from the inliers by calculating the minimum bounding rectangle. The inliers are removed from the available keypoints, thus each keypoint is only used for one model. We repeat the process as long as a single iteration generates a valid model, else all insulators have been detected and the algorithm terminates. Using the described procedure we first optimize the recall of the detection. In the next step we use the estimation of the fundamental period of the insulator partitioning to verify the detections by evaluating the repetitive structure within the insulator, which is not present in false positive detections.

4 Insulator Fault Detection

The detected insulators are then analyzed for faults. Each insulator is described by its major axis and divided into its individual caps along this axis, as shown in Fig. 2d. Further, we calculate an elliptical descriptor from each cap to generate a score, which serves as a level of faultiness. By using a cap-wise partitioning of the insulator we can localize faults more accurately and invariant to differently sized and truncated insulators.

4.1 Insulator Partitioning

The first step of the insulator partitioning is the estimation of the overall orientation Θ. This orientation estimation and correction is important in order to provide a fixed layout for the following cap extraction. Therefore we use a binary segmentation mask obtained from the actual detection using GrabCut [20]. On the mask we apply an image moment-based method which exploits the elongated shape of the insulator. The moment M of order $(i+j)$ is calculated as [10]

$$M_{ij} = \sum_x \sum_y x^i y^j I(x,y) \tag{1}$$

where x and y are the image coordinates and $I(x,y)$ the pixel intensity. Using M we calculate the insulator orientation as

$$\begin{aligned}\Theta = &\frac{1}{2}\operatorname{atan2}\left(2\left(\frac{M_{11}}{M_{00}} - \frac{M_{10}}{M_{00}}\frac{M_{01}}{M_{00}}\right),\right.\\ &\left.\frac{M_{20}}{M_{00}} - \left(\frac{M_{10}}{M_{00}}\right)^2 - \frac{M_{02}}{M_{00}} - \left(\frac{M_{01}}{M_{00}}\right)^2\right).\end{aligned} \tag{2}$$

The orientation is derived from the covariance matrix of the normalized second order image moments of the image mask.

In the second step of the partitioning we separate the insulator into its individual caps. Therefore we detect separation candidates by extracting Canny edges [3] and intersect them with the major axis of the insulator. Because these candidates are noisy, we employ signal processing methods to estimate and extract the separations. The separation candidates are formed as an impulse in a 1D signal $i[u]$ along the major axis u using

$$i[u] = \begin{cases} 1, & \text{if major axis intersects with edge} \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

The signal consists of impulses which are not properly aligned and thus create high frequency responses. Therefore we apply a Gaussian filter which provides a low pass filter and serves as noise estimate. For the Gaussian kernel we use $\sigma = \frac{1}{3}\lfloor\frac{1}{N}\sum_{i=1}^{N-1} d_{i,i+1}\rfloor$ where $d_{i,i+1}$ is the distance between two consecutive impulses. This kernel has been chosen to adaptively minimize the leverage on neighboring impulses due to filtering, which is necessary because of the high variance of cap sizes.

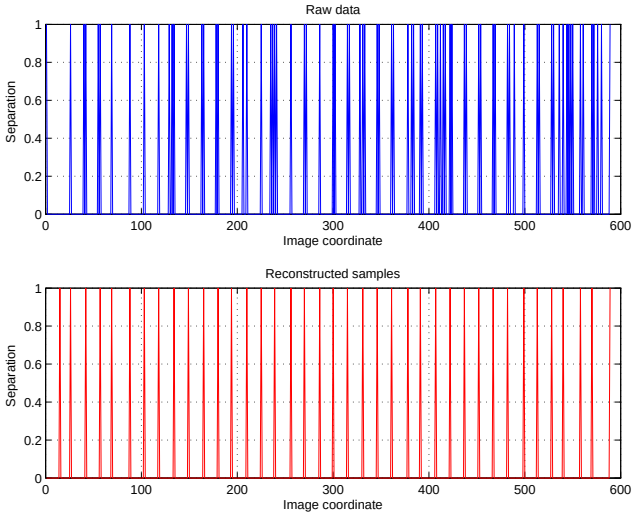
The fundamental period f of the filtered signal $x[u]$ is used to estimate the repetitive structure within the insulator. From $x[u]$ we calculate the N_{fft}-point Fast Fourier Transform (FFT) as $\mathbf{X} = \mathcal{F}\{x\}$ where $N_{fft} = 2^{\lceil \mathrm{ld}(length(x[u]))\rceil}$. In order to improve the period estimate we use a weighting function $\mathbf{w}[k]$ to suppress unwanted frequency parts, which can be caused by wrong separation candidates. $\mathbf{w}[k]$ is again a Gaussian distribution with $\sigma = \frac{k_c}{\sqrt{2}}$ centered at $k_c = \lfloor\frac{N_{fft}}{w/2}\rfloor$ where w is the width of the insulator masks bounding box which serves as an estimate. Further, $f = \lfloor\frac{N_{fft}}{k_{max}}\rfloor$ and k_{max} is determined by

$$k_{max} = \arg\max_k (|\mathbf{X}| \cdot \mathbf{w}[k] \quad | \quad k > 3) \tag{4}$$

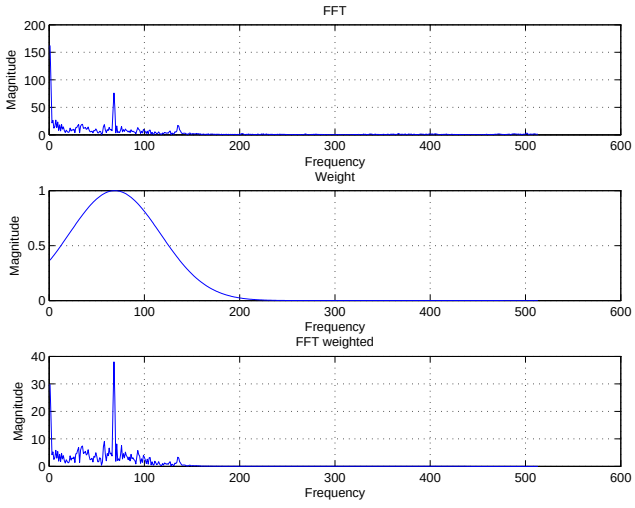
where $k > 3$ is used to suppress the dominant constant component of the signal. We estimate f on the major axis and on two lines parallel to this axis and use the median of the detected frequencies, which improves the robustness.

The alignment of the partitioning within the insulator is calculated using cross-correlation of the filtered input signal $x[u]$ and an idealized partitioning created from the fundamental period. The determined offset is used for matching the calculated to the nearest detected separations. If a separation is missing, e.g. not detected, a separation is thus automatically added at the most likely position.

In Fig. 3a the input signal $i[u]$ is compared to the resulting partitioning. Note the noisy input signal with multiple responses for each separation and with noise within the caps, which are removed in the result. In Fig. 3b the shown spectrum exhibits a strong peak at $k_{max} = 67$ for a $N_{fft} = 1024$ point FFT, which correctly results in $f = 15px$.



(a)



(b)

Figure 3: Insulator partitioning (measured $f = 15px$). (a) shows the reconstruction of the partitioning and (b) shows the estimation of the fundamental period.

4.2 EGL Descriptor

Based on the partitioning we extract an elliptical descriptor from each cap, which fits an insulator cap very well and thus minimizes the influence of neighboring caps or background clutter. In contrast to the CGL descriptor our Elliptical GLOH-like (EGL) descriptor contains gradient histograms for elliptical spatial regions (planar elliptical coordinates [18]). The spatial layout of the descriptor is shown in Fig. 2e, which again is divided into 17 regions with a 16-bin histogram of gradient orientations for each region. The axes of the ellipse are given by the size of the insulator cap. The EGL descriptor is not rotational invariant, which is not needed, because the insulator tilt can be corrected by using the insulator orientation. The illumination invariance and scale invariance are the same as with the CGL descriptor.

4.3 Fault Detection

Based on the extracted descriptors from each insulator cap we determine outliers, which are the faulty caps. We have to use an unsupervised outlier detector due to the lack of faulty caps for training and due to high intraclass variations of the insulators (background, viewpoint). Further, we do not treat the faults as binary classification problem, but we assign a score to each cap of an insulator. A higher score characterizes a higher dissimilarity to the other caps of the insulator and thus a more likely faulty cap. Therefore we use the Local Outlier Factor (LOF) approach proposed by Breunig et al. [2], which provides a score for the dissimilarity by using the distance of a descriptor to the k nearest neighbors as an estimate for the local descriptor density. We assume that an outlier has a lower local density in high dimensional space. The relation of this distance to the distance of its neighbors is used to identify outliers. As distance measure $d(A, B)$ we use the normalized L1 distance between two descriptors A and B in D-dimensional space,

$$d(A,B) = \sum_{i=1}^{D} \frac{|A_i - B_i|}{max_i - min_i} \tag{5}$$

where max_i and min_i are the minimal and maximal values of dimension i over all descriptors. We use $k = 3$ neighbors, initialize the distances with all descriptors of an insulator, and add each test descriptor separately to the set in order to preset the distances to a task-specific range [12].
The scores exhibit different ranges for each insulator. For a global representation we normalize the scores for each insulator to a range of $[0 \ldots 1]$. Further, we enhance the distinctiveness by thresholding the score sequence and set all values that do not excess the confidence level of $\frac{1}{\sqrt{2}}$ of the standard deviation above mean to 0. This provides a more discriminative visualization as shown in Fig. 2f.

5 Evaluation

In this section we present and discuss the results of the evaluation. To the best of our knowledge, there is currently no publicly available dataset for insulator detection. Therefore we use our own evaluation set, which contains 400 images (size $2352 \times 1568px$) with 375 labeled insulators, whereat 20

out of over $11.3k$ caps are labeled faulty (4 flashover damages and 16 cracked caps). For evaluation and training we use a segmented ground-truth, which is generated by manually segmenting the insulators using a GrabCut [20] based labeling tool. Truncated insulators with less than five visible caps are not labeled. We evaluate on a computer with a Core 2 Duo with 2.6GHz and 4GB of RAM.

5.1 Insulator Recognition

We first evaluate the keypoint classification and subsequently the recognition based on the classified keypoints. For the evaluation of the recognition there has not been published a work that provides proper evaluation metrics which could be used as baseline, but only practical demonstrations.

5.1.1 Keypoint Classification The evaluation of the keypoint classification is based on the true positive rate (TPR), the fraction of correctly identified caps to all caps, and the true negative rate (TNR), the fraction of correctly identified clutter to all clutter samples. For the keypoint classification a high TPR and TNR are required in order to distinguish between cap and clutter keypoints. From our evaluation set we automatically extract $11.3k$ caps located on DoG keypoints within the ground-truth mask and $11k$ clutter samples randomly selected from the background as ground-truth.
Tab. 1 shows the classification results for different descriptor types. The classification rates are obtained by using a kNN classifier with 2-fold cross-validation. Our CGL descriptor scores the highest TPR and TNR rate, thus it is the most suitable descriptor for this task. The high TNR of 99.7% is essential for efficient clutter suppression, but a slightly smaller TPR of 92.7% can be tolerated as not all caps of an insulator must be detected in order to detect the insulator itself due to our part-based model.

Descriptor	**CGL**	SIFT [15]	SURF [1]
TPR	**92.7%**	92.5%	75.8%
TNR	**99.7%**	92.4%	90.5%
Average runtime	1.4ms	1.9ms	**0.25ms**

Table 1: Average keypoint classification rates. Our descriptor (CGL) performs best.

The detection of keypoints works best with a DoG [15] detector. A dense grid (e.g. 7 scales from 20 to $100px$) is not applicable because of the high runtime and the high number of false positive detections. Other detectors as e.g. an approximated Hessian detector [1] cannot locate the insulator caps accurately, or detectors that detect corner-like structures rather locate the keypoints on the boundaries than in the center of the caps.

5.1.2 Recognition We evaluate the insulator detector on our evaluation set, where each connected component in the ground-truth mask is determined as insulator and the minimal bounding rectangle is used as ground-truth bounding box. Note that these are rotated rectangles, therefore they fit the insulators very well.
In order to evaluate the localization of our method we use the well-known Pascal score [5], which is calculated from the overlap of our generated bounding box B_c to the ground-truth B_{gt} by

$$p(B_c, B_{gt}) = \frac{area(B_c \cap B_{gt})}{area(B_c \cup B_{gt})}. \tag{6}$$

An object is considered detected if $p(B_c, B_{gt}) > 0.5$.
As objective we want to maximize the number of correct detections and minimize the number of false detections. The two used evaluation metrics are precision, the fraction of correct detections to the total number of detections made by our detector, and recall, the fraction of correctly detected objects to the number of annotated objects. Our detector provides a score for each detection, which is used to vary the trade off between these two metrics. The score is calculated by

$$score(\mathbf{p}, L) = \left(1 - \frac{\sum_{i=1}^{N} d_{\mathbf{p},L}(\mathbf{p_i}, L)}{\sum_{i=1}^{N} sz(\mathbf{p_i})/2}\right) \cdot N \tag{7}$$

where $\mathbf{p}$ is the inlier set, L the estimated model, N is the number of inliers and $d_{\mathbf{p},L}$ the Euclidean distance from inlier to model. The score is higher if a model contains more inliers or better fitting inliers.
We use the Precision-Recall Curve (PRC) as performance measurement, which is shown in Fig. 4 for our detector. The curve is constructed in accordance to [5] by using the interpolated precision. We perform 2-fold cross-validation and plot the averaged precision and recall values. The recall reaches a maximum of 98% and drops rapidly for higher recall scores. Most false positives are caused by insufficient overlap of the detected to the ground-truth bounding box. Only about 7% of the false positives are actually located on the background or on the pylons, thus showing efficient clutter suppression.

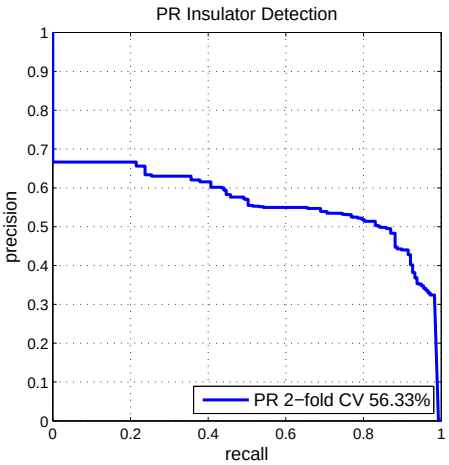
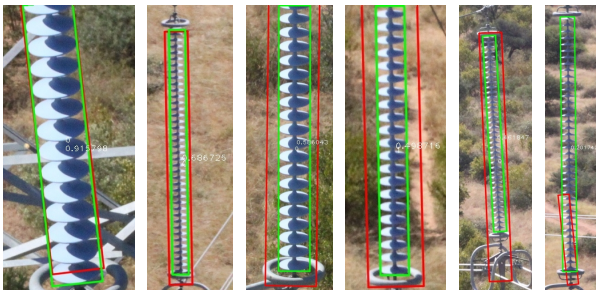


Figure 4: PRC for insulator detection. The average precision is over 56%.

Badly detected boxes are identified by a low overlap value as shown in Fig. 5. For our evaluation the required Pascal criterion is very strict due to the fact that we use rotated bounding boxes, which is not originally intended [5].

Subjectively speaking, an overlap score of 0.5 already fits the insulator very well, whereas a lower threshold might be concerned, e.g. $p(B_c, B_{gt}) > 0.4$ already improves the recall to 100%. Using an overlap threshold of 0.5 the insulators in Fig. 5 (a)-(c) would be considered correct detections, and Fig. 5 (d)-(f) false detections although the first two fit well. Only the detection in Fig. 5f completely fails, because the line model is initialized badly and further the detected insulator caps in the upper part are not included into the model due to a large distance from the model.



(a) 0.9 (b) 0.68 (c) 0.58 (d) 0.48 (e) 0.45 (f) 0.2

Figure 5: Detection results (red) with overlap score $p(B_c, B_{gt})$ stated. Best viewed in color.

5.2 Insulator Fault Detection

For the fault detection, a proper insulator partitioning is essential. Therefore we first evaluate the insulator partitioning and then the fault detection itself. Fig. 1 shows the fault detection results for a mechanical damage and Fig. 2f for a flashover damage.

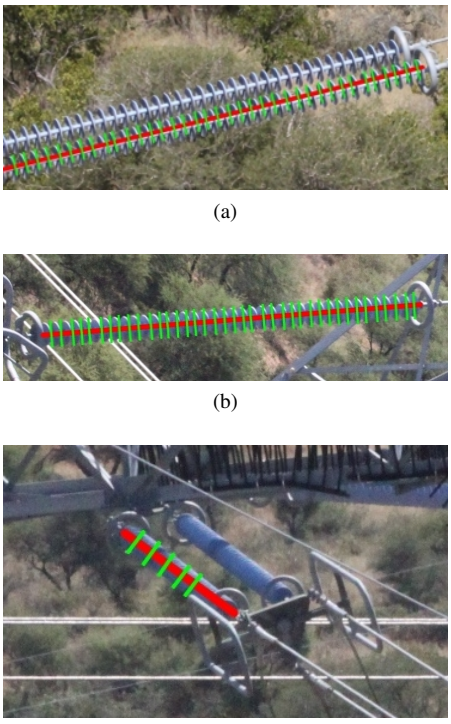
5.2.1 Insulator Partitioning For the evaluation of the partitioning we compare the calculated orientation to the orientation of the minimum bounding rectangle of the ground-truth segmentation mask. We have implemented the Hough-based method of [25] as baseline. As evaluation criterion we use the angular error $e(\Theta, \Theta_{gt}) = |\Theta - \Theta_{gt}| \pmod{\frac{\pi}{2}}$, where Θ_{gt} is the orientation of the ground-truth and Θ the orientation provided by the method. Only the absolute orientation matters due to the symmetric shape of the insulator. The mean $\mu(e)$ and the standard deviation $\sigma(e)$ of the angular error are used to compare the methods. A smaller mean indicates a more accurate orientation estimation and a smaller standard deviation shows, that the errors are less scattered.
The evaluation results in Tab. 2 show, that our moment-based method is more accurate and requires less computation time than the Hough-based approach. The orientation of the bounding box of the actual detection requires no additional computation, but it is less accurate than our proposed method and prone to outliers. A more precise orientation provides a higher accuracy of the partitioning.

For the evaluation of the partitioning we measured the fundamental period from the images and use it as ground-truth. The fundamental period is correctly computed for 84% of all insulators, w.r.t. a tolerable deviation of $\pm 5px$ or 10%. For the cap extraction the creation of a ground-truth

Method	$\mu(e)$	$\sigma(e)$	Average runtime
Moment-based	$\mathbf{0.33}^\circ$	$\mathbf{0.48}^\circ$	12ms
Hough-based [25]	1.57°	2.03°	430ms
Bounding box	1.02°	1.24°	**0ms**

Table 2: Insulator orientation evaluation.

is not feasible, thus resulting in manual checks. A proper partitioning requires a cap width of at least $10px$, otherwise the separation features between the individual caps vanish. The partitioning works for different perspectives as shown in Fig. 6a and 6b, but fails if there are no separation features as depicted in Fig. 6c. For that sample also humans fail to separate the caps.



(a)

(b)

(c)

Figure 6: Partitioning results. Note the different sizes (red line is $5px$ wide). Best viewed in color.

5.2.2 Fault Detection We compare our fault detection method to the approach of [25] based on Gray-Level Co-occurrence Matrices (GLCM). In order to make their scores comparable we use our partitioning to calculate their features, instead of constantly ten parts.
A detailed view of a score sequence is given in Fig. 7, which shows the scores for a flashover damage (see Fig. 2e) at cap 1. The scores are normalized, but not thresholded as described in Section 4.3 for a better illustration. The ground-truth has a value of 1 for a faulty cap and 0 for non-faulty caps. Both methods score high values for the faulty caps, but the values scored for non-faulty caps are quite different. By using our EGL-based approach, we efficiently suppress background clutter and are invariant to faults in the segmentation mask, in contrast to the GLCM-based method, which

cannot distinguish between a faulty cap and a defect in the segmentation mask.

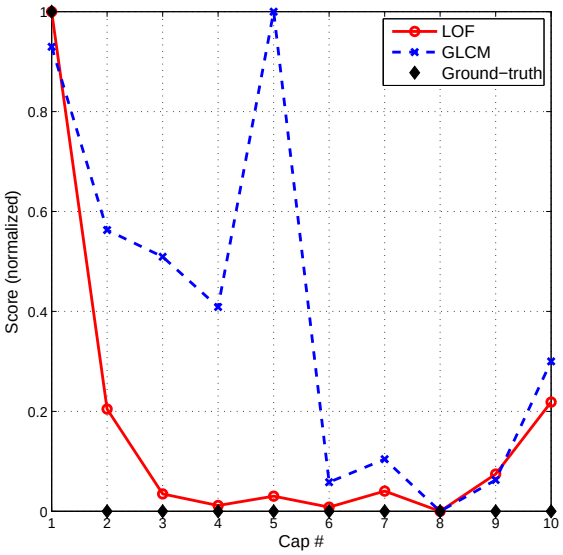


Figure 7: Score sequence for a flashover damage at cap 1. Our LOF-based approach can clearly distinguish between faulty and non-faulty caps.

For the evaluation of our fault detection we use the Receiver Operating Characteristic (ROC), which is independent of class skew [6] and thus advantageous in the case of fault detection because the faulty samples make up only a fraction of all samples. Although we use a continuous score for the fault detection, the evaluation can be seen as a binary classification problem: faulty caps are the desired positive class, non-faulty caps are the negative class. As performance measurements we use the TPR, the fraction of successfully detected faulty caps to all faulty caps, and the false positive rate (FPR), the fraction of mistakenly identified faulty caps to all non-faulty caps. A comparison of different ROC for insulator fault detection is shown in Fig. 8. The ROC is created by using the fault score as threshold. Our EGL descriptor with LOF performs best and achieves a TPR of 95% at a FPR of 12%.

6 Conclusion

In this work we have presented a novel approach for insulator recognition and a subsequent automatic fault detection from aerial images. We introduced a method for insulator recognition using a part-based model with local image features and a RANSAC-based clustering approach, which enables the detection of insulators in a highly cluttered environment regardless of their orientation, size or combinations.

Further, we proposed a method for insulator fault detection based on a descriptor with elliptical spatial support. We used LOF to assign a score of faultiness to each insulator cap extracted by our automatic partitioning algorithm. This provides an accurate cap-wise fault assessment of the insulator under different photometric and geometric conditions.

Both methods have been evaluated thoroughly and we have established a baseline, which can be used for comparison in future works.

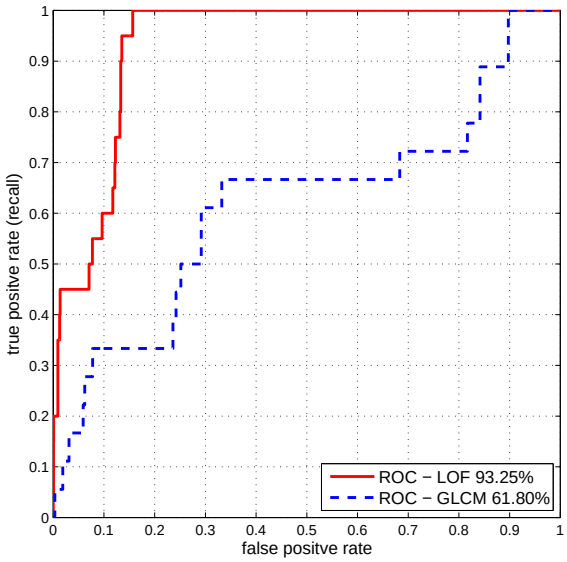


Figure 8: ROC for fault detection, with area under the curve stated in the caption. Our proposed method (LOF with EGL descriptor) significantly outperforms GLCM [25].

Acknowledgement

This work has been supported by the Austrian Research Promotion Agency (FFG) project FIT-IT Pegasus (825841/10397).

References

[1] Herbert Bay, Tinne Tuytelaars, and Luc J. Van Gool. SURF: Speeded Up Robust Features. In *Proc. ECCV*, 2006.

[2] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. LOF: Identifying density-based local outliers. In *Proc. SIGMOD*, 2000.

[3] John Canny. A computational approach to edge detection. *Trans. PAMI*, 8(6):679–698, 1986.

[4] Jaime Del-Cerro, Antonio Barrientos, Pascual Campoy, and Pedro J. García. An autonomous helicopter guided by computer vision for inspection of overhead power cable. In *Proc. IROS Workshops*, 2002.

[5] Mark Everingham, Luc J. Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *IJCV*, 88(2):303–338, 2010.

[6] Tom Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

[7] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. ACM*, 24(6):381–395, 1981.

[8] Yumin Ge, Baoshu Li, Shutao Zhao, and Chengzong Pang. Detection of the insulator dirtiness based on the computer vision. In *Proc. CICED*, 2006.

[9] Irene Y. H. Gu, Unai Sistiaga, Sonja M. Berlijn, and Anders Fahlström. Online detection of snowcoverage and swing angles of electrical insulators on power transmission lines using videos. In *Proc. ICIP*, 2009.

[10] Ming-Kuei Hu. Visual pattern recognition by moment invariants. *Trans. Information Theory*, 8(2):179–187, 1962.

[11] Kentaro Kawamura, Mark D. Wheeler, Osamu Yamashita, Yoichi Sato, and Katsushi Ikeuchi. Localization of insulators in electric distribution systems by using 3D template matching from multiple range images. In *Proc. IROS*, 1998.

[12] Aleksandar Lazarevic and Vipin Kumar. Feature bagging for outlier detection. In *Proc. ICKDDM*, 2005.

[13] Weiguo Li, Gaosheng Ye, Feng Huang, Shikun Wang, and Wenzhi Chang. Recognition of insulator based on developed MPEG-7 texture feature. In *Proc. ICISP*, 2010.

[14] Zheng Li and Yi Ruan. Fault diagnosis system for the inspection robot in power transmission lines maintenance. In *Proc. OIT*, 2009.

[15] David G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[16] Xin Mei, Tiecheng Lu, Xiaoyun Wu, and Bo Zhang. Insulator surface dirt image detection technology based on improved watershed algorithm. In *Proc. PEEC*, 2012.

[17] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *Trans. PAMI*, 27(10):1615–1630, 2005.

[18] Parry Hiram Moon and Domina Eberle Spencer. *Field Theory Handbook: Including Coordinate Systems, Differential Equations and Their Solutions*. Springer, 1971.

[19] Julien Rabin, Julie Delon, Yann Gousseau, and Lionel Moisan. MAC-RANSAC: a robust algorithm for the recognition of multiple objects. In *Proc. 3DPVT*, 2010.

[20] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM TOG*, 23(3):309–314, 2004.

[21] Murari Mohan Saha, Jan Jozef Izykowski, and Eugeniusz Rosolowski. *Fault Location on Power Networks*. Springer, 2010.

[22] Hanno Scharr. *Optimale Operatoren in der digitalen Bildverarbeitung*. PhD thesis, Ruprecht-Karls-Universitt Heidelberg, 2000.

[23] Simon Winder, Gang Hua, and Matthew Brown. Picking the best DAISY. In *Proc. CVPR*, 2009.

[24] Jian Zhang and Ruqing Yang. Insulators recognition for 220kv/330kv high-voltage live-line cleaning robot. In *Proc. ICPR*, 2006.

[25] Xinye Zhang, Jubai An, and Fangming Chen. A method of insulator fault detection from airborne images. In *Proc. GCIS*, 2010.

[26] Xinye Zhang, Jubai An, and Fangming Chen. A simple method of tempered glass insulator recognition from airborne image. In *Proc. ICOIP*, 2010.

[27] Jingjing Zhao, Xingtong Liu, Jixiang Sun, and Lin Lei. Detecting insulators in the image of overhead transmission lines. In *Proc. ICIC*, 2012.

Computer Vision Winter Workshop 2014, Zuzana Kúkelová and Jan Heller (eds.)
Křtiny, Czech Republic, February 3–5
Czech Society for Cybernetics and Informatics

Detecting handwritten signatures in scanned documents

İlkhan Cüceloğlu[1,2], Hasan Oğul[1]

[1]Department of Computer Engineering, Başkent University, Ankara, Turkey
[2]DAS Document Archiving and Management Systems CO., Ankara, Turkey
hogul@baskent.edu.tr, ilkhan.cuceloglu@das.com.tr

Abstract *Automated localization of a handwritten signature in a scanned document is a promising facility for many banking and insurance related business activities. We describe here a discriminative framework to extract signature from a bank service application document of any type. The framework is based on the classification of segmented image regions using a set of representative features. The segmentation is done using a two-phase connected component labeling approach. We evaluate solely and combined effects of several feature representation schemes in distinguishing signature and non-signature segments over a Support Vector Machine classifier. The experiments on a real banking data set have shown that the framework can achieve a reasonably good accuracy to be used in real life applications. The results can also provide a comparative analysis of different image features on signature detection problem.*

1 Introduction

In spite of a drastic increase in the use of electronic data in bank applications, a signature in a printed document is still considered to be most reliable way of user commitment, approval and verification. Bank officers usually scan a signed copy of the document to be processed and manually inspect the signature. Manual inspection is very suspect to errors and misleading interpretations. Furthermore, it requires an additional workload, which causes either an increase in the cost of human resources for bank managerial or excessive working hours for current officers. Therefore, it has become essential to use intelligent software techniques for automated analysis of documents to perform signature-related tasks.

An important task in automated processing of scanned bank application forms is to find the position of the signature. Although the signature analysis has received a great attention of the researchers in the field of document analysis and recognition, their effort has been enriched in the direction of signature verification problem [8,16,18], where it is required to model the identity between two previously segmented signature images. There is too fewer attempt in the analysis of signature presence or position in documents.

The task of detecting signatures in scanned documents poses several challenges. First, these types of document images have usually very low resolution, which makes them difficult to enhance. Second, the background of each document is different and usually not known beforehand. Third, documents are subject to restricted processing time due to the urgency of applications. Finally, and maybe the most importantly, the documents often contain auxiliary lines and other handwritten characters that resemble or overlap with signatures.

In one of the earliest studies, Djeziri et al. [5] tackled the signature extraction problem by an intuitive approach that is supposed to mimic the human visual perception. They introduced the concept of filiformity as a criterion for the curvature characteristics of handwritten signatures. Though successful in clean bank cheques, this approach fails when other filiform objects are present in the document. Madasu et al. [12] tried to crop the image segment by estimating the area in which the signature lies using a sliding window. They then analyzed the local entropy derived from the pixel-based density of the region to decide its being signature or not. This approach disregarded the noise and therefore high-density regions are reported as signatures incorrectly. Madasu et al. [13] and Chalechale et al. [2] used geometric features including area, circularity, aspect ratio, size and position to analyze segmented regions. These features are compared using pairwise similarity metrics such as Manhattan distance. Jayavdean et al. [9] proposed another method based on variance analysis in a grid placed on the putative signature position in gray-level cheque images. Zhu et al. introduced the concept of multi-scale saliency feature to define signature characteristics [19] and also used it for signature verification [20]. A three-stage procedure was proposed by Mandal et al. [14] to extract signatures. First stage locates signature segment using word-level feature extraction. Second stage separates signature strokes that overlap with the printed text. Final stage uses skeleton analysis to classify real signature strokes. They used gradient-based features to feed a machine learning classifier. Ahmet et al. [1] used Speeded Up Robust Features (SURF) to classify segmented image blocks. Esteban et al. [6] approached the problem with the assumption that signature detection is normally followed by a verification stage and applied an evidence accumulation strategy in order to utilize a set of known signatures at hand. Although they achieved a high accuracy with this approach, their assumption is not

generally true since the bank application forms are often filled by new customers and this process is not followed by a verification but an archiving step.

In this study, we propose an automated framework to extract handwritten signatures from multi-page bank application documents assuming that the customer has no previous sign in the current database. The framework is discriminative in that a set of model parameters are learned through positive and negative signature samples. The learning model is built upon Support Vector Machines (SVMs) while the feature sets that feed SVM are selected using several local and global image feature representation schemes. The experiments on real data sets have shown that the framework can achieve a reasonably good accuracy to be used in real life applications. The study can also provide a comprehensive comparison analysis of the contributions of different image descriptors in signature extraction problem.

2 Materials and Methods

The general handwritten signature detection framework is shown in Figure 1. The process starts with the pre-processing stage to acquire and get a better view of input image. In the second stage, an image segmentation procedure is applied to obtain signature candidates. The third stage is where the candidate signatures are represented by a set of numerical features extracted from image content. The feature vectors are fed into machine learning classifiers in the final stage. The details of the proposed framework are given as follows.

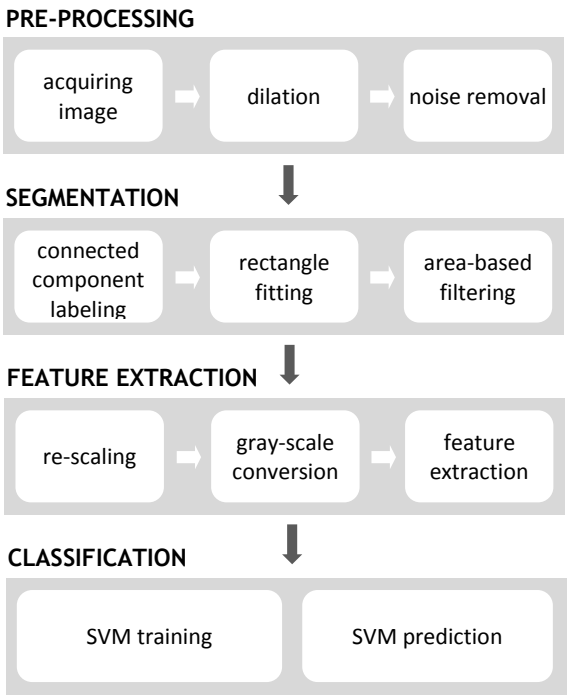


Figure 1: A brief outline of proposed framework for signature extraction

2.1 Pre-processing

Pre-processing stage involves acquiring the input image and extracting single page samples from multi-page input. To enhance the input image, we consider applying a simple dilation operator to make the lines more visible and a noise removal step by Median filtering to smooth the image. No other pre-processing step is applied to keep as much as possible information for segmentation phase.

2.2 Segmentation

Segmentation is a crucial step in signature detection. It is desired to catch all true positive (signature-containing) segments while allowing false negatives to some degree as they are expected to be removed in following steps. For an effective segmentation, we follow a two-scan connected component labelling approach described in He et al. [7]. This approach involves three processes: (1) assigning each pixel a provisional label by a 4-neigbor mask and finding equivalent labels, (2) recording equivalent labels and finding a representative label for each equivalent provisional label, (3) replacing each provisional label by its representative label. For a more efficient version, the image pixels and lines are processed two by two as opposed to conventional methods that perform these processes one by one. After connected component labelling, corresponding image regions are fit into a rectangle to record candidate signature segments. The segments involving less than 350 pixels are removed since the signature regions are often larger.

2.3 Feature Extraction

Since we follow a discriminative approach for signature detection, each segment should be vectorized to be fed into a machine learning classifier. This vectorization is made by selecting and extracting a set of content-based features to represent the segment as being a signature or not. Before feature extraction, a number of pre-processing steps are performed. First, the segments are re-scaled into 126x126 pixels. Then, the bitonal image is converted into a gray-scale image by applying 2x2 median filtering 5 times. We evaluate several feature representation schemes to distinguish signatures from other connected components.

***Gradient-based features*:** First feature set is based on the local pixel representatives that are evaluated by gradient vectors. To extract this feature set, the re-scaled segment image is partitioned into 9x9 blocks. The arc-tangent (strength) of the gradient is quantized into 16 directions and the strength is accumulated with each of the quantized direction. This is implemented by a Robert's operator. The histograms of the values of 16 quantized directions are computed in each of 9x9 blocks. After a down-sampling from 9x9 to 5x5 by a Gaussian filter, the resulting feature vector has 400 dimensions.

HOG: Another popular gradient-based feature called HOG (Histogram of Oriented Gradients) [4] is also considered with its default parameters. Each HOG vector is composed of 8100 features, which includes 9-bin gradients on 15x15 blocks with 4 cells each.

SIFT: Third feature set is based on interest-point descriptors. SIFT (Scale Invariant Feature Transform) generates features for gray-level images called keypoints which is invariant to image scaling, rotation and partially to change in illumination [11]. The statistics of local gradient directions of image intensities are accumulated to give a summary description of the local structures in a neighbourhood around each keypoint. The feature set is highly distinctive if a sufficient number of keypoints is found.

LTP: LTP (Local Ternary Patterns) is a spatial method for modeling texture in an image. It was recently introduced by Suruliandi and Ramar [17] as an extension of Local Binary Patterns [15]. LBP is based on recognizing certain local binary texture patterns termed 'uniform'. The central pixel is compared with P pixels at the radius R of a circular neighborhood. The binary level comparisons computed along the boundary of the circular neighborhood are used to find a uniformity measure. It indeed refers to the number of spatial transitions. A pattern with uniformity level less than a predefined threshold is assigned with a label in the range 0 to P. Non uniform patterns are assigned to a single label, e.g. P+1. The LBP feature representation consists of a vector of the discrete occurrence histogram of these uniform patterns computed over the area under consideration. LTP allows operating on ternary pattern instead of binary pattern. It permits to detect the number of transition or discontinuities in the circular presentation of the patterns. The uniformity of a pattern is evaluated by such transitions that are found to follow a rhythmic pattern. The occurrence frequency of these patterns over the larger region then represents the LTP feature representation.

Global features: Last feature set is related to global description of segments. This set includes entropy, aspect ratio, and energy. Given a block of image i, and pixel density of P_i, the *entropy* is given by $E_i=-P_i logP_i$. The entropy is a measure of global information contained in that region. The *energy* is calculated by adding up the squares of all pixel intensities and dividing by the area of the segment. The *aspect ratio* is another global feature that is given by the ratio of width to height of the segment.

2.4 Classification

The classification of a segment is done using a popular machine learning method called Support Vector Machines (SVMs). SVM is a binary classifier that works based on the structural risk minimization principle. The inputs of an SVM in training phase are *n*-dimensional feature vectors which represent the predefined properties of the training samples. The SVM non-linearly maps its *n*-dimensional input space into a high dimensional feature space. In this high dimensional feature space a linear classifier is constructed. In the prediction phase, this linear classifier provides a discriminant score corresponding to the sample in question. In a binary classification task, a positive value of this score indicates that the test sample is belonging to that class. In our study, we use SVMs having a linear, polynomial and Gaussian kernels with their default input parameters in LIBSVM [3].

3 Results

3.1 Datasets

We used two datasets for performance evaluation. First dataset is an extension of a benchmark set, called Tobacco-800 [19]. It contains 755 segments, 353 of which are signatures and the others are not. It is available at www.baskent.edu.tr/~hogul/signds.rar. Second dataset contains real document images obtained from a currently operating local bank with a bilateral privacy agreement. In this dataset, there are 2670 multi-page documents where the total number of pages is 9943. In 4082 of the pages, there exists at least one handwritten signature. 5861 of the pages have no signature on them.

3.2 Experimental setup

The task in first dataset is to detect if given segment is a signature or not. For the second dataset, the task is to identify whether a given document contains a signature or not. Basically, we set out a rule that; if any of the page yields at least a signature segment as a classifier output, then this document is labelled as a signature-containing page. The actual position of the signature segment in the document is reported as the location of the signature. A 5-fold cross-validation is performed in first dataset for evaluation. To discern the practical ability of the framework in signature extraction, the system is trained using all segments, including positive and negative samples, in first data sets, and run through the documents in second independent dataset. Two datasets have no common documents. The performance evaluation is done by measuring the accuracy (the proportion of correctly predicted labels), the sensitivity (the proportion of positive samples which are correctly identified as such) and the specificity (the proportion of negative samples which are correctly identified as such). Here, a sample corresponds to an individual segment and a document page in the first and second datasets, respectively. The experiments are conducted for several classifier models alternating for feature representation schemes, noise removal application and SVM kernel used. For each model, only the results for the kernel that leads to the best accuracy are reported.

3.3 Empirical results

Table 1 shows the results for signature prediction task in the first dataset. The table demonstrates the results obtained with single feature representation schemes and some their combinations as well. It is obvious that SIFT feature set can not contribute so much in signature detection; all performance metrics with solely SIFT feature is too low in comparison to the others. HOG feature set provides the highest specificity with a terrible sensitivity and lower accuracy. The gradient-based and LTP feature sets can achieve the highest accuracy levels with a reasonable balance between specificity and sensitivity, whilst the LTP feature set performs slightly better. On the other hand, the performance achieved by solely use of LTP can not be enhanced with the addition of other features in an integrated feature set. Although

global feature set looks useless in its single use, it can improve the overall accuracy when it is combined with gradient-based features. This combination indeed attains the highest performance for all metrics. An interesting result is that a noise removal step is not useful even harmful for signature detection performance. This is probably because of the fact that connected lines inside the signatures, which are considered to be major indicators of being signature, are slightly removed by the noise reduction procedure. Knowing this fact, the noise removal is not applied in the models of integrated feature sets and the experiments for second dataset.

The experimental results for the task of detecting whether a whole document page contains a signature are given in Table 2. This experiment is performed on the second dataset containing complete document pages with a classifier model trained by all signature samples in the first data set. The results still suggest that the use of gradient-based feature sets with global features can serve the most reliable way of detecting signatures in scanned documents.

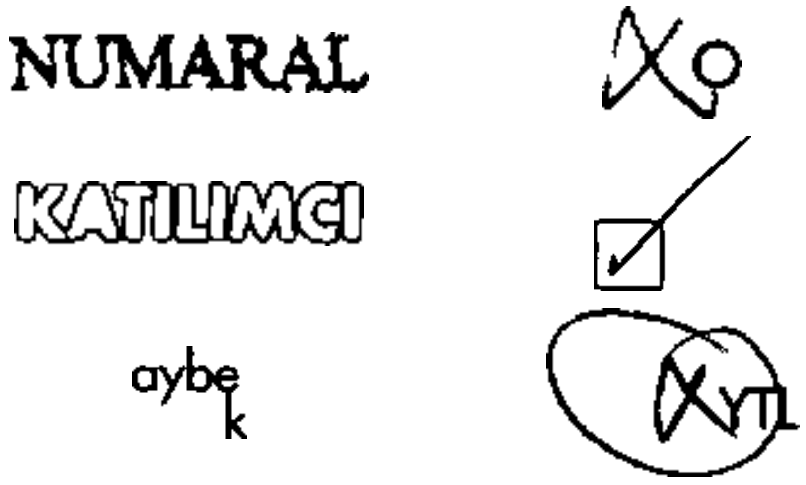
Manual inspection of individual results reveals some difficulties in signature detection problem. It evidently appears that there are two major reasons for false positives: (1) machine printed texts with highly connected fonts, and (2) handwritten initials, marks or other signs to check some choices on the application documents (Figure 2). Missing true positives is usually caused by the intervening signatures with other texts or figures of the document, especially when the signature is much smaller than the other intervened part or the faint fonts due to low scan quality or resolution (Figure 3).

Features	**Noise removal applied**	**SVM Kernel***	**Sensitivity**	**Specificity**	**Accuracy**
Gradient	*yes*	*linear*	87,0	92,8	90,1
	no	*linear*	90,1	93,0	91,7
HOG	*yes*	*linear*	39,9	94,0	68,7
	no	*linear*	35,1	93,8	66,4
SIFT	*yes*	*linear*	70,5	76,9	73,9
	no	*linear*	77,9	75,9	76,8
LTP	*yes*	*polynomial*	93,8	90,0	91,8
	no	*polynomial*	92,9	91,0	91,9
Global Features	*yes*	*polynomial*	58,9	40,8	49,3
	no	*polynomial*	71,7	66,9	69,1
Gradient + Global	*no*	*linear*	94,9	95,0	95,0
LTP + Global	*no*	*linear*	92,9	93,0	92,9
Gradient + LTP+ Global	*no*	*polynomial*	94,1	91,8	92,8

Table 1: Results with first dataset for signature prediction task in image segments.

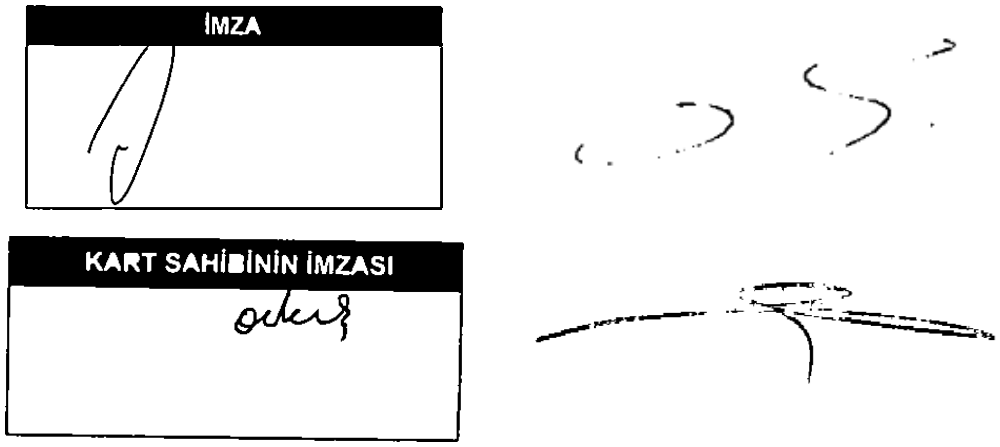
Features	**Sensitivity**	**Specificity**	**Accuracy**
Gradient	93,8	47,7	66,6
LTP	97,6	29,7	57,5
SIFT	98,5	16,3	51,2
Global Features	99,8	14,3	49,3
Gradient + Global	95,0	54,3	71,0
Gradient + Global + LTP	94,9	44,8	65,4
LTP + Global	97,8	29,1	57,3

Table 2: Results with second dataset for signature detection task in whole documents.



(a) (b)

Figure 2: Most common false positives: examples for (a) machine printed texts, (b) handwritten initials or check marks.



(a) (b)

Figure 3: Most common false negatives: examples for (a) intervened signatures, (b) faint fonts.

4 Conclusion

A framework is introduced for detecting handwritten signatures in scanned documents. The framework involves a robust and reliable segmentation stage. A number of descriptive image features are studied to discern their performances on distinguishing signature and non-signature images. Based on the empirical results, it is shown that gradient-based and LTP features are more useful in classifying signature segments in their single uses. In some cases, combining feature representation schemes can enhance the reliability of the predictions. In that sense, global features such as the aspect ratio, energy and entropy of the candidate segments can serve as lucrative complementary properties. The experimental results on real daily bank operation documents motivate the use of the framework in real life applications. The framework is quite extensible with the use of additional domain-specific local image features. Our feature work involves adding various rule-bases into different stages of the current framework to filter out improbably signature segments to obtain higher accuracy in predictions. We also anticipate that integrating the predicted document type from entire page content as a latent variable into feature sets will probably improve the detection performance.

Acknowledgement

This study was supported by the Turkey Ministry of Science, Industry and Technology by SANTEZ Project Grant No. 01522.STZ.2012-2

References

[1] S. Ahmed, M.I. Malik, M. Liwicki, A. Dengel. Signature Segmentation from Document Images. *International Conference on Frontiers in Handwriting Recognition.* 2012.

[2] A. Chalechale, G. Naghdy, P. Premaratne, A. Mertins. Document Image Analysis and Verification Using Cursive Signature. *IEEE International Conference on Multimedia and Expo*. 2004.

[3] C.C. Chang and C.J. Lin. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1--27:27, 2011.

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2005.

[5] S. Djeziri, F. Nouboud, R. Plamondon. Extraction of signatures from check background based on a filiformity criterion. *IEEE Trans. Image Process.* 7(10), 1425–1438, 1998.

[6] J.L. Esteban, J.F. Vélez, Á. Sánchez. Off-line handwritten signature detection by analysis of evidence accumulation. *IJDAR*, 15:359–368. 2012.

[7] L. He, Y. Chao, K. Suzuki. A New Two-Scan Algorithm for Labeling Connected Components in Binary Images. *Proceedings of the World Congress on Engineering*. 2012.

[8] D. Impedovo and G. Pirlo. Automatic Signature Verification: The State of the Art. *IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews*, 38:5. 2008.

[9] R. Jayadevan et al. Variance based extraction and hidden Markov model based verification of signatures present on bank cheques. *International Conference on Computational Intelligence and Multimedia Applications.* 2007.

[10] J. Li, N.M. Allinson. A comprehensive review of current local features for computer vision. *Neurocomputing*, 7:1771– 1787. 2008.

[11] D.G. Lowe. (1999). Object recognition from local scale-invariant features. *7th International Conference on Computer Vision.* 1999.

[12] V.K. Madasu, B.C. Lovell. Automatic segmentation and recognition of bank cheque fields. *Digit. Image Comput. Tech. Appl.*, 80(1): 33–40. 2005.

[13] V.K. Madasu, M.H.M. Yusof, M. Hanmandlu, K. Kubik. Automatic extraction of signatures from bank cheques and other documents, *DICTA'03*. 2003.

[14] R. Mandal, P.P. Roy, U.Pal. Signature Segmentation from Machine Printed Documents using Conditional Random Field. *International Conference on Document Analysis and Recognition*. 2011.

[15] T. Ojala, M. Pietikäinen, D. Harwood. A Comparative Study of Texture Measures with Classification Based on Feature Distributions. *Pattern Recognition*, 29:51-59. 1996.

[16] R. Plamondon, S.N. Srihari. On-line and off-line handwriting recognition: a comprehensive survey. *IEEE Trans. Pattern Anal.Mach. Intell.*, 22(1), 63–84. 2000.

[17] A. Suruliandi, K.Ramar. Local Texture Patterns –A Univariate Texture Model for Classification of Images. *16th International Conference on Advanced Computing and Communications*. 2008.

[18] W.K.H. Weiping, Y. Xiufen. A survey of off-line signature verification. *International Conference on Intellingence Mechatronics and Automation*. 2004.

[19] G. Zhu, Y. Zheng, D. Doermann, S. Jeager. Multi-scale Structural Saliency for Signature Detection. *IEEE Conference on Computer Vision and Pattern Recognition*. 2007.

[20] G. Zhu, Y. Zheng, D. Doermann, S. Jaeger. Signature detection and matching for document image retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31, 2015–2031. 2009.

Computer Vision Winter Workshop 2014, Zuzana Kúkelová and Jan Heller (eds.)
Křtiny, Czech Republic, February 3–5
Czech Society for Cybernetics and Informatics

Evaluation of performance of smart mobile devices in machine vision tasks

Danijel Skočaj, Domen Tabernik, Domen Rački, Matjaž Hegedič, Alen Vrečko, and Matej Kristan

University of Ljubljana, Faculty of Computer and Information Science
{danijel.skocaj,domen.tabernik,alen.vrecko,matej.kristan}@fri.uni-lj.si

Abstract *The recent advance in mobile processing power and imaging devices opened the door to a wide range of mobile vision applications. However, in contrast to typical specially designed machine vision systems in industrial environment, the mobile devices, expected to address the same mobile vision problems, significantly vary in imaging sensors as well as processing power. In this paper we present the results of the experimental study in which we evaluated 22 smart mobile devices in terms of accuracy that can be achieved when using these devices for measuring distances between points in the plane. The results show that smart phones and tablet computers can be used as a high precision measuring device, achieving sub-milimeter accuracy for measurement in the plane defined by a commonly accessible reference object such as an A4 sheet of paper.*

1 Introduction

In recent years we have witnessed a huge growth in popularity of mobile devices such as smart phones and tablet computers. As is often the case, this popularity growth comes hand in hand with the constant growth in performance of the mobile devices in terms of size, processing speed, interaction with the user, integrated sensors, etc. Among sensors, the integrated camera is one of the most important and prominent ones. Nowadays, owning a decently powerful computer and a decent camera, encapsulated in a small mobile device, is often considered a basic social necessity. It seems only natural that these devices are becoming an important platform for application of computer vision and machine vision solutions. These range from devices to aid visually-impaired [6], scene reconstruction [8], augmented reality [11], mobile text translation [9] to visual landmark identification [1]. It is to be expected even a greater use of mobile vision applications in the future.

The history of machine vision has seen many successful industrial applications; part of this success, however, can be certainly attributed to the controllable environment they are usually operating in. When designing a classical machine vision system, such as quality control or a system for optical measurement, there are always three major factors we have to consider: (i) the imaging device we are using, (ii) the environment (especially the illumination of the object and scene), and (iii) the position and the orientation of the object with respect to the scene and the camera. Based on these three factors we can optimize our selection of software and hardware for a specific task (e.g. we can select the camera and the lenses that suit to the problem most). Sometimes, we can (or have to) even adapt the original environmental conditions to meet the requirements of a specific solution. When some of these factors can not be controlled enough, we have to compensate that by robustifying the methods applied.

In contrast, when designing machine vision applications for mobile devices, we have very little control over the factors mentioned above, since it is the global market that dictates the choice of software and hardware (camera included) our application has to run on. All we can do is to state the minimal requirements, which again have to be often as minimal as possible. It can be quite a shock to replace a dedicated industrial camera with a plethora of different mobile phone cameras, primarily designed for social networking. Clearly, in this case the robustness of our machine vision methods faces much greater challenges. In addition our applications are now supposed to be used by non-expert users, and must, hence, be easy-to-use and possibly conform to other related requirements.

In this paper we discuss some of the challenges we face when implementing machine vision applications on smart mobile devices. Our focus is on applications that require very accurate image formation and detection, as well as localization of the individual parts of the image (e. g. for accurately measuring the size of the captured objects). We conducted an experimental study in which we assessed the ability of different models of mobile devices to handle such tasks.

Several authors have studied computer vision algorithms in the context of the processing power of modern mobile devices. Recently [2] have published results of a large-scale experiment that benchmarks computer vision algorithms potentially interesting for mobile applications. They analyze single and multi-thread implementations of these algorithms and identify major areas where architecture can improve the performance. Wang et al. [10] consider the possibility of levereging the GPU in mobile phones to speed up computer vision algorithms. Our focus was on the analysis of imaging capabilities of these devices in the context of machine vision tasks. In total we evaluated 22 mobile devices. We present the results on their appropriateness for using them as vision-based measuring devices.

The paper is organized as follows. In Section 2 we present the methods that were used to evaluate the smart mobile devices. In Section 3 we describe the evaluation proto-

col and then present the evaluation results in Section 4. We conclude with the final remarks and the future work.

2 Methods

2.1 Problem description

We can instantiate the general questions stated in the introduction with the following specific problems: How accurately can we measure distances between points in a plane with smart mobile devices and how do different mobile devices differ in their performance? Since no additional information about various devices is known in realistic scenarios, we are restricted to the information that can be obtained directly from the device and the captured image. No explicit calibration of the camera is therefore allowed.

Since we want to measure the distances in the metric absolute space, we need the means of determining the scale of the elements in the captured image. In principle, we could obtain that information from the integrated accelerometers and gyroscopes, however the results obtained in this way are not very accurate [7]. Therefore we allow the use of a reference object of a known size to determine the scale. We also assume the widespread use of our application, which means not only that this object is supposed to be of a standard size, but also be very ordinary an easily accessible. Considering all this an A4 sheet of paper seems like a natural choice. Therefore all the measurements that we will make will be done with respect to the A4 sheet of paper. The application has first to automatically detect the edges and the corners of the sheet of the paper from which the camera external parameters can be estimated using standard solutions from camera geometry.

In the following subsections we briefly describe the methods that are used to implement this process. We used the fundamental and well established computer vision techniques. The primary purpose of this implementation was to evaluate a wide range of mobile devices and to empirically estimate the upper bound of the error, and not to advance the individual methods used (which could be done and would slightly improve the overall results).

2.2 Detection of reference object

The procedure for robust detection of the paper sheet edges consists of the following well-known computer vision steps: We first apply the *Canny edge detection algorithm* to the grayscale image containing the reference object. Next, we detect the lines in the binary edge image using the *Hough transform*. Lines that lie within each others' delta regions are grouped together. Lines that withstand the filtering process are then fitted to the points in the binary edge image, using the least squares method. To ensure robust and precise line fitting, we discard the points lying in delta regions of other lines. The remaining set of points is then sorted according to the point-to-line distances. Finally, the lines are fitted to a fraction of their closest points, thus eliminating outliers from the line fitting process. In this way we detect the four edges of the reference object and then calculate their intersections with subpixel accuracy. The four intersection points are then subjects of further processing.

2.3 Estimation of camera parameters

The origin of the world coordinate system is placed in the top-left corner of the reference object, with X and Y axes running along its shorter and longer edge, respectively. To determine the camera orientation, we apply a flat-marker approach often used in augmented reality applications. The *pinhole camera model* [3] establishes the following relation between the 3D paper-sheet corners X_i and their 2D image projections x_i:

$$\mathbf{x}_i = \mathbf{K}[\mathbf{R}|\mathbf{t}]\mathbf{X}_i, \tag{1}$$

where $\mathbf{K}$, $\mathbf{R}$ and $\mathbf{t}$ are the camera calibration, rotation and translation matrices, respectively. The calibration matrix can be estimated directly, by reading the relevant parameters from the mobile device. We assume that the camera sensor matrix is not skewed and that the principal point is well approximated by the image center. The only unknown parameters are then the focal lengths in pixels (f_x,f_y) which can be estimated indirectly from the x and y field-of-view angles α_x and α_y, respectively,

$$f_x = \frac{1}{2}W\tan^{-1}(\frac{\alpha_x}{2})\ ;\ f_y = \frac{1}{2}H\tan^{-1}(\frac{\alpha_x}{2}), \tag{2}$$

where W and H are sensor width and height, respectively, in pixels. These parameters fully define the camera calibration matrix.

With the calibration matrix estimated, we use the standard approach by noting that the 3D corners X_i of the paper-sheet lie on a plane with Z coordinates equal to zero. This means that the transformation of 3D points on the sheet of paper and their projected image points is governed by a homography matrix $\mathbf{H}$, i.e.,

$$\mathbf{H} = \mathbf{K}[\mathbf{r}_1, \mathbf{r}_2, \mathbf{t}], \tag{3}$$

where $\mathbf{r}_1$ and $\mathbf{r}_2$ are the first two columns of the rotation matrix. The homography is calculated directly from the 2D-3D correspondences. Since $\mathbf{H}$ and $\mathbf{K}$ are known, we can calculate $\mathbf{r}_1$, $\mathbf{r}_2$ and $\mathbf{t}$ by inversion of (3) and from the orthogonality constraint we get the last column of the rotation matrix as $\mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2$.

3 Experimental protocol

3.1 Calibration

In our experimental study we used a reference object with a printed checkerboard pattern with a known size of the squares. Similar calibration patterns are commonly used in computer vision for camera calibration. In fact, as a first step of our study we calibrated the camera of every device using this pattern and estimated the intrinsic and extrinsic camera parameters. Calibration was performed using the Camera Calibration Toolbox for Matlab[1]. Since the calibration was done in a controlled environment we considered these parameters as as accurate as possible and used them as a reference.

[1] http://www.vision.caltech.edu/bouguetj/calib_doc/index.html

3.2 Test application

To standardize the process of capturing evaluation images, we developed a special Android application that guides the user through the process of image acquisition (see Fig. 1). Besides ensuring the uniformity of image acquisition, this approach also simulates potential real mobile applications. A blue wireframe that was supposed to be approximately aligned with the paper sheet edges guided the user how to position the camera. 18 images were captured from the hemisphere above the calibration pattern. All of these images were used in the processes of camera calibration and evaluation.

In addition, we implemented several standard computer vision algorithms that were run on every Android mobile device to test the CPU performance. OpenCV[2] implementation of these algorithms was used.

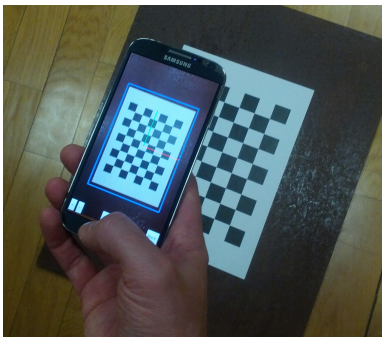


Figure 1: Test application.

3.3 Evaluation procedure

After the evaluation images were captured, we applied the procedures described in Section 2 to calculate the homography, that mapped the image points into the world coordinate frame defined by the reference object. The following estimates and errors were then calculated:

- **Error in corner detection.** Corner points of the individual squares that were detected in the process of calibration were mapped to the world coordinate frame using the estimated homography. We calculated the distance (in mm) between the detected corner points and the ground truth. In this way we estimated the error that was caused by the approximative estimation of homography, reflecting the inaccuracies in image formation and reference object corner points detection.
- **Error of estimated checkerboard width.** We calculated the distance between two specific points in the image, i.e. the width of the checkerboard (the distance between the first and the last column of squares) and compared it to the ground truth.

We also wanted to estimate the influence of the radial distortion in the images. We therefore calculated the above mentioned errors in three ways, i.e. by processing three types of images:

[2]http://docs.opencv.org

- **Original images**. We used the original captured images without any preprocessing; no image undistortion was considered.
- **Images undistorted with specific distortion coefficients**. We processed the original images, however we corrected all the detected image points (before we mapped them into the metric world coordinate frame) using the estimated camera matrix and distortion coefficients obtained in the process of the calibration of the same camera that was used to capture the image. We therefore achieved a similar result as it would be achieved by processing images undistorted with the specific distortion coefficients.
- **Images undistorted with generic distortion coefficients**. In general use, the specific distortion coefficients are not known. We therefore also performed the evaluation using generic distortion coefficients (obtained as a median of the distortion coefficients of all evaluated cameras). These parameters were fixed and used with all evaluated devices.

We also evaluated the obtained camera parameters:

- **Intrinsic parameters**. We estimated the intrinsic camera parameters and compared them with the parameters obtained in the calibration process.
- **Camera position**. By using the estimated intrinsic camera parameters and the estimated homography we estimated the positions of cameras during the image acquisition with respect to the reference object. We compared these positions with the extrinsic camera parameters obtained in the calibration process.

Finally, we measured the processing power of the individual devices. We therefore measured the **performance of the CPUs**. On every Android device we run four tests composed of the algorithms commonly used in computer vision:

- *Canny*. Canny edge detection.
- *Hough*. Line detection with Hough transform.
- *GrabCut*. Image segmentation with GrabCut method.
- *Gauss*. Image smoothing with Gaussian kernel.

Every test was executed on different images with different resolutions. The same images and algorithms were used on all evaluated devices.

4 Evaluation results

In this section we present the results of the experimental evaluation. We tried to estimate the errors up to 0.1 mm accuracy. However, since we did not have an adequate highly accurate measuring device to measure the ground truth data on the reference object, we probably also made small measurement errors, which, however, did not significantly influence the overall results and the main conclusions we can draw from them.

There was also a small number of images (around 3%), were the homography estimation process described in Section 2 failed due to several reasons, mainly due to bad quality of the captured images. Since the robustness of the homography estimation process was not a topic of this research, the results obtained on such images were not considered in this experimental study.

4.1 Evaluated devices

We evaluated 22 smart mobile devices; 12 Android phones, 6 Android tablets, and 4 iOS devices. The list of the devices is presented in Table 1.

Table 1: Evaluated smart mobile devices, CPU type, version of operating system.

No	Device	CPU	OS
1	Samsung Nexus	v7a	4.1
2	Samsung Galaxy Note 2	v7a	4.1
3	Samsung Galaxy S4	v7a	4.2
4	Sony Xperia	v7a	2.3
5	Huawei U8850	v7a	2.3
6	HTC Desire HD A9191	v7a	2.3
7	Samsung Galaxy S3 mini	v7a	4.1
8	Samsung Galaxy S2 GT-I9100	v7a	4.1
9	Sony Xperia ST27i	v7a	2.3
10	HTC Sensation	v7a	4.0
11	HTC Wildfire	v6j	2.3
12	Samsung Galaxy Mini GT-S5570	v6	2.3
13	Samsung Galaxy Tab 10.1 GT-P7500	v7a	3.2
14	Asus TF300T	v7a	4.2
15	Samsung Galaxy Tab 10.1 GT-P7500	v7a	3.1
16	ASUS Transformer TF201	v7a	4.1
17	Samsung Galaxy Note 10	v7a	4.1
18	Samsung Galaxy Tab 2 7.0 GT-P3110	v7a	4.1
19	Apple iPhone 4	A4	6.1
20	Apple iPhone4S	A5	6.1
21	Apple iPhone 5	A6	6.0
22	Apple iPad 2	A5	6.1

4.2 Checkerboard detection

4.2.1 Error in corner detection. In the first experiment we compared the locations of the detected corners of the checkerboard pattern (in mm) with the reference (known) coordinates. The results obtained by processing the original images are presented in Fig. 2.

The average error was 0.34 mm (with st. deviation 0.35 mm) and the median error was 0.28 mm. The system was therefore able to very accurately detect the corners of the A4 paper sheet and to correctly estimate the homography. The top-left plot in Fig. 2 presents the error histogram; most of the errors are smaller than 0.5 mm, almost all of them are smaller than 1 mm.

The top-right plot presents the errors obtained with the individual devices; the device numbers correspond to the numbers listed in Table 1. With the exception of a couple of devices, the results do not differ to a large extent. One has to note that we have to be rather careful when interpreting these results. Although the images were taken from the approximately same directions on all devices, they were taken by different persons, with different vigilance in different illumination conditions, therefore a direct comparison between the results of different devices is not very appropriate.

The bottom-left plot presents the average error by image numbers (i.e., viewing angles) while the bottom-right plot depicts the average error for all 48 points (corners) on the checkerboard (listed from top left to bottom right). The latter plot indicates that the error in the middle of the checkerboard is smaller than the error at its edges, which could be caused by a different influence of the radial distortion.

We wanted to better check the role of the radial distortion, therefore we undistorted the images (i.e., the detected points) before estimating the homography and projecting the detected points in the world coordinate frame as described in Section 3. The results are shown in Fig. 3(a). The bars depict the mean error for all three types of input data: original images (*orig*), points corrected using specific (*undSpec*) and generic distortion coefficients (*undGen*). The red lines depict the error median. The first plot shows the mean error in localisation of pattern corners. The mean error slightly reduces when the distortion coefficients are considered.

Fig. 4 shows more detailed results obtained with the generic distortion coefficients. By comparing this figure with Fig. 2 we can see that the results improve for most of the devices. There are a few exceptions, however; the devices where the distortion coefficients deviate from the general trend, and they should be treated separately.

4.2.2 Error of estimated checkerboard width. The errors in estimating the checkerboard width are presented in Fig. 3(b). In this task the correction of the radial distortion proved to be very useful; the mean width error decreased from the 0.91 mm to 0.55 mm. This is probably due to the fact that two points between which we measured the distance lay on the edge of the reference object where the influence of the radial distortion is bigger. A very similar improvement of the results we obtained in both cases; when the specific as well when the generic distortion coefficients were used. This is rather a positive finding, since this means that the cameras on mobile devices are sufficiently similar that can be modelled using the same set of distortion coefficients.

4.3 Camera parameters

4.3.1 Image resolution Fig. 5 depicts image sizes that were used when capturing images with different devices. We can see that the resolution (width and height of images) considerably vary across devices. This is partially due to different resolution of the camera sensors. Additionally, we did not always use the maximal camera sensor size due to limitations of certain devices, where the ratio of the width versus the height must match with the selected camera preview size and with the actual captured picture size, otherwise the device would produce invalid picture or it would not produce the same picture as seen on the display (preview may have been cropped). Being restricted by those limitations in some cases we selected smaller picture sizes than maximal

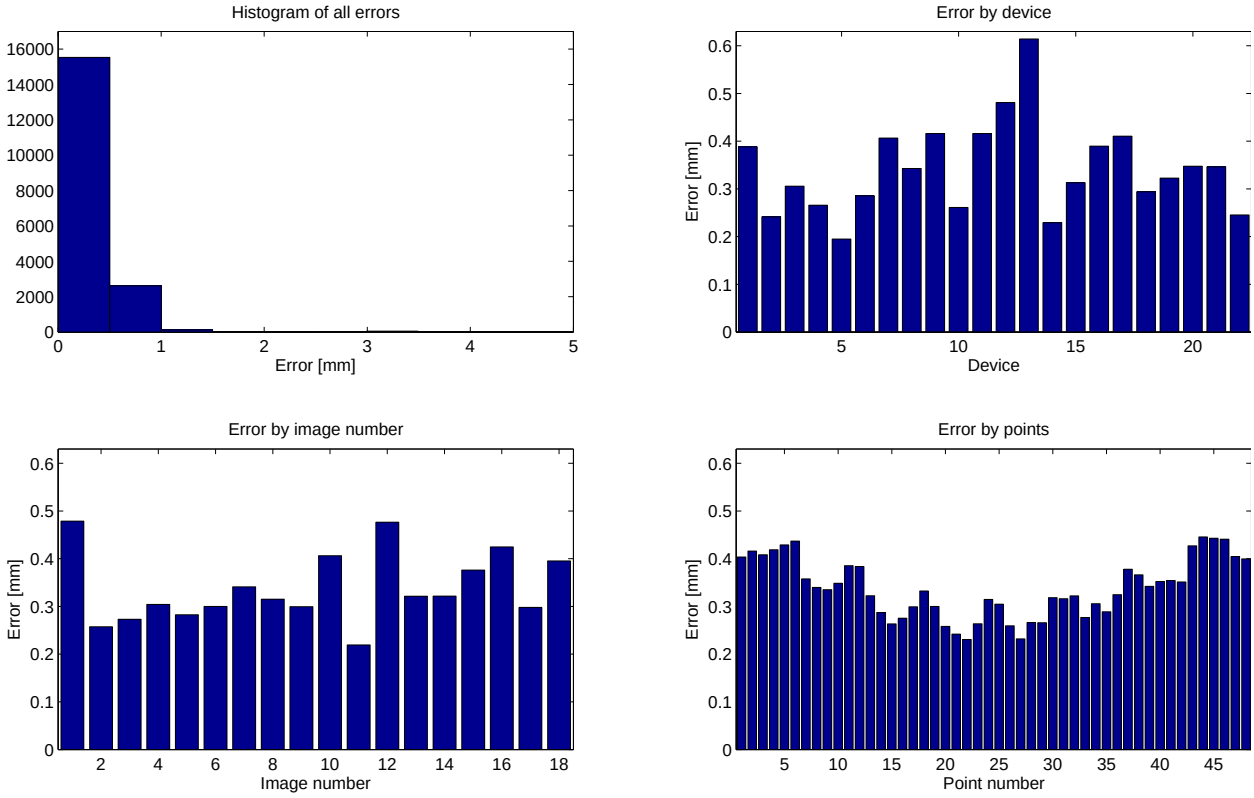
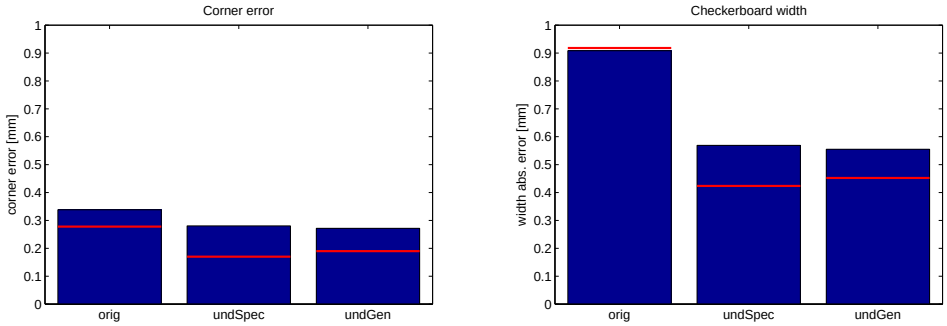


Figure 2: Results obtained on original images: histogram of all errors, error by device, error by image number and error by points.



(a) (b)

Figure 3: Results obtained by processing different types of images (i.e., points): (a) corner error, (b) checkerboard width error.

supported by the camera.

4.3.2 Intrinsic camera parameters. In this experiment we tried to estimate the intrinsic camera parameters from the information provided by the mobile device (as described in Section 2) and compare these parameters with the reference values obtained in the calibration process.

Fig. 6 depicts the results for the individual devices. The top-left plot compares the calibrated and the estimated focal lengths. We can see that the estimates are mostly sufficiently close to the reference values. A similar observation we can also make about the top-right plot, which shows the coordinates of the principal points.

The bottom-left plot shows the pixel ”squareness”, i.e. the ratio between the width and the height of the individual imaging element. Our assumption (that the elements are square) is mostly confirmed with the parameters obtained in the calibration process. We can see that squareness is almost always very close to 1; small deviations are probably caused by the imperfect calibration.

The bottom-right plot depicts the calculated ratios between the width and the height of the imaging sensor obtained by considering viewing angles as reported by the imaging device as well as the ratios obtained by considering the maximal resolution of the images as offered by the camera. We can see that at certain devices there is a considerable deviation, mainly due to not sufficiently accurate specification of the viewing angles.

Nevertheless, we can conclude that only by considering the data provided by the imaging device we can construct a reasonably credible camera matrix. However, it would be worth comparing the estimated focal length with the focal length obtained directly from the images by, e.g., considering the detected known four or five points in the reference object [5, 4].

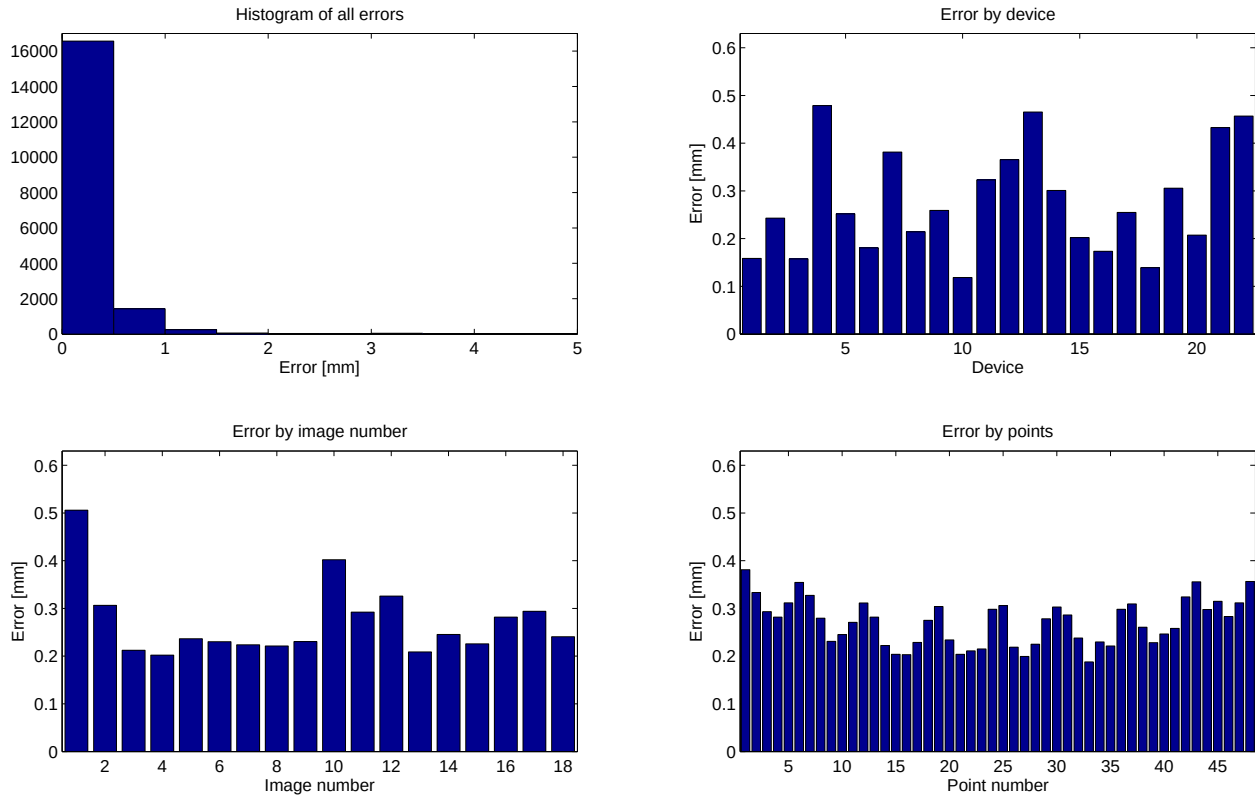


Figure 4: Results obtained on images undistorted with generic distortion coefficients: histogram of all errors, error by device, error by image number and error by points.

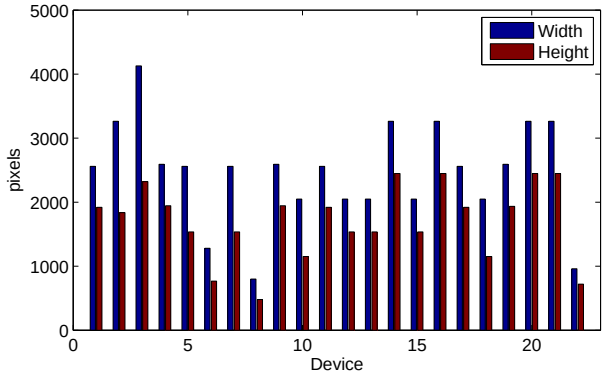


Figure 5: Image resolution that was used when capturing images.

4.3.3 Camera position In this experiment we compared the estimated camera positions (obtained as explained in Section 2) with the camera extrinsic parameters obtained in the calibration process. The overall mean results are presented in Fig. 7(a). The mean euclidian distance between the estimated and calibrated camera positions are presented, as well as the distances along the x, y, and z axes. We can see that the error is relatively small, since it does not exceed 1 cm along individual axis. Fig. 7(b) plots mean errors for every device. Except a few exceptions, there are no large deviations.

4.4 CPU performance

To conclude the experimental section, we present the results of CPU performance evaluation on the Androind smart mobile devices presented in Table 1. Rather than relying on many publicly available performance comparisons of mobile devices, we decided to do our own performance evaluation, using typical computer vision algorithms. Hence we developed a test application where we implemented the four standard computer vision algorithms as described in Section 3.3.

We measured processing times for each evaluation image and calculated the weighted means for all four algorithms. The weight was determined based on the size of the corresponding image. The results are expressed in terms of milliseconds necessary to process 1000 pixels. They are presented in Figure 8. As the processing time depends on the content of the image, the absolute values are not very informative. However, the relative comparison of these results indicates well the differences in the performance of the individual devices. As we can see two devices (no. 11 and 12) stand out from the rest. We can see from Table 1 that these are the only two devices that do not feature the *armeabi-v7a* architecture, but an older one. The other platforms achieved comparable results: a few exceptions apart, they were less than 2.5 times slower than the fastest. Based on these evaluation results we can deem the *armeabi-v7a* mobile architecture suitable for running relatively demanding machine vision application. The version of the Android operating system had no influence on the evaluation results.

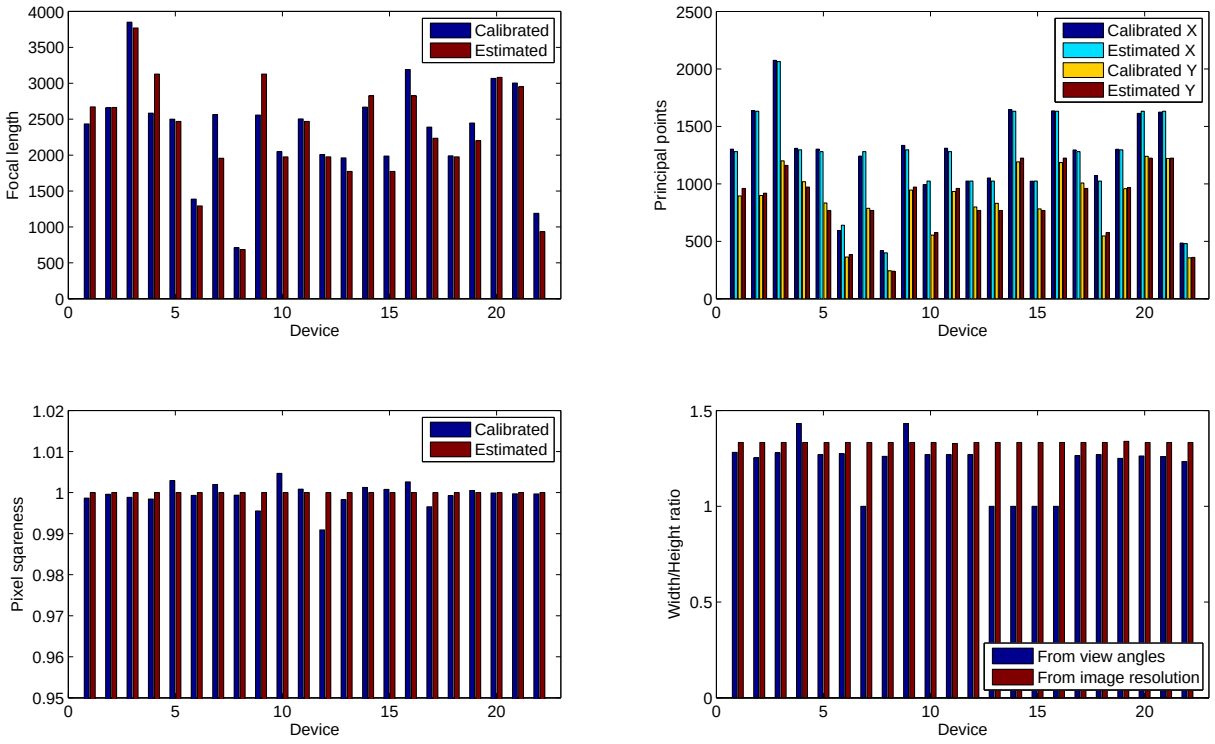


Figure 6: Intrinsic camera parameters. Comparison between parameters obtained with calibration and estimated from the factory settings. (a) Focal length in pixels. (b) Principal point. (c) Pixel squareness. (d) The ratio between the image width and height.

5 Conclusion

In this paper we have explored the feasibility of using a modern mobile phone or a tablet as an integrated device for performing visual measurements. We have constrained our scope of applications to flat-marker-based systems. Such a system uses a known reference object to estimate the camera parameters. To maximize the applicability we have chosen a widely-available reference object – the A4 paper sheet. We have developed a simple processing pipeline that detects the sheet of the paper. Using known dimensions of the sheet, we estimate the external parameters, while the camera internal parameters are constructed by reading the parameters provided by the device.

We have performed an in-depth analysis of accuracy of our mobile measuring system, using 22 different mobile devices. The results show that the error in detecting the corners of the checkerboard pattern is below 0.5 mm in all devices. Considering the size of the reference object, the relative error is below 0.2%. We have observed that the radial distortion is indeed significant when considering sub-milimeter measurements. A notable result is that generic distortion parameters (obtained from calibration results over all devices) reduce the errors induced by the radial distortion. This implies similar lens across the different phones. The analysis of approximation of internal parameters showed that the focal length is sufficiently well approximated by reading the viewing angle directly from the device, as well as the principal point is well approximated by the image center. The analysis of the extrinsic parameters estimation shows that homography-based approach introduces errors that are below 10 mm in all three coordinate axes, making the overall camera position error smaller than 15 mm. We have further measured the variability of the processing capabilities of the tested mobile devices by running some basic computer vision operations. Results show that the processing power does not prohibitively vary in this respect for the newer architectures, however, the processing is significantly slower for the older ones.

Our results show that modern smart phones and tablets can indeed be used as a high-precision measuring device even using a simple flat-caliber-based approach, achieving sub-milimeter accuracy in measurements in the reference object plane. An important result is also the fact that intrinsic camera parameters, along with radial distortion coefficients, may be estimated without specialized calibration procedures. This is especially important property as it significantly widens the application domain of mobile-phone-based measuring systems.

Our future work will extend in several directions. We will study accuracy of more advanced approaches for camera parameters estimation (e.g., [5, 4]) in our setting. Another direction of further research will be to conduct the feasibility study of high-precision measuring of 3D objects placed in the center of the sheet of paper.

References

[1] D.M. Chen, G. Baatz, K. Koser, S.S. Tsai, R. Vedantham, T. Pylvanainen, K. Roimela, Xin Chen, J. Bach, M. Pollefeys, B. Girod, and R. Grzeszczuk. City-scale landmark identification on

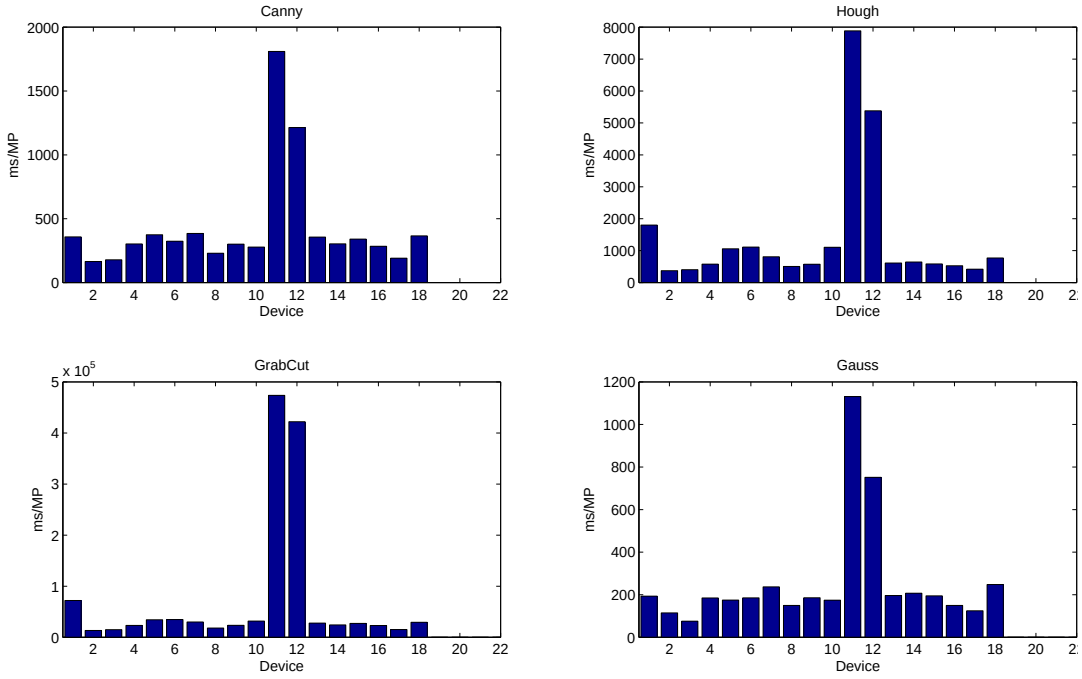


Figure 8: CPU benchmark results.

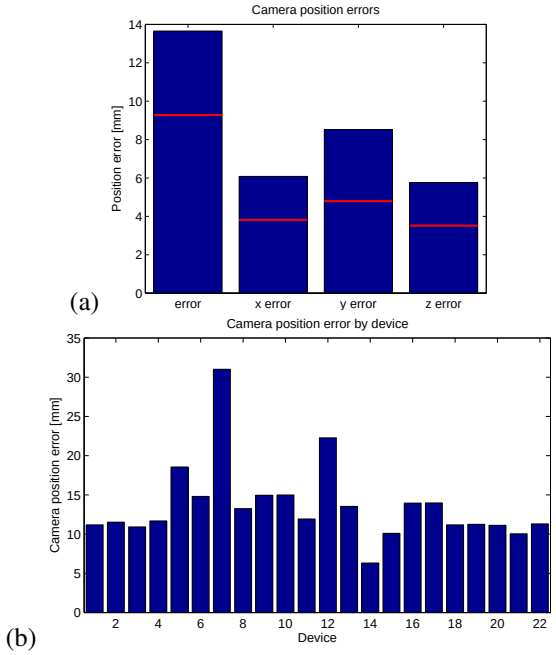


Figure 7: Camera position errors with respect to the positions obtained in the calibration process: (a) overall error, (b) errors of individual devices.

mobile devices. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 737–744, 2011.

[2] J. Clemons, Haishan Zhu, S. Savarese, and T. Austin. MEVBench: A mobile computer vision benchmarking suite. In *Workload Characterization (IISWC), 2011 IEEE International Symposium on*, pages 91–102, 2011.

[3] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[4] Z. Kukelova, M. Bujnak, and T. Pajdla. Real-time solution to the absolute pose problem with unknown radial distortion and focal length. In *International Conf. Comp. Vision*, 2013.

[5] Z. Kukelova, J. Heller, and T. Pajdla. Hand-eye calibration without hand orientation measurement using minimal solution. In *Computer Vision – ACCV 2012*, volume 7727 of *Lecture Notes in Computer Science*, pages 576–589, 2013.

[6] R. Manduchi, S. Kurniawan, and H. Bagherinia. Blind guidance using mobile computer vision: A usability study. In *Proceedings of the 12th International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '10, pages 241–242, 2010.

[7] L. Meier F. Camposeco Paulsen O. Saurer M. Pollefeys P. Tanskanen, K. Kolev. Live metric 3D reconstruction on mobile phones. In *Proc. of Int. Conf. on Computer Vision, ICCV 2013*, Sydney, 2013.

[8] Q. Pan, C. Arth, G. Reitmayr, E. Rosten, and T. Drummond. Rapid scene reconstruction on mobile phones from panoramic images. In *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, pages 55–64, 2011.

[9] M. Petter, V. Fragoso, M. Turk, and Charles Baur. Automatic text detection for mobile augmented reality translation. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 48–55, 2011.

[10] Guohui W., Yingen X., J. Yun, and J.R. Cavallaro. Accelerating computer vision algorithms using OpenCL framework on the mobile GPU - a case study. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 2629–2633, 2013.

[11] Y. Xin and C. Kwang-Ting. LDB: An ultra fast feature for scalable augmented reality on mobile devices. In *Mixed and Augmented Reality (ISMAR)*, pages 49–57, 2012.

Computer Vision Winter Workshop 2014, Zuzana Kúkelová and Jan Heller (eds.)
Křtiny, Czech Republic, February 3–5
Czech Society for Cybernetics and Informatics

Combining Structural and Statistical Approach to Online Recognition of Handwritten Mathematical Formulas

Jan Stria[1], Daniel Průša[1], and Václav Hlaváč[2]

[1]Center for Machine Perception, Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Czech Republic
{striajan,prusapa1}@cmp.felk.cvut.cz

[2]Department of Theoretical Computer Science and Mathematical Logic, Faculty of Mathematics and Physics, Charles University in Prague, Czech Republic
hlavac@cmp.felk.cvut.cz

Abstract *This paper introduces a novel method for online recognition of handwritten mathematical formulas. The method is based on the combination of a structural analysis with a statistical model specifying relations of individual symbols. A description of all recognition phases is given, focusing mostly on the structural analysis stage. The recognition process following a bottom-up manner is driven by a spatial grammar, which expresses mathematical formulas unambiguously. As the grammar describes the relative positions only roughly, a statistical model learned from a database of annotated formulas is employed to refine the description. Several experimental results are also reported.*

1 Introduction

Recognition of mathematical formulas has been widely studied in past years [1, 16]. There are two main motivations for dealing with it. Firstly, there is a need of interpreting mathematical expressions found in scanned documents including books and handwritten notes. In this case, the offline source data are formed by a set of images. On the other hand, we also may want to recognize formulas captured by a mouse, a tablet or a touch screen. Here we deal with online data in a form of strokes of points ordered in a time manner. The main motivation for online formulas recognition is the ability of entering mathematical expressions to a computer using handwriting. It is far more natural then utilizing an arbitrary existing format for math description (TEX, MathML). Once such an expression gets correctly interpreted, it can be pasted to an article, evaluated by a mathematical program or plotted as a graph. Recognition of online handwritten mathematical formulas is a subject of this paper.

Algorithms for formulas recognition are originally based on conventional methods for recognition of a simple text [9]. However, recognition of mathematical expressions is a more challenging task, as they usually comprise a rich spatial structure. For this reason, even the top performing recognizers are able to correctly recognize only approximately 2 out of 3 formulas [8]. So there is still much room for improvements.

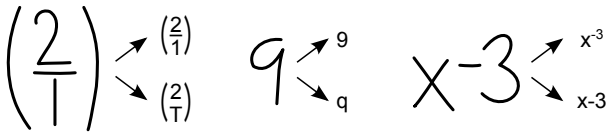


Figure 1: Ambiguities in segmentation (fraction vs. combinatorial number), symbol recognition (digit 9 vs. letter q) and structure (subtraction vs. power).

Online recognition of a handwritten mathematical formula can be usually separated into three relatively independent phases. In the first phase, the input strokes are segmented into groups potentially forming individual symbols. During the second phase, each symbol candidate is passed to a character recognizer. This is another source of errors and ambiguities, as it may be difficult to recognize the symbol properly. Finally in the third phase, spatial positions of the identified symbols are examined to reveal the whole expression. This is the last main source of ambiguities, as it may be difficult to decide what is the exact relation of two or more symbols. The sources of ambiguities are summarized in Figure 1.

There are two approaches of dealing with the presented ambiguities. The first approach finds the suboptimal solution in each of these phases [2, 4]. It identifies only one concrete segmentation of the strokes and interprets each group as one particular character. It is obvious that if this is done incorrectly then the formula cannot be successfully recognized. Thus we rather try to find all reasonable, possibly mutually conflicting, segmentations and we let each segment to be interpreted as various characters. The ambiguities are then solved during the third phase of recognition when the structure of a formula is analyzed [10, 11]. The general idea of this approach has been explicated as the structural construction paradigm in [12].

Quite common formalisms used to model the recursive character of mathematical formulas are constituted by various types of spatial grammars [3, 6]. Two-dimensional grammars are a quite powerful tool, however, they also bring a large challenge which is the computational complexity [5].

The planar placement of terminal units can easily cause that the worst-case parsing time grows exponentially (in contrast to the parsing of terminals in a linear sequence). The grammars thus require careful design and usage.

The main contribution of this paper is in four areas:

- We introduce a spatial grammar which describes mathematical formulas unambiguously. In contrast to previous approaches with productions which relate at most two nonterminals, our grammar contains production rules with many target terminal and nonterminal symbols. These more complex production rules enable more precise spatial analysis of symbols' positions.
- We present how to enhance capabilities of a spatial grammar by combining productions with statistical models refining the description of symbol alignments. The experiments prove that this leads to a significant increase of the correct rate of the structural analysis.
- We show that parsing of two-dimensional structures can be done efficiently. This is demonstrated by measuring time performance of the experiments.
- We further develop our recent work on structurally driven symbols segmentation. New methods for the creation of segmentation hypothesis are given.

The text is organized as follows. Section 2 emphasizes the most important components of our recognizer. Section 3 introduces the spatial grammar and describes how it is used to drive the structural analysis. Section 4 explains the statistical model of individual symbols relations. Section 5 provides experimental results achieved on databases of collected formulas. Finally, Section 6 summarizes presented ideas and proposes potential future improvements.

2 Overview

The input of the recognizer is formed by sequences of points in a plane, which were captured by a certain input device. The sequences are called strokes and they are ordered in a manner they were written. On the output, we provide the recognized formula in TEX and Presentation MathML format. The resulting formula is also rendered using a third-party library.

The recognizer was implemented as a web application. It can be seen in Figure 2. The application employs a HTML5 interface running in any modern web browser and a recognition engine implemented as a web service running at a server. The client part is capable of capturing the handwritten input, which is continuously being sent to the server. The received strokes are being recognized by the server in a background and their interpretation is sent back to the client. The interface is responsible for visualizing the recognized formula. The user is also allowed to erase previously written strokes or interactively correct misrecognized symbols which makes the process of recognition truly interactive. Moreover, correcting a few symbols usually leads to the successful recognition of the entered formula. More details about the web application can be found in [14].

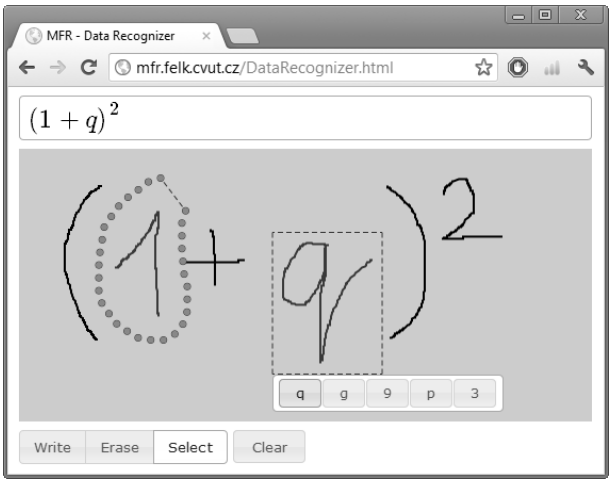


Figure 2: User interface of web application for mathematical formulas recognition. It provides a functionality of selecting and correcting a misrecognized symbol.

In the first phase of recognition, the input strokes are segmented into groups forming candidates for individual symbols. We allow one stroke to be a part of more symbol candidates. However, we allow one symbol to comprehend three strokes at most to avoid creation of too big groups. This constraint seems reasonable as perhaps no character requires more than three strokes when written in a standard way. The segmentation algorithm is based on finding two nearest neighbors of each stroke. Then each stroke forms one group by itself, two groups of two with each of its neighbors and one group of three with both of its neighbors.

The obtained groups of strokes are further checked to remove the non-perspective ones. The first heuristic algorithm relies on an empirically learned fact that strokes forming one symbol overlap horizontally or they are at least very close. The second heuristic rejects candidates whose strokes significantly intersect with strokes from other candidates. This rejects e.g. the horizontal and the vertical stroke in $+$ operator to form their own single-stroke groups.

The identified symbol candidates are then examined by two character recognizers. The first of them is the recognizer included in Microsoft Tablet PC platform. Its main advantage is the independence on the writing style. However, it is not able to handle special mathematical notation. Moreover, there is no way how to extend it. It grants up to ten characters for each candidate which are evaluated by a confidence as strong, intermediate or poor. These heuristics restrict the amount of symbol candidates significantly as shown in Figure 3.

The second recognizer is based on template matching. It searches for k-nearest classes of template symbols for each symbol candidate. One class corresponds to the unique character contained in the database of handwritten symbols, which was extracted from the database of annotated formulas. We store ten symbols per single class, i.e. character. To cover various writing styles of individual characters, all the extracted symbols belonging to the same class are hierarchically clustered and only one symbol is selected from each cluster. The distance used in the hierarchical clustering is

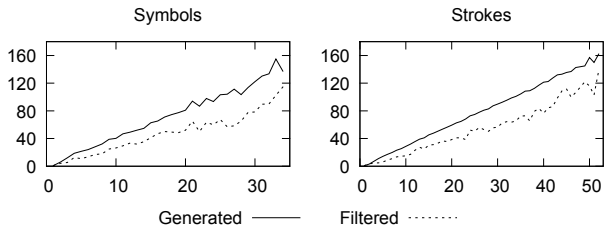


Figure 3: Average amounts of identified symbol candidates given the amount of individual symbols and strokes. Both amounts generated by the original segmentation algorithm and amounts after filtering are shown.

the same as the one used for finding the nearest classes. It is based on a so called elastic matching algorithm of the individual strokes where the optimal matching of strokes' points is found utilizing a dynamic programming approach [15].

The Tablet PC recognizer and our template matching recognizer are combined in the following way. Each symbol candidate is passed to the Tablet PC recognizer and the recognized characters are used as a filter for template matching. Moreover, characters unsupported by Tablet PC recognizer are also considered for template matching. This ensures that each symbol candidate is evaluated by the same algorithm and so there is no problem how to combine evaluations obtained from two different recognizers.

3 Structural analysis

Mathematical formulas can have very complex spatial structure. However, there are rigid constraints on a correctly formatted expression. These constraints can be naturally described using a 2D grammar. The grammar is defined in a separate text file as demonstrated on the following snippet:

```
Sum->[sum]|LowBound@B|UpBound@T|Expr@R
AddSub->AddSubOp|Term@L|Expr@R
AddSubOp->[+]
Expr->Sum
```

Each line corresponds to one production rule. There is a source nonterminal on the left side and a sequence of target nonterminal and terminal elements on the right side. The left side and the right side are separated by an arrow `->`. The elements on the right side are separated by vertical lines `|`. The terminals are enclosed in brackets and the nonterminals always start with a capital letter. In the first listed production, there is a nonterminal `Expr` representing an expression and a terminal `[sum]` representing a summation symbol. The terminals correspond to individual characters identified by the symbol recognizer. The nonterminals denote classes of terminals (e.g. `AddSubOp` or `Digit`) or logical parts of the formula (e.g. `Sum` or `Expr`) There is also a special axiom nonterminal `Formula` which represents a correct formula.

The productions are of two types. If there is only one target element on the right side, we speak about a 1-production (the last two lines of the snippet). By contrast, a n-production comprehends more than one target element on its right side (the first two lines). The 1-productions make it possible to maintain the grammar simple and readable. By contrast, the n-productions describe spatial relations of several target terminal and nonterminal elements, which form a more complex source nonterminal. The spatial relations of target elements are expressed relatively to the first target element, which is called a main element.

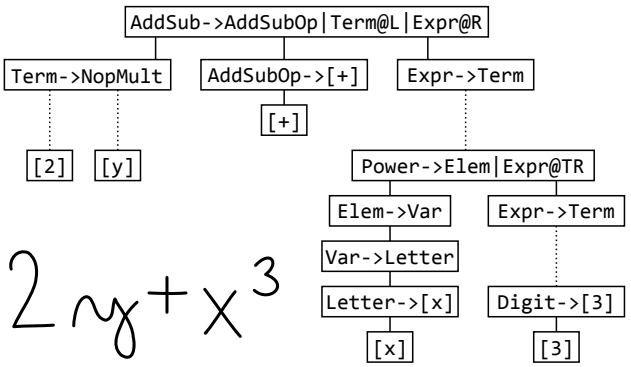


Figure 4: Handwritten formula interpreted as a derivation tree (with some nodes omitted for simplicity). Each node is assigned a terminal or a production. Nodes representing the same subset of strokes belong to the same segment.

Consider the first production describing a summation. The main element is a terminal `[sum]` denoting a summation symbol. The nonterminal `LowBound` expresses a lower bound of the summation. The suffix `@B` says that the lower bound is placed at the bottom of the main element. The `UpBound@T` expresses an upper bound of the summation placed at the top of the summation symbol. Totally, eight different relative positions are distinguished: left, right, top, bottom, top-left, top-right, bottom-right and inside.

The complete version of the grammar comprises approximately 200 terminals, 120 nonterminals and 370 productions. Approximately 100 of them are n-productions and 270 are 1-productions. There is also a reduced version of the grammar comprising approximately 100 terminals and 80 n-productions. The reduced grammar omits characters and constructs which are not supplied in the available databases of handwritten formulas. Both grammars describe the formulas unambiguously, i.e. for each formula there is only one possible description based on the grammar. The grammars were created manually by a human.

The proposed grammar drives the structural analysis of a handwritten formula, which consists of an iterated construction of derivation trees in a bottom-up manner. Figure 4 shows an example of such a derivation tree. The construction begins with an assignment of the appropriate terminals to symbols identified in the character recognition phase. Such symbols form leaf nodes of the derivation tree. All nodes sharing the same group of strokes are joined to a so called segment, i.e. derivation nodes belonging to one segment express various interpretations of that segment's strokes. Subsequently, 1-productions are repeatedly applied on each node, deriving new nodes having the original ones as their children. Consider that a node derived by 1-production belongs to the same segment as the original node.

The n-production nodes are derived by joining several existing nodes according to a certain n-production. The re-

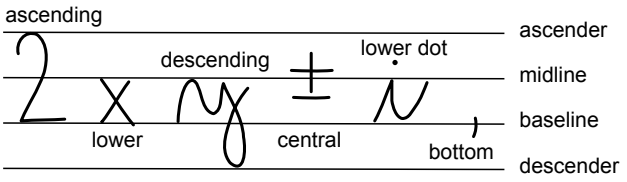


Figure 5: Various types of terminals based on their vertical positioning on four horizontal lines related to a writing line.

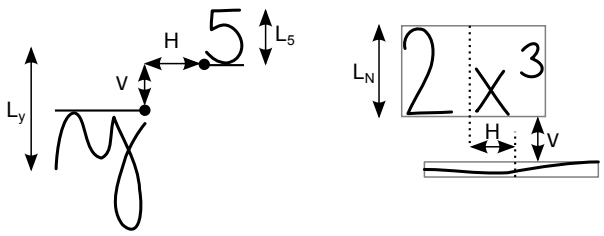


Figure 6: Relative distances and size of bind symbols. For a power y^5, the position mid segment of y and base segment of 5 are considered to determine their horizontal distance H and vertical distance V. The distance are normalized by the estimated line height L_y. The relative size is computed from L_5 and L_y. For the numerator $2x^3$ and the fraction line, their bounding rectangles are considered.

sulting node belongs to a segment created by joining segments of the original nodes. The original nodes become children of the derived node. There are several constraints on the original segments. First of all, their sets of strokes have to be pairwise disjoint because each stroke can participate only in a unique symbol of one derivation tree. Other constraints serve for reducing the amount of derived nodes. Based on the bounding rectangles of joint nodes and their relative positions, various polygons are constructed. These polygons are checked for an intersection with bounding rectangles of strokes not participating in the nodes. If the relative size of the intersection area is above a predefined threshold, the derivation is rejected.

The parsing algorithm is driven by the amount of strokes in nodes derived by n-productions. The nodes comprising k strokes are constructed in the k-th step. This systematic approach ensures derivation of more complex structures from less complex ones. It begins with derivation of nodes comprising two strokes and ends with nodes comprising all the input strokes. It should be obvious that many trees are constructed in parallel sharing common subtrees. Each derivation node is assigned a value. This value is equal to the belief obtained by the character recognizer for terminal nodes which correspond to individual symbols. For nodes derived by n-productions, the value is based on values of the child nodes and on their relative positions as described in Section 4. The derivation tree having the highest evaluated root node is chosen at the end of the parsing algorithm.

4 Statistical model

The productions describe relative positions of target terminal and nonterminal elements with respect to the main target element, as described in Section 3. However, the relative positions are only expressed as several discrete values (left, right, top, bottom, top-left, top-right, bottom-right and inside). To be able to define the spatial relations more precisely, we employ a statistical model of the expected positions. The model is extracted from the training database of 1500 handwritten formulas. Because some productions occur only sparsely in the database, it is not possible to store separate distributions for each of them. It is rather necessary to categorize derivation nodes to classes based on the properties of the involved terminals.

We distinguish various types of terminals (i.e. characters) based on their positioning on a writing line which is determined with respect to four horizontal lines: descender, baseline, midline and ascender. It can be seen in Figure 5. The ascending terminals (e.g. digit 2) are written between the baseline and ascender, the lower terminals (e.g. letter x) between the baseline and midline, the descending terminals (e.g. letter y) between the descender and midline, etc. The type of the terminal determines the base, mid and ascender segment for each terminal node. Not all segments can be determined for each terminal (e.g. there is only the mid segment for the operator $+$ because it is not aligned with the baseline or the ascender line). The vertical position of each segment is aligned with the appropriate horizontal line. The horizontal coordinates of the mid segment are coordinates of its strokes bounding rectangle. The horizontal coordinates of the base and ascender segments are determined based on the vertically local minimum and maximum of the strokes horizontal coordinates. If two of the segments are defined for the node at least, it is possible to estimate its line height.

The proposed segments and bounding rectangles of symbols are utilized to measure their relative horizontal and vertical distance and their size ratio. All these values are measured relatively to the estimated line height to deal with variously sized handwriting styles. The employed productions as well as the types of the terminals determinate which segments or bounding rectangles are considered. E.g. for the node representing a power y^5, we utilize the mid segment of the node y and the base segment of the node 5 to compute the horizontal distance H and the vertical distance V, as shown in Figure 6. The relative distances are computed by normalizing H and V with the estimated line height L_y. The ratio of the estimated line heights L_5 and L_y gives the relative height of the exponent 5 with respect to the base y. Figure 6 also shows how the bounding rectangles of a numerator and a fraction line are used to evaluate their distance. Here the vertical distance V of bounding rectangles and the horizontal distance H of bounding rectangles' centers are normalized by the estimated line height of the numerator L_N.

While evaluating nodes in left, right, top-left, top-right and bottom-right relative positions, only the segments of the neighboring side terminal nodes are considered. E.g. while evaluating positions in the addition $2y + x^3$ from Figure 4, we utilize node y which is the rightmost terminal in $2y$, node $+$, and node x which is the leftmost terminal in x^3. We call every sidemost terminal subnode of its parent node as a bind node, because it is responsible for binding its parent node to the other nodes. E.g. in the addition $2y + x^3$, the node 2

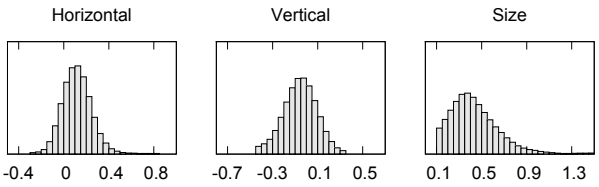


Figure 7: Examples of histograms of relative horizontal and vertical distance and relative size of a variable and its subscript. The histograms for various pairs of terminals and their relative positions form a statistical model.

is the left bind node and the node x is the right bind node because the addition cannot be bound via the exponent 3.

Totally, we distinguish 26 various categories, for which we evaluate the relative distances and the relative heights separately. The classification is based on the applied production and on the types of the terminal bind nodes. We store up to three distributions (relative horizontal and vertical distance and relative size) per each class as 1D histograms, as shown in Figure 7. It would be also possible to store the joint distribution for horizontal and vertical distance as 2D histogram. However, it would require more training data. Moreover, the relative horizontal and vertical distance are largely independent on each other, as the absolute value of the correlation coefficient is mostly around 0.1 and always under 0.2 for all 26 classes.

When evaluating a relative position of two nodes, we compute the appropriate horizontal and vertical distance and the height ratio. We also select the appropriate distributions among the aforementioned 26 classes. They give us beliefs for the computed relative values. These beliefs are combined linearly with evaluation of the individual child nodes to get the evaluation of the derived node. The derived node is constructed only if its value is over a predefined threshold. This strategy prevents deriving many poorly evaluated false nodes.

5 Experiments

To verify the proposed methods for online math recognition, we have tested them on databases of handwritten formulas containing rather challenging samples. The first of them is our own database called MfrDB which is publicly available [13]. It contains 2000 formulas, comprising 185 various expressions obtained from 232 users. The dataset was randomly split into 1500 training and 500 testing samples. The formulas were obtained using a web interface and then they were annotated semi-automatically. The semantics of formulas are described in an extended MathML format.

The second database is the one used in CROHME competitions. The CROHME 2012 [7] database contains 1366 training and 488 testing samples. Both training and testing samples are split into three parts based on the complexity of expressions (in a sense of average count of strokes per formula, number of various symbols included in formulas and complexity of a grammar describing formulas in a dataset). The latest CROHME 2013 [8] database contains 8836 training and 671 testing samples. They are even more complex than those included in CROHME 2012 database. Both training and testing samples are publicly available for CROHME 2012, however only training samples are available for CROHME 2013. That is why we do not use CROHME 2013 for a detailed evaluation.

Results	CROHME 2012			MfrDB
	Part I	Part II	Part III	
Symbol	67.6	67.8	65.6	67.3
Structure	**74.0**	**58.8**	**54.9**	**65.4**
Formula	9.7	5.8	3.9	5.2

Table 1: Recognition results on testing samples of CROHME 2012 and MfrDB databases. First row presents accuracy of isolated symbol classification. Second row gives independent performance of structural analysis given perfect segmentation and symbol classification. The third row shows rates of completely recognized formulas.

System	2012	2013		
	0 err.	0 err.	1 err.	3 err.
MFR (our system)	**3.9**	**2.7**	9.7	20.7
Tokyo Univ.	24.0	20.0	34.1	42.9
Univ. of São Paulo	6.4	9.4	18.5	27.3
Univ. of Valencia	24.9	**23.4**	37.9	47.8
Tech. Inst. Rochester	12.6	14.3	24.7	36.2
Sabanaci Univ.	11.9	8.4	19.1	26.2
Vision Objects	**70.2**	**60.4**	80.3	86.1
Univ. of Nantes	**32.9**	18.3	32.0	42.9

Table 2: Results of CROHME 2013 competition for all 8 participants. For the previous 2012 dataset, only the overall accuracy expressing rates of completely correctly recognized formulas is stated. For the latest 2013 dataset, also rates of formulas recognized with at most 1 and 3 errors are stated.

To train our system, we have used 1500 training samples from MfrDB and 8836 training samples from CROHME 2013. The experiments were performed on three CROHME 2012 testing datasets comprising 108, 301 and 488 formulas (part II comprises all formulas from part I and part III comprises both parts I and II), as well as on 500 testing formulas from MfrDB.

The recognition results are shown in Table 1. The first row shows rates of correctly recognized symbols in all three CROHME 2012 testing datasets and in MfrDB testing dataset. The symbols were extracted from formulas contained in the datasets and classified independently on their context in formulas. The symbol is considered to be correctly recognized if the symbol recognizer is able to assign a correct label to its strokes. The third row of Table 1 presents overall rates of fully recognized formulas. The formula is considered to be fully correctly recognized if the resulting derivation tree corresponds to the annotation and all the strokes are correctly assigned to their symbols. We have also tested the structural analyzer independently on the

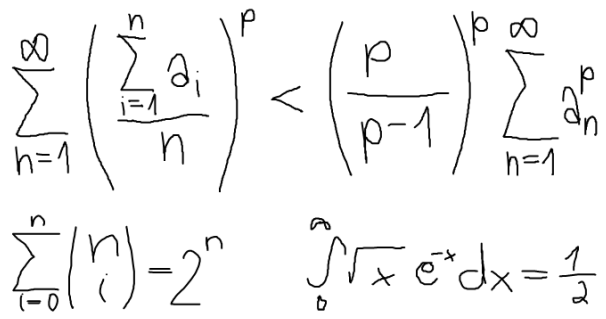


Figure 8: Examples of formulas which were correctly parsed by our structural analyzer.

Figure 9: Example of the same handwritten formula written by two various users. The left one was parsed successfully as $e^{i\pi} + 1 = 0$. The right one was misinterpreted as $e^{i^{\pi}+1} = 0$.

strokes segmentation and symbol recognition. To accomplish that we have assigned correct labels to all actual symbols. It simulates a case of having perfectly identified and recognized symbols. The results of structural analysis are presented in the second row of Table 1. Some of the correctly and incorrectly parsed formulas are shown in Figure 8 and Figure 9.

Table 2 presents results from the most recent CROHME 2013 competition. Totally 8 systems including ours were evaluated on part III of CROHME 2012 database and on CROHME 2013 database. Ratios of completely recognized formulas were published for both databases. For the latter also ratios of formulas recognized with 1 and 3 errors were published [8]. There were two winners of CROHME 2013 competition: non-commercial system developed at University of Valencia and commercial software by Vision Objects company. Both of them outperform our system by an order of a magnitude.

The main reason for a poor overall recognition rate of our system is the symbol classifier. It is only able to recognize approximately $2/3$ of all symbols compared to 85–98 % accomplished by other systems [7]. Unfortunately, one unrecognized symbol makes it impossible to recognize the whole formula correctly.

On the other hand, the structural analysis gives very promising results. We believe that an improvement of the symbol recognizer will make the overall results more satisfying. By that time, the user of our application can interactively select the misrecognized symbols and correct them manually while using the web application [14].

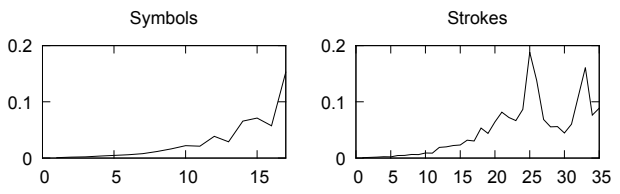


Figure 10: Performance of the structural analyzer expressed as an average time (in seconds) spent on analyzing formulas comprising the specified amount of symbols and strokes. The times on the vertical axes are measured in seconds.

Figure 10 shows an average parsing time spent on analyzing formulas comprising the specified amount of symbols. It was evaluated on a notebook with Intel Core i5 M430 2.27 GHz processor and 8 GB RAM. It can be seen that the results are provided in real time, almost immediately. This proves that although the parsing algorithm has theoretically an exponential complexity, it can be used in practice thanks to the described restrictions on the derived nodes.

6 Conclusion

We have introduced an algorithm for the structural analysis of mathematical formulas. It is driven by a rigid description of formulas employing an unambiguous spatial grammar. However, it also utilizes a statistical model of the expected mutual relative positions of individual symbols. The introduced probability distributions give natural evaluations of derivation nodes. We have tested the proposed method on two independent databases and we obtained promising results.

In future, we would like to focus on the first two phases of online handwritten math recognition. The amount of the symbol candidates generated in the segmentation phase should be reduced, omitting the non-perspective candidates which obviously do not form a symbol. In the symbol recognition phase, amount of the recognized alternative characters per one symbol candidate should be also reduced. Besides, we would like to improve the character recognition itself, perhaps by replacing the template matching algorithm by a better performing one.

Acknowledgement

The first author was supported by the EC project FP7-288553 CloPeMa. The first and second author were supported by the Grant Agency of the Czech Technical University under the project SGS13/205/OHK3/3T/13. The third author was supported by the Grant Agency of the Czech Republic under the project P202/10/1333.

References

[1] Kam-Fai Chan and Dit-Yan Yeung. Mathematical expression recognition: A survey. *International Journal on Document Analysis and Recognition*, 3(1):3–15, 2000.

[2] T. Kanahori, K. Tabata, W. Cong, F. Tamari, and M. Suzuki. On-line recognition of mathematical expressions using automatic rewriting method. In *Proc. 3rd International Conference on Advances in Multimodal Interfaces (ICMI 2000)*, pages 394–401, Beijing, China, 2000.

[3] Stéphane Lavirotte and Loïc Pottier. Mathematical formula recognition using graph grammar. In *Proc. SPIE on Document Recognition*, pages 44–52, San Jose, California, USA, 1998.

[4] Chuanjun Li, Timothy S. Miller, Robert C. Zeleznik, and Joseph J. LaViola. Online recognition of handwritten mathematical expressions with support for matrices. In *Proc. 19th International Conference on Pattern Recognition (ICPR 2008)*, pages 1–4, Tampa, Florida, USA, 2008.

[5] Percy Liang, Mukund Narasimhan, Michael Shilman, and Paul Viola. Efficient geometric algorithms for parsing in two dimensions. In *Proc. 8th International Conference on Document Analysis and Recognition (ICDAR 2005)*, pages 1172–1177, Seoul, Korea, 2005.

[6] Erik G. Miller and Paul A. Viola. Ambiguity and constraint in mathematical expression recognition. In *Proc. 15th National Conference on Artificial Intelligence (AAAI 1998)*, pages 784–791, Madison, Wisconsin, USA, 1998.

[7] Harold Mouchère, Christian Viard-Gaudin, Dae Hwan Kim, Jin Hyung Kim, and Utpal Garain. ICFHR 2012 — competition on recognition of online mathematical expressions (CROHME2012). In *Proc. 13th International Conference on Frontiers in Handwriting Recognition (ICFHR 2012)*, pages 1497–1500, Bari, Italy, 2012.

[8] Harold Mouchère, Christian Viard-Gaudin, Richard Zanibbi, Utpal Garain, Dae Hwan Kim, and Jin Hyung Kim. ICDAR 2013 CROHME: Third international competition on recognition of online handwritten mathematical expressions. In *Proc. 12th International Conference on Document Analysis and Recognition (ICDAR 2013)*, pages 1460–1464, Washington, D.C., USA, 2013.

[9] Réjean Plamondon and Sargur N. Srihari. On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):63–84, 2000.

[10] Daniel Průša and Václav Hlaváč. Mathematical formulae recognition using 2D grammars. In *Proc. 9th International Conference on Document Analysis and Recognition (ICDAR 2007)*, pages 849–853, Curitiba, Brazil, 2007.

[11] Daniel Průša and Václav Hlaváč. Structural construction for on-line mathematical formulae recognition. In *Proc. 13th Iberoamerican Congress on Pattern Recognition (CIARP 2008)*, pages 317–324, Havana, Cuba, 2008.

[12] M. I. Schlesinger and Václav Hlaváč. *Ten Lectures on Statistical and Structural Pattern Recognition*. Springer, Berlin, Germany, 2012.

[13] Jan Stria, Martin Bresler, Daniel Průša, and Václav Hlaváč. MfrDB: Database of annotated on-line mathematical formulae. In *Proc. 13th International Conference on Frontiers in Handwriting Recognition (ICFHR 2012)*, pages 540–545, Bari, Italy, 2012.

[14] Jan Stria and Daniel Průša. Web application for recognition of mathematical formulas. In *Proc. 11th Conference on Theory and Practice of Information Technologies (ITAT 2011)*, pages 47–54, Vrátna dolina, Slovak Republic, 2011.

[15] Seiichi Uchida and Hiroaki Sakoe. A survey of elastic matching techniques for handwritten character recognition. *IEICE Transactions on Information and Systems*, E88-D(8):1781–1790, 2005.

[16] Richard Zanibbi and Dorothea Blostein. Recognition and retrieval of mathematical expressions. *International Journal on Document Analysis and Recognition*, 15(4):331–357, 2012.

Computer Vision Winter Workshop 2014, Zuzana Kúkelová and Jan Heller (eds.)
Křtiny, Czech Republic, February 3–5
Czech Society for Cybernetics and Informatics

Using discriminative analysis for improving hierarchical compositional models

Domen Tabernik[1], Matej Kristan[1], Marko Boben[1], and Aleš Leonardis[1,2]

[1]Faculty of Computer and Information Science, University of Ljubljana, Slovenia
[2]CN-CR Centre, School of Computer Science, University of Birmingham
{domen.tabernik,matej.kristan,marko.boben,ales.leonardis}@fri.uni-lj.si

Abstract *In this paper we propose a method to extract discriminative information from a generative model produced by a compositional hierarchical approach. We present discriminative information as a score computed from a weighted summation of the activation vector. We base the activation vector on individual activations of features from a parse tree of the detection. We utilize the score to reduce false positive detections by removing generative models with poor discriminative information from the vocabulary and by thresholding the detections with low discriminative score. We evaluate our approach on the ETHZ Shape Classes database where we show a reduction in the number of false positives and a decrees in detection time without reducing the detection rate.*

1 Introduction

In the field of the visual object class detection, many complex descriptors have been proposed that produce excellent results, e.g. [12], but such descriptors are computationally expensive. This problem becomes even more apparent in the combination with sliding windows therefore additional preprocessing steps are required to eliminate as many irrelevant windows as possible before applying complex descriptors. Alexe et. al. have addressed this issue to some extent but there are other alternatives such as hierarchical methods [13, 9, 4, 5, 8] with their generative models that do not require any preprocessing at all. In this case, each visual category is represented by multiple object models that cover visual variability within a category. An object model is a composition, consisting of increasingly complex features, that captures the complexity of objects through hierarchical arrangement of features. Due to hierarchical nature of object models each detection is a parse tree of features. One benefit of hierarchical arrangement is shareability of simple features across different visual categories. Common and simpler features are formed in the lower layers of the hierarchy and can then be used by the higher layers as needed for specific categories. This allows for more compact representation of objects on the one hand and on the other hand brings efficient scalability when introducing more categories, since detected simple features for one category can immediately be re-used by any other category.

Having shearability is one important benefit that in some hierarchical approaches originates from generativeness of models. However, the generative nature of construction leads to models that tend to overgeneralize and may not capitalize on discriminative information. For instance, a generative model of a horse must capture different variability of a horse seen from different viewpoints. We may be able to capture this by using multiple object models for different viewpoints, but models will still have to capture variability among different configurations of head and legs positions, while at the same time encompassing the variability between different breeds of horses. Generative models will be able to capture such variations but they will also capture representation of many other categories, especially visually similar ones. This can quickly lead to over-generalization. In the case of a horse the model will frequently misidentify a cow for a horse as both categories are similar. Additionally, we have also observed that some object models frequently hallucinate on the background objects that have low visual similarities with the detected object. The effect is most prominent on highly textured backgrounds. This allows a generative model to find enough features for a (false) positive match but at the same time the model does not capitalize on the discriminative features present in the representation that could point to an absence of detected object. Not considering such discriminative information in the end introduces noise into the detection process and can significantly reduce performance of the detector.

This problem was partially addressed in our previous work. In [10], we introduced a global HoC descriptor and included the discrimination of categories through non-linear support vector machine. As shown in [11], using such an approach as an additional hypothesis verification step applied after the detection stage, produces excellent results in the *ETHZ Shape Classes* dataset but adding non-linear decision boundary breaks the hierarchical approach and introduces computational complexity due to expensive computation of non-linear support vectors.

In [6], we eliminated the need for a non-linear SVM which allowed for principled integration into the concept of hierarchies. Using a sparse logistic regression we searched for weights to differentiate between two categories and used a score from weights to re-score the final detection. But the method still relied upon a global descriptor computed from histograms of activations of parse within detected bounding box. This constrained us to a discriminative information of a whole category and prevented us from obtaining discriminative information for specific object model. At the same

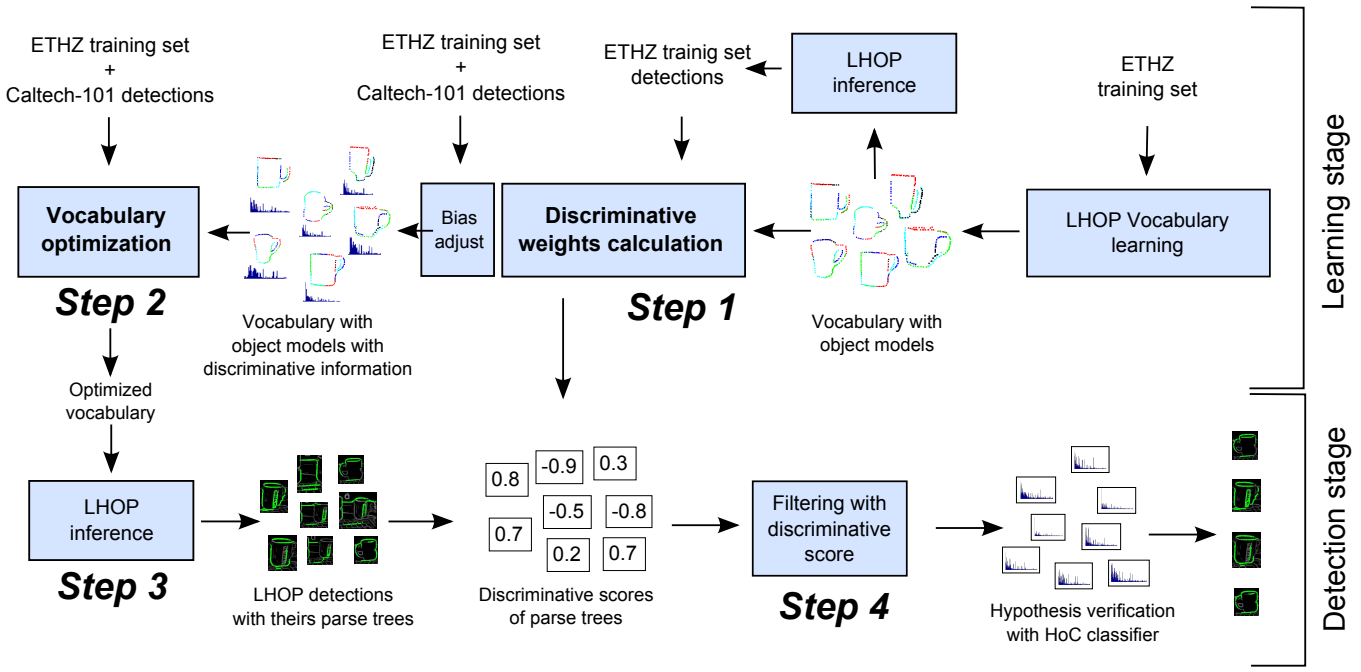


Figure 1: Overview of our method in learning and detection stage. In step 1 we extract discriminative information in form of weights, in step 2 we optimize vocabulary by removing object parts with poor discriminative information, in step 3 we produce detections with optimized vocabulary and in step 4 we filter out detections with low discriminative score.

time global descriptor may have also removed parts of spatial information important for the discrimination.

In this paper, we focus on finding discriminative information within a category by analyzing each object model. Our contribution can be divided into two parts: (i) introduction of discriminative information extracted from activation vectors of detection parse tree and (ii) vocabulary optimization of the object models. We introduce discriminative information in form of linear combination of weights applied to the activation vector of the parse tree. Weights are computed for each object model separately which slows to fully capture spatial information of the activated features of the parse tree and get better precision in identifying features that are responsible for the discrimination. With the vocabulary optimization we eliminate object models that have poor discriminative capabilities and often activate on false positive detections. As a last step we also perform hypothesis verification with the HoC descriptor to re-score detections.

The benefits of our approach are two-fold. We eliminate many false positive detections through vocabulary optimization by removing non-discriminative object models. False positives are also removed by thresholding the weighted combination of parse tree response values and removing detections with low discriminative score. The second benefit is detection speedup that comes with the reduced number of detections. This translates into less computation time spent in the hypothesis verification step with the non-linear HoC classifier and less computation time in the inference stage. Additionally, having less positive detections also slightly increases the final detection rate. As in both [11] and [6] we apply our solution to the learnt-hierarchy-of-parts (LHOP) model [4].

The remainder of the paper is structured as follows. In Section 2 we introduce discriminative node of whole parse tree and provide further details of how we integrate discriminative information into vocabulary optimization. In Section 3 we present our evaluation procedure with final results and we conclude with the discussion in Section 4.

2 LHOP with discriminative information

We introduce discriminative information and vocabulary optimization into the learnt-hierarchy-of-parts process as shown in Figure 1. In the learning stage we start with the learned vocabulary containing object models and extract detections of object models in a form of parse tree. For each object model we learn linear classifier on positive and negative activation vectors of the detections and obtain weights used in the filtering stage of the detection. This process is detailed in Section 2.2. The optimization process, detailed in Section 2.3, is then applied to the vocabulary containing object models with their corresponding weights to obtain a smaller vocabulary with removed object models that produce too many false detections.

The detection stage starts with inference of the image with the optimized vocabulary. This produces a set of detections for each image together with its parse tree containing activation parts. Based on detected object model we apply appropriate weights to the activation vector of each detection and calculate its discriminative score. We perform filtering based on the score and remove as many false positive detections as possible. Finally, we calculate HoC descriptor from each detection and compute its final score by applying HoC classifier to it.

2.1 Learnt hierarchy of parts

We first provide a notation for LHOP model and refer the reader to [4] for further details. In the following we will denote the vocabulary of hierarchical parts trained for up to L layers as a set of N compositions $\mathcal{L} = \{P_i^l\}_{i=1:N}$, where P_i^l is an identifier of i-th composition and belongs to the l-th layer of the vocabulary. At the last layer L each composition directly identifies one trained category, i.e. for each category we have only one corresponding composition on the L-th layer. We also define a set of $Links(P_i^l)$ which holds a list of linked sub-compositions on $l-1$ layer for the P_i^l part:

$$Links(P_i^l) = \{(ind_j, P_j^{l-1})\}_{j=1..num\_subparts(P_i^l)},$$

where P_j^{l-1} is the linked sub-composition type and ind_j is the local index number for this sub-compositional (ind_j goes through 0 and $num\_subparts(P_i^l) - 1$). For category layer parts (L-th layer), this subset holds object parts $(ind_o, P_o^{L-1}) \in Links(P_i^L)$ where each object part models a different view or an object of a category associated with the P_i^L part.

Applying the vocabulary $\mathcal{L}$ on a given image $\mathcal{I}$, the algorithm of hierarchical models infers a set of K detected parts, $\mathcal{C}(\mathcal{I}, \mathcal{L})$,

$$\mathcal{C}(\mathcal{I}, \mathcal{L}) = \{\pi_k^l\}_{k=1:K},$$

where the k-th detected part on the l-th layer $\pi_k^l = [P_k, \mathbf{c}_{\pi_k}, \lambda_k]$ is defined by its vocabulary identifier P_k^l, its location $\mathbf{c}_{\pi_k}$ in the image and its response values λ_k. All the inferred parts of the last layer L directly correspond to detected objects in the image:

$$\mathcal{D}(\mathcal{I}, \mathcal{L}) = \{\pi_j^L\}_{j=1:J},$$

where $\mathcal{D}(\mathcal{I}, \mathcal{L})$ is a set of J detected objects in the image $\mathcal{I}$ processed with the vocabulary $\mathcal{L}$. We also define a set of links $\Lambda(\pi_k^l)$ with a list of subparts for π_k^l defined on previous layer $l-1$:

$$\Lambda(\pi_k^l) = \{(off_p, ind_p, \pi_p^{l-1}))\}_{P=1..num\_subparts(\pi_k^l)},$$

where π_p^{l-1} is the linked subpart, off_p is offset location (x, y) relative to π_k^l and ind_p is the index matching to the corresponding sub-composition in $Links(P_p^l)$. For a detected category this always corresponds to one subpart representing detected object part $(off_p, ind_p, \pi_p^{L-1}) = \Lambda(\pi_k^L)$. Note, that we can have multiple detections of the same category in an image but they will have different location $\mathbf{c}_{\pi_k}$.

While each detected category part is defined the same as detected part at L-th layer $\pi_j^L = [P_j^L, \mathbf{c}_{\pi_j}, \lambda_j]$, we can also add a category information since a vocabulary identifier P_j^L from the L-th layer always directly matches to one learning category. We can use $\Lambda(\pi_k^L)$ to recursively obtain list of all subparts for π_k^L on all layers. List of all subparts obtained this way is called *an inferred parse tree*, while using the $Links(P_i^l)$ in the same recursive process would yield a list of all recursively traced compositions of vocabulary called *a vocabulary parse tree.* Minimal and maximal locations of all traced sub-parts in inferred parse tree define a bounding box of detected image. We can therefore define a set of detected objects from a given image $\mathcal{I}$ as:

$$\mathcal{D} = \{(\pi_j^L, c_j, r_j)\}_{j=1:J},$$

where c_j is detected category and $r_j = (x, y, w, h)$ is a detection bounding box.

2.2 Discriminative interpretation of parse tree

We perform analysis on a set of training images $\mathcal{I} \in training\_set$ inferred with the initial vocabulary $\mathcal{L}$. This gives us a set of detections $\mathcal{D}(\mathcal{I}, \mathcal{L})$ with their corresponding parse trees for each image $\mathcal{I}$. We focus on analyzing each object model P_o^{L-1} individually and extract only detections for that specific model:

$$\mathcal{D}_{P_o^{L-1}} = \{\ (\pi_j^L, c_j, r_j) \quad | \quad [P_o^{L-1}, \mathbf{c}_{\pi_o}, \lambda_o] = \Lambda(\pi_j^L)\ \}.$$

From each parse tree of detection $\pi_j^L \in \mathcal{D}_{P_o^{L-1}}$ we construct a discriminative descriptor h_j. We define h_j as activation vector where components of the vector correspond to the activated parts in the parse tree. We collect all activation parts from an inferred parse tree and create activation vector using parts response values. Additionally, we also include relative offset location of each activated part relative to its parent into the activation vector. Based on discriminative descriptor h_j of size G we calculate a discriminative score using a weighted summation:

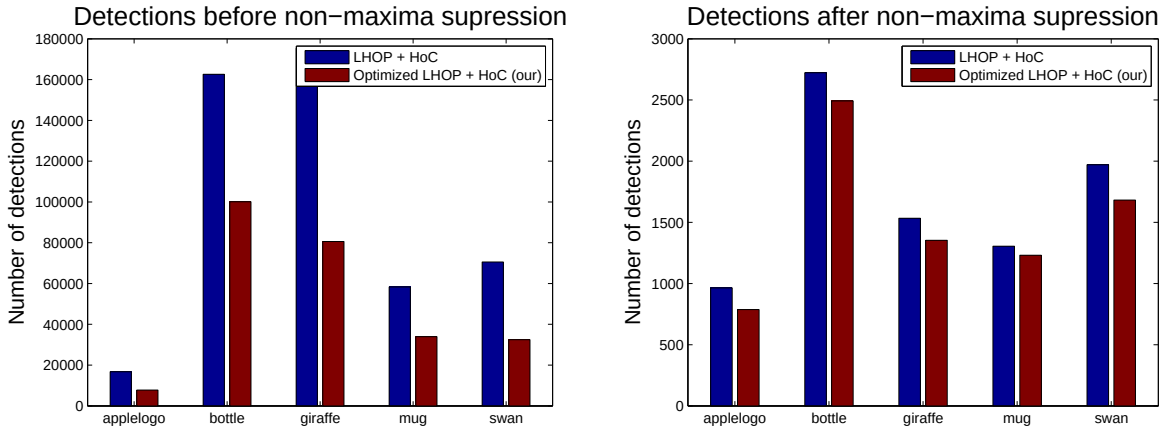
$$f(h_j; \Theta_o) = \sum_{g=1}^{G} \theta_o^{(g)} h_j^{(g)} + \theta_o^{(0)},$$

where $\Theta_o = [\theta_o^{(0)}, ..., \theta_o^{(G)}]$ is a vector of weights defining linear hyperplane between model object P_o^{L-1} and a background clutter or any other model. Calculated discriminative score $f(h_j; \Theta_o)$ is later used in filtering process during the detection stage to retain only detections with high score.

We estimate the weights Θ_o via a linear Support Vector Machine from positive and negative detections of object model P_o^{L-1}. Additionally, we adjust bias by changing $\theta_o^{(0)}$ to eliminate as many potential false detections in the filtering stage as possible.

2.3 Vocabulary optimization

We start the vocabulary optimization process by assessing how well each object model P_o^{L-1} performs on our training dataset. Using discriminative descriptor h_j we obtain a discriminative scoring from a parse tree of all detections of object model P_o^{L-1} and evaluate it according to its groundtruth data. We remove an object model if it does not contribute to the detection rate and produce mostly false positive detections. The optimization process tries to minimize the number of object models while retaining as high score as possible. The optimization is performed using a greedy approach with the following procedure: we start with an empty vocabulary and simulate individually adding every object model P_o^{L-1} to the current vocabulary separately and calculate a performance score with added candidate model. Performance score is measured as an area under the ROC curve



(a) (b)

Figure 2: Number of detections shown for **original vocabulary (blue)** and **our method (red)** with left figure (2a) showing detections before any non-maxima suppression and right figure (2b) showing detections after non-maxima suppression. We used HoC scoring in non-maxima suppression.

score (AUC) over all training images but only counting detections from current vocabulary objects plus the current candidate object model. Out of all possible candidates we add only the model which improves performance score the most and repeat the procedure until performance cannot be improved any more. Using this iterative process we produce optimized vocabulary $\mathcal{L}_{opt}$. With current optimization we only remove parts from the $L-1$ layer and fix L-th layer to account for now missing parts, while we leave parts on other layers unchanged.

2.4 Hypothesis verification with HoC

Using optimized vocabulary we perform another inference of training images and calculate the HoC descriptor for every detected object $\mathcal{D}_{opt}(\mathcal{I}, \mathcal{L}_{opt})$. From detection we obtain category information c_j and detected bounding box r_j and within this bounding box we calculate a HoC descriptor $\mathcal{H}_j$ from all activated parts of second and third layer. We calculate HoC using second and third layer of the vocabulary. All computed descriptors $\mathcal{H}_j$ are then re-scored by a non-linear Support Vector Machine for a category model c_j. We used libSVM [1] with an RBF kernel and χ^2 distance function. As the final step we perform a non-maxima suppression.

3 Experiments and results

We evaluated our method on the *ETHZ Shape Classes [2]* dataset with the same procedure as in [11]. Each category was evaluated independently and experiments were repeated five times to account for randomness in selecting training examples. Single experiment for each category was performed as follows. As training set we randomly selected half of the images that contained evaluated category and used as testing set used the other half combined with other images that did not contain any objects of the evaluated category. Similarly as in [11], we trained LHOP vocabulary $\mathcal{L}$ up to sixth layer, where 6th layer parts represent a category models and 5th layer parts represent an object models. We then ran an inference process with the LHOP vocabulary $\mathcal{L}$ on training images only and produced a set of initial training detections. Before the inference we also resized all the images by a factor of 1.2 and then started scaling them by a factor of $\sqrt{2}$ to produce around 4-7 scales per image.

3.1 Learning discriminative weights

From *ETHZ Shape Classes* training set we learn weights Θ_o for each object model P_o^5 from initial vocabulary. We first produce detections with the initial vocabulary $\mathcal{L}$ on all training images. All detections with the overlap (using PASCAL intersection/union function) of less than 0.3 are classified as negative while all detections with overlap over 0.7 are classified as positives. Any other detection that falls in between is classified as undefined and we avoid using them. From positive and negative detections we create activation vector h_j and use linear libSVM [1] to obtain final weights Θ_o. We additionally adjust bias for discriminative weights by performing detections on independent set of negative images. We randomly select 20% of images from Caltech-101 as independent set of images. In the detection stage weights are used to calculate discriminative score $f(h_j; \Theta_o)$ of each detection and any detection with score below threshold value of zero is filtered out.

3.2 Vocabulary optimization

In the vocabulary optimization step we used initial vocabulary $\mathcal{L}$ augmented with the discriminative weights for each object model and ran optimization process to produce optimized vocabulary $\mathcal{L}_{opt}$. As training set we used detections from all training images of *ETHZ Shape Classes* dataset for the specific category plus additional hard-negative detections from an independent set of images. The same criteria of 0.3 and 0.7 overlap threshold is used to select positive and negative detections. Similarly as in bias adjustment we also randomly selected 20% of images from Caltech-101 dataset. The object parts of both optimized and unoptimized vocabulary are show in Figure 6.

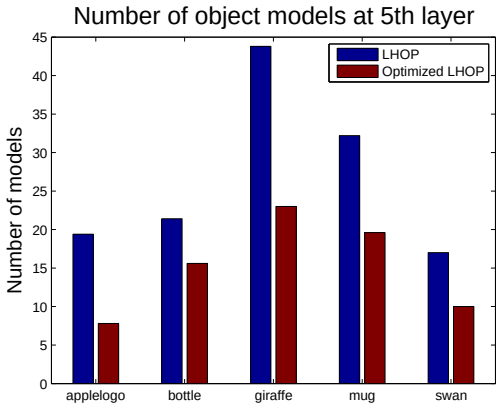


Figure 3: Number of object parts in 5th layer of our **optimized vocabulary (red)** compared to **original vocabulary (blue).**

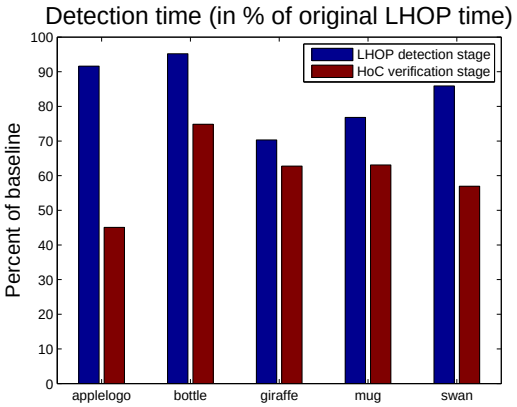


Figure 4: Computation time speedup. We report time required to complete **LHOP detection stage (blue)** and **HoC verification stage (red)** when using optimized vocabulary expressed in percent of the time taken by the unoptimized vocabulary.

3.3 Learning HoC classifier

In this process we used optimized vocabulary $\mathcal{L}_{opt}$ to produce hard-negative examples on *ETHZ Shape Classes* training set. Similarly as before we used detections overlapping less than 0.3 with the groundtruth as negative and examples overlapping more than 0.7 as positive. Together with the extracted original positive examples (regions were scaled to 120 pixels before inference process) we trained HoC classifier for each category.

3.4 Evaluation and results

In Figures 2 we measured the absolute number of all detections for each category separately. Figure 2a shows the number of detections *before* the non-maxima suppression and Figure 2b shows detections *after* the non-maxima suppression process. By looking at the detections before non-maxima suppression we see that the our method produces approximately 50% of all detections compared to the original vocabulary. The reductions is highest in the *apple logo*, *giraffe* and *swan* categories with only 45% of original detections while in categories *bottle* and *mug* there we retained 60% of all original detections. After the non-maxima suppression step this difference becomes less significant but we still produce by approximately 10% less detections.

We also measured a number of object models retained in the vocabulary after the optimization process. Figure 3 shows the absolute number of parts in 5th layer for each category compared to the number of parts from the original vocabulary. We can see that in each category we were able to significantly reduce number of parts. In particular the category *apple logo* has retained only 40% of original parts while category *giraffe* has around 52% of original parts. Slightly less parts were removed for other categories with 73% of original parts for *bottle*, 61% for *mug* and 58% of original parts for category *swan*.

Our optimizations also improved processing time in the detection stage. In Figure 4 we report for each category the time required for LHOP stage and for HoC verification stage in percent of the time required by the original vocabulary. Focusing on HoC verification stage (red bar), the biggest speedup has been achieved for the *apple logo* category with 45% of original time. With other categories the improvements are: 75% for *bottle*, 62% for *giraffe*, 63% for *mug* and 57% for category *swan*. We can also notice slight improvements in the LHOP inference stage (blue bar) with the averaged time of 83% of original time. Note, that we have optimized only 5th layer while other layers remained unchanged with the same number of parts. Considering that only 1/5th of the vocabulary has been optimized, the 17% speedup is quite significant. Overall we still need to take into account that LHOP inference stage takes most of the time and therefore overall speedup combined with HoC verification time is still around 17%.

We also evaluate overall detection rate to ensure we did not remove any information that is crucial for the detection. Looking at the result of Table 1 we see similar detection rate of our method and original method of [11]. In some cases our method performed by a percent or two better, while in other cases it performed slightly worse. In particular, our method achieved higher scores in categories *apple logo* and *mug*, while original method performed better in categories *bottle, giraffe* and *swan*. On average our method performed by less then half a percent worse, but still by almost 3% better then LHOP without any HoC verification step. We also noticed that both methods produced high variance, particularly in categories *bottle* and *swan*. The variations between different iterations are observable from Figure 5, where we see detection rate of category *bottle* varied between 70% and 90% and detection rate of category *swan* varied almost between 60% and 90%. Low performance is particularly noticeable in our method where at least two iterations in *swan* category produced detection rate of only 60%. This also explains a significant drop in overall performance of this category compared to the original method. Note, however overall performance is still at least at the same level as the results from [11].

We also report detection rate at 1.0 FPPI in Table 2 to compare our approach to the discriminative nodes of Kris-

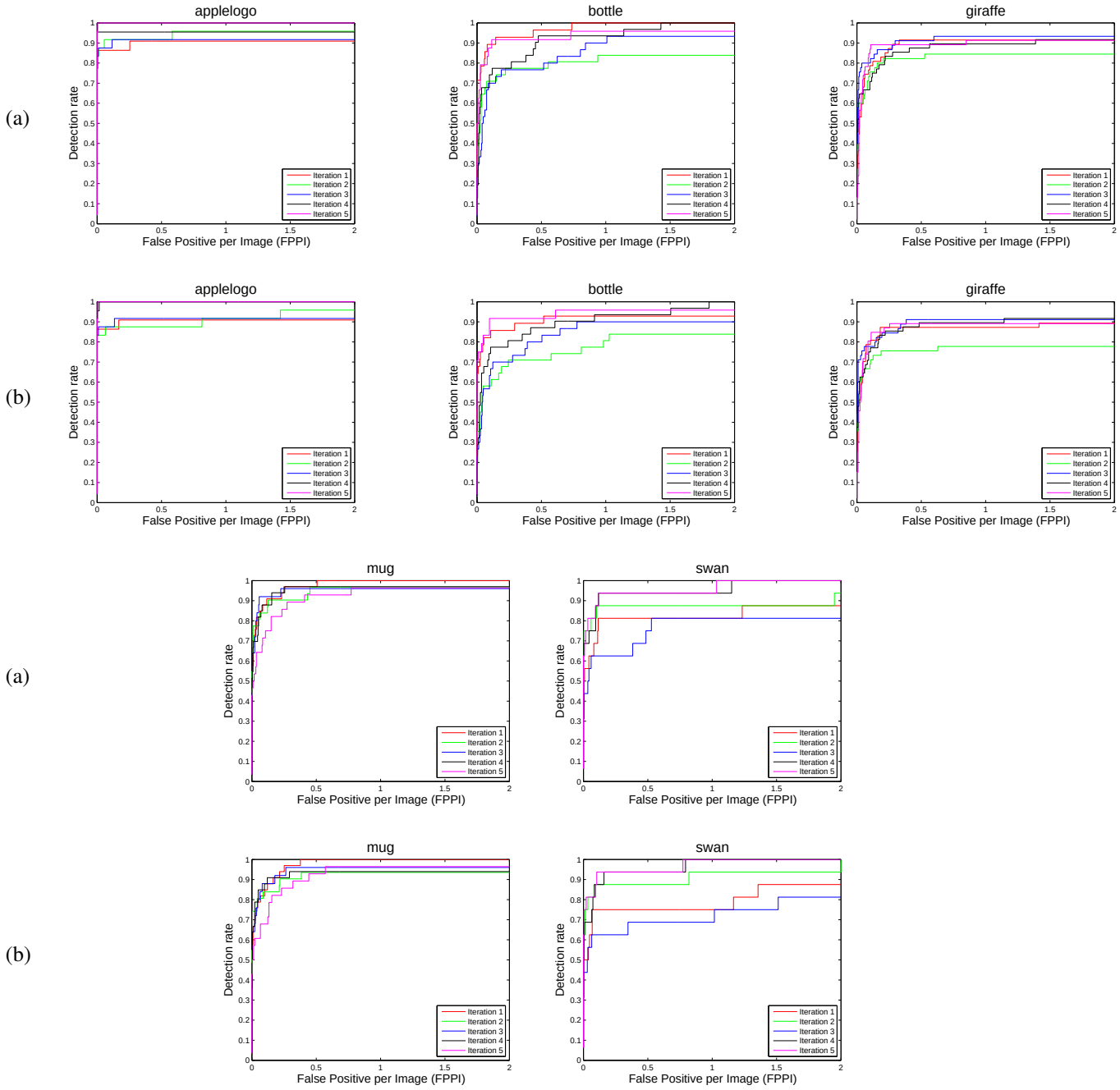


Figure 5: Detection rate over different FPPI rates (false positives per image) for each category from the *ETHZ Shape Classes* dataset. We ran five iterations and sampled examples randomly in each iteration to obtain different training/testing splits. Results are separated into row **(a) for unoptimized vocabulary** and row **(b) for optimized vocabulary (our).** Note, we used identical training/testing split for evaluation of both libraries.

	Apple logo	Bottle	Giraffe	Mug	Swan	*Average*
our method	**94.0 (5.6)**	83.1 (8.2)	86.1 (6.1)	**94.5 (3,9)**	83.7 (11.4)	88.3
LHOP + HoC verification [11]	93.9 (3.8)	84.5 (4.4)	**87.9 (4.0)**	93.9 (3.8)	**85.0 (10.5)**	***89.0***
LHOP only [3]	88.2 (3.4)	**87.6 (1.5)**	83.5 (1.1)	86.1 (2.0)	80.0 (3.5)	*85.1*
Maji et al. [7]	95.0	96.4	89.6	96.7	88.2	93.2

Table 1: Evaluation result on *ETHZ Shape Classes* with reported detection-rate (%) at **0.4 FPPI** averaged over five iterations (standard deviation values are shown in parentheses)

	[6]	our method
Apple logo	92.5	**95.7**
Bottle	85.4	**86.6**
Giraffe	82.3	**87.0**
Mug	86.5	**96.0**
Swan	70.5	**87.5**
Average	*83.4*	**90.4**

Table 2: Performance comparison to LHOP with discriminative node from [6] on *ETHZ Shape Classes* with reported detection-rate (%) at **1 FPPI** averaged over five iterations.

tan et al. [6]. We notice considerable improvements in the performance across all categories, with our average detection rate of 90.4% compared to detection rate of 83.4% in [6].

4 Discussion and conclusion

In this paper we introduced filtering with the discriminative information and vocabulary optimization to reduce number of false positive detections produced by hierarchical method learnt-hierarchy-of-parts (LHOP) [4]. We achieve this through vocabulary optimization to eliminate object models with poor discriminative information and through additional detection filtering step with the discriminative information. In contrast to [6] we introduce a discriminative information based on activation vector computed from response values and spatial information of activations from a detection's parse tree.

Using the *ETHZ Shape Classes [2]* database we have demonstrated that our method does not reduce performance in terms of detection rate but considerably reduces number of false positive detections while retaining only smaller set of object models in the vocabulary. In some cases, such as *apple logo* category, we are able to even increase the detection rate while we reduce false positives and the vocabulary of 5th layer by more then 60%. While detections before non-maxima suppression get reduced considerably, detections after non-maxima suppression are reduced only slightly. This indicates that in original method the false positive detections were handled mostly by good scoring of HoC descriptor and proper non-maxima suppression, while in our method they are now reduced even before non-maxima suppression. Additionally, we showed that our method also positively affect computation time. We were able to decrease computation time of LHOP inference stage by around 17% and by around 40% for the HoC verification stage. Lowered computation time of the HoC verification comes directly from the lower number of detections as there are less detections required to compute theirs HoC score with the non-linear kernel of support vectors. In the LHOP inference stage, saved computation time comes directly from the reduced number of object models in the optimized vocabulary. But since we optimize only 5th layer of the vocabulary we are only able to achieve 17% improvements in speedup.

In the future work we plan to reduce vocabulary parts in lower layers as well, in particular, we could remove parts connected to the 5th layer object parts that have been removed by our optimization method. We also plan to incorporate optimization step directly into learning of object layers to immediately select and reduce to the most optimal set of vocabulary parts.

Acknowledgments. This work was supported in part by ARRS research program P2-0214 and ARRS research projects J2-4284, J2-3607 and J2-2221.

References

[1] Chih Chung Chang and Chih Jen Lin. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[2] Vittorio Ferrari, Tinne Tuytelaars, and Luc Van Gool. Object detection by contour segment networks. In *Proceeding of the European Conference on Computer Vision*, volume 3953 of *LNCS*, pages 14–28. Elsevier, June 2006.

[3] Sanja Fidler, Marko Boben, and Aleš Leonardis. A coarse-to-fine taxonomy of constellations for fast multi-class object detection. In *ICCV*, pages 687–700, Berlin, Heidelberg, 2010. Springer-Verlag.

[4] Sanja Fidler and Ales Leonardis. Towards scalable representations of object categories: Learning a hierarchy of parts. In *CVPR*. IEEE Computer Society, 2007.

[5] Iasonas Kokkinos and Alan Yuille. Inference and learning with hierarchical shape models. *Int. J. Comput. Vision*, 93(2):201–225, June 2011.

[6] Matej Kristan, Marko Boben, Domen Tabernik, and Aleš Leonardis. Adding discriminative power to hierarchical compositional models for object class detection. In *18th Scandinavian Conference on Image Analysis*, Jun 2013.

[7] Subhransu Maji and Jitendra Malik. Object detection using a max-margin hough transform. In *CVPR*, pages 1038–1045. IEEE, 2009.

[8] Björn Ommer and Joachim M. Buhmann. Learning the compositional nature of visual objects. In *Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

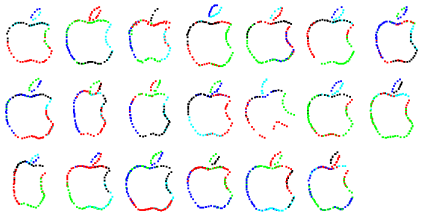
[9] Marc'Aurelio Ranzato, Fu J. Huang, Y. Lan Boureau, and Yann LeCun. Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1–8, 2007.

[10] D. Tabernik, M. Kristan, M. Boben, and A. Leonardis. Learning statistically relevant edge structure improves low-level visual descriptors. In *International Conference on Pattern Recognition*, 2012.

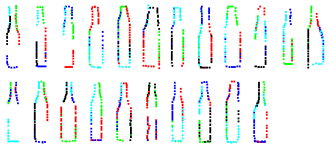
[11] Domen Tabernik, Matej Kristan, Marko Boben, and Aleš Leonardis. Hypothesis verification with histogram of compositions improves object detection of hierarchical models. In *Proceedings of the 21th International Electrotechnical and Computer Science Conference, ERK*, 2013.

[12] K. E. A. van de Sande, T. Gevers, and C. G. M. Snoek. Evaluating color descriptors for object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1582–1596, 2010.

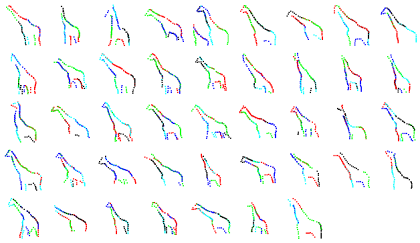
[13] Long Zhu, Yuanhao Chen, A. Torralba, W. Freeman, and A. Yuille. Part and appearance sharing: Recursive compositional models for multi-view. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1919 –1926, june 2010.



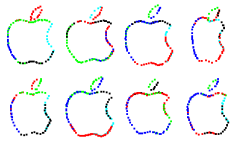
(a) Applelogo - unoptimized vocabulary



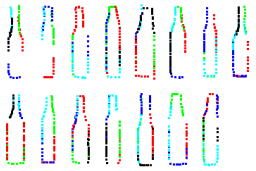
(b) Bottle - unoptimized vocabulary



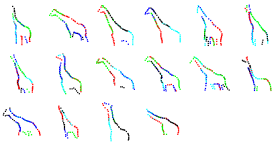
(c) Giraffe - unoptimized vocabulary



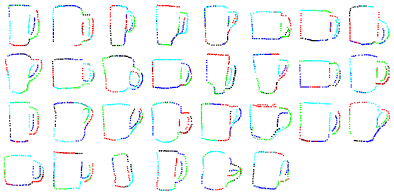
(d) Applelogo - optimized vocabulary (our)



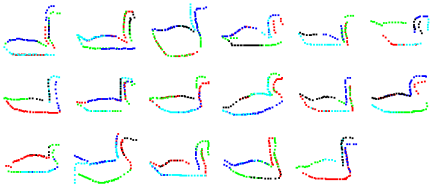
(e) Bottle - optimized vocabulary (our)



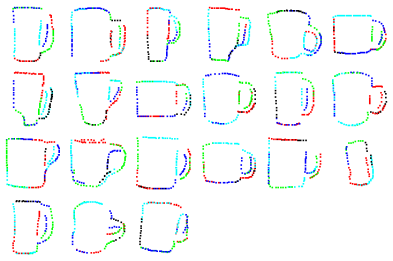
(f) Giraffe - optimized vocabulary (our)



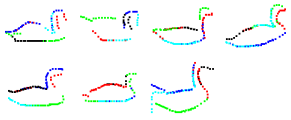
(g) Mug - unoptimized vocabulary



(h) Swan - unoptimized vocabulary



(i) Mug - optimized vocabulary (our)



(j) Swan - optimized vocabulary (our)

Figure 6: Examples of 5th layer object parts for each category. In the first and third row are parts of the original vocabulary while in the second and fourth row are parts of our optimized vocabulary.

Computer Vision Winter Workshop 2014, Zuzana Kúkelová and Jan Heller (eds.)
Křtiny, Czech Republic, February 3–5
Czech Society for Cybernetics and Informatics

Canonical Encoding of the Combinatorial Pyramid

Fuensanta Torres and Walter G. Kropatsch

PRIP, Vienna University of Technology, Austria
{fuensanta, krw}@prip.tuwien.ac.at

Abstract *This paper presents a novel encoding scheme for a combinatorial pyramid. A combinatorial pyramid is a hierarchy of successively reduced combinatorial maps. Important properties of the combinatorial pyramids such as topology preservation, the process global and local features within the same data structure, etc. made them useful for image processing and pattern recognition tasks. Their advantages have been widely proved in the literature. Nevertheless, the main disadvantage of this approach is the high rate of memory requirement. A combinatorial map of an image maybe stored in an array of size approximately equal to four times the number of pixels of the image. Furthermore, every level of the combinatorial pyramid stores a different combinatorial map. In respond to this problem a canonical encoding of the combinatorial pyramid is provided. It consists of a single array where its elements are ordered with respect to the construction history of the pyramid. In this manner the memory consumptions are equal to the size of the initial combinatorial map and do not depend on the number of pyramid's levels. In addition, this canonical encoding allows the whole reconstruction of the pyramid in both directions: from the base to the top level and from the top to the base level, without additional information.*

1 Introduction

A 2D combinatorial map [11, 15] defines a data structure to encode the subdivision of the plane in different regions. It has more advantages compared to the traditional Region Adjacency Graphs (RAG):

- The combinatorial maps represent topological information (multi-adjacency or inclusion relations). Unlike the RAG, where two topologically different images could be represented by the same RAG.
- They can be extended to higher dimensions (nD).
- They allow efficient algorithms to retrieve information and modify the partition.

The combinatorial pyramid [4] is a stack of successively reduced combinatorial maps. Such structure takes advantages of the combinatorial maps as well as benefits from additional properties:

- The combinatorial pyramids preserve topology.
- They process global and local features within the same data structure.

Nowadays, the combinatorial maps and the combinatorial pyramids are applied for various tasks such as image segmentation [1, 8], map matching [9, 14, 13, 17], 3D mesh representation [10], etc. These structures require high memory consumptions, and this requirement is even more exacerbated by the pyramid -the pyramid structure stores one combinatorial map at each level of its hierarchy.
The literature reports several methods to reduce the memory consumptions [12, 16, 5].
Goffe et al [12] segment the initial image -they pre-process the initial image. They find the combinatorial map for this segmented image -it is the top level of the pyramid. And they progressively create the lower levels of the hierarchy by increasing the size of the combinatorial maps.
[16] and [5] define the implicit encoding of the 2-dimensional and n-dimensional combinatorial pyramid, respectively. For the 2D implicit encoding, Brun et al [5] store the combinatorial map of the initial image and two functions which describe the construction history -one function specifies the type of operation done over each element of the initial combinatorial map and the other function defines the highest level of the pyramid until which the element survives.

The canonical encoding of the combinatorial pyramid stores the whole pyramid and its construction history in the same memory space than the initial combinatorial map. The operation applied to each element of the initial combinatorial map is implicitly encoded in the representation; and the highest level until which the element survive are implicitly encoded in the order of the elements. In this manner the memory consumptions are equal to the initial combinatorial map and do not depend on the number of levels of the pyramid, as previous works. It allows the construction of the pyramid from the top to the base level without additional information. Meanwhile Goffe et al [12] require additional information (parent/child relations).

The rest of this paper is organized as follows: Basic definitions on combinatorial maps and combinatorial pyramids are recalled in sections 2 and 3. Section 4 describes our contribution -the canonical encoding of the combinatorial pyramid. The experimental results proving the theoretical concept of the canonical encoding are described in Section 5. Finally, conclusions and future work are unfolded at Sec-

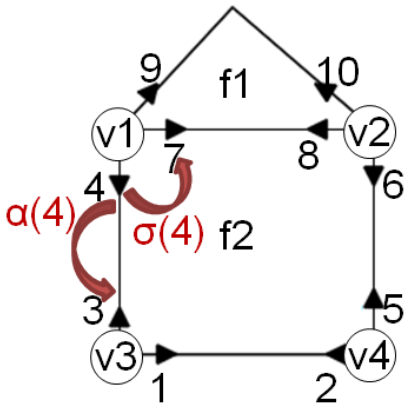


Figure 1: Combinatorial map example.

Table 1: Combinatorial Map (G=($\mathcal{D}$,α,σ)) of Fig. 1.

darts (d)	1	2	3	4	5	6	7	8	9	10
α	2	1	4	3	6	5	8	7	10	9
σ	3	5	1	7	2	10	9	6	4	8

tion 6.

2 Recalls on Combinatorial Maps

Definition 1. ***2D Combinatorial Map:*** *A 2D combinatorial map encodes the subdivision of the plane in different regions. It represents the inclusion and adjacency relations between the regions. This principle enables to fully describe the topology of the plane partition [11, 15]. The 2D combinatorial map (G) is defined by a triplet G=($\mathcal{D}$, α, σ), where:*

- *$\mathcal{D}$ is a finite set of darts.*
- *α is an involution on the set $\mathcal{D}$.*
- *σ is a permutation on the set $\mathcal{D}$.*

An additional explanations of the definition above: The edges (paths connecting two vertices) are divided into two half edges called darts. α is an one-to-one mapping between consecutive darts forming the same edge, such that $\alpha(\alpha(d))=d$. σ is a mapping between consecutive darts around the same vertex while turning counterclockwise.

Example. 2D Combinatorial Map: Fig. 1 and Tab. 1 give an example of a 2D combinatorial map. Where:

- $\mathcal{D} = \{ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 \}$.
- $\alpha(4) = 3$.
- $\sigma(4) = 7$.

3 Recalls on Combinatorial Pyramids

Definition 2. ***Combinatorial Pyramid:*** *A combinatorial pyramid is a stack of successively reduced combinatorial maps (Fig. 2). The size of the combinatorial maps is successively reduced by contractions and removals. The contraction and the removal kernels specify a set of darts, which will be contracted and removed, respectively, between two consecutive levels (as previously described [7]). Given an initial combinatorial map G_0=($\mathcal{D}$,α,σ) and the sequence of kernels (k_1, k_2, k_3,..., k_n) we can build the stack of successively reduced combinatorial maps G_1, G_2,..., G_n. The combinatorial maps at all the levels preserve the topology of the initial combinatorial map [3, 6].*

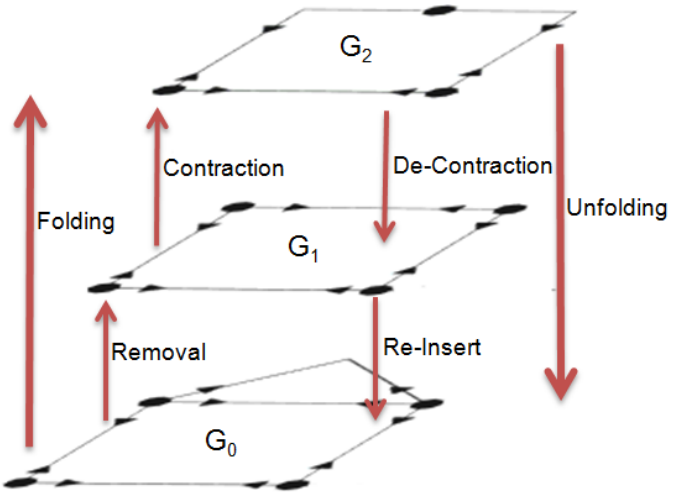


Figure 2: Combinatorial pyramid example.

Example. Combinatorial Pyramid: Fig. 2 gives an example of a combinatorial pyramid. Where:

- G_0 (the base level in the pyramid) is equal to the combinatorial map of Fig. 1.
- We reduce the number of darts in G_0 by the removal kernel k_1 = {9, 10} and we obtain G_1.
- We apply on G_1 the contraction kernel k_2 = {7, 8} and we get the top level (G_2) of the pyramid.

4 Folding and Unfolding the Pyramid

The aim of this section is to explain the canonical encoding of the combinatorial pyramid.

4.1 Folding the Pyramid

The operations used to build a pyramid are the removal and contraction operations. The kernels K (as described in Sec. 3) selects the darts which will be removed or contracted. The removal and contraction involve 6 dependent darts (Fig. 3): d, $\alpha(d)$, f= $\sigma^{-1}(d)$, g= $\sigma(d)$, h= $\sigma^{-1}(\alpha(d))$, i= $\sigma(\alpha(d))$. Special cases are the empty self-loop and the pending edge where $\sigma(d) = \alpha(d)$ and $\sigma(\alpha(d)) = \alpha(d)$, respectively.
We need additional definitions in order to be able to define the contraction and removal operations [2]:

Definition 3. *σ^* is the σ orbit, which defines all the darts belonging to the same vertex.*

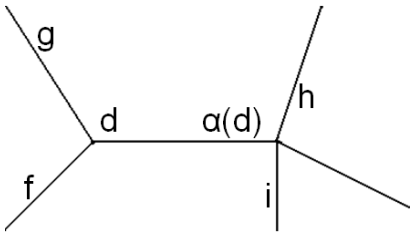


Figure 3: Dependent Darts in the operations.

NOTE: $\sigma^{-1}(d)$= $\sigma^n(d)$ with n= min$\{i \mid \sigma^i(d) = d\}$; where n is the number of successive applications of σ on the dart d.

Definition 4. *α^* is the α orbit, which defines the pair of darts belonging to the same edge.*

Definition 5. ***Empty Self loop:*** *A pair of darts are an empty self loop, iff $\sigma(d) = \alpha(d)$ for $d \in \alpha^*$.*

Definition 6. ***Pending edge:*** *A pair of darts are a pending edge, iff $\sigma(\alpha(d)) = \alpha(d)$ for $d \in \alpha^*$.*

Definition 7. ***Removal Operation [2]:*** *We remove a pair of darts $\alpha^*(d)$ from G=($\mathcal{D}$,α,σ), $\mathcal{D}'$= $\mathcal{D} \setminus \alpha^*(d)$. And in addition, we modify the permutation σ and we obtain σ'. The values of σ' for the all the d' $\in \mathcal{D}'$ are:*

- *if $\alpha^*(d)$ is not an empty self loop ($\sigma(d) = \alpha(d)$) and neither a pending edge ($\sigma(\alpha(d)) = \alpha(d)$):*

$$\sigma'(\sigma^{-1}(d)) = \sigma(d) \tag{1}$$

$$\sigma'(\sigma^{-1}(\alpha(d))) = \sigma(\alpha(d)) \tag{2}$$

- *if $\alpha^*(d)$ is an empty self loop ($\sigma(d) = \alpha(d)$):*

$$\sigma'(\sigma^{-1}(d)) = \sigma(\alpha(d)) \tag{3}$$

- *For the rest of darts, which are not included in the cases above:*

$$\forall d' \in D \setminus \{\sigma^{-1}(d), \sigma^{-1}(\alpha(d))\}\sigma'(d') = \sigma(d') \tag{4}$$

Parts disappearing from G are saved as left over of the removal (LoR). LoR is a quadruplet $\{p, q, r, s\} \in \mathcal{D}^4$, where p= d, q= $\alpha(d)$ r=$\sigma(d)$ and s=$\sigma(\alpha(d))$.

Proposition 1.

1. *The triplet ($\mathcal{D}'$, α, σ') form a valid combinatorial map(G').*
2. *In an empty self-loop:*

$$r = q \tag{5}$$

3. *In a pending edge:*

$$s = q \tag{6}$$

Proof. (1) The proof that G' is a valid combinatorial map without $\alpha^*(d)$ can be found in [2] □

Proof. (2) $r = \sigma(d)$, $q = \alpha(d) \Rightarrow \sigma(d)$= $\alpha(d)$ is an empty self-loop. □

Proof. (3) $s = \sigma(\alpha(d))$, $q = \alpha(d) \Rightarrow \sigma(\alpha(d))$= $\alpha(d)$ is a pending edge. □

Definition 8. ***Contraction Operation [2]:***

We remove a pair of darts $\alpha^*(d)$ from G=($\mathcal{D}$,α,σ), $\mathcal{D}'$ = $\mathcal{D} \setminus \alpha^*(d)$. And in addition, we modify the permutation σ and we obtain σ'. The values of σ' for the all the d' $\in \mathcal{D}'$ are:

- *if $\alpha^*(d)$ is not an empty self loop ($\sigma(d) = \alpha(d)$) and neither a pending edge ($\sigma(\alpha(d)) = \alpha(d)$):*

$$\sigma'(\sigma^{-1}(d)) = \sigma(\alpha(d)) \tag{7}$$

$$\sigma'(\sigma^{-1}(\alpha(d))) = \sigma(d) \tag{8}$$

- *if $\alpha^*(d)$ is a pending edge ($\sigma(\alpha(d)) = \alpha(d)$):*

$$\sigma'(\sigma^{-1}(d)) = \sigma(d) \tag{9}$$

- *For the rest of darts, which are not included in the cases above:*

$$\forall d' \in D \setminus \{\sigma^{-1}(d), \sigma^{-1}(\alpha(d))\}\sigma'(d') = \sigma(d') \tag{10}$$

Parts disappearing from G are saved as left over of the contraction (LoC). LoC is a quadruplet $\{p, q, r, s\} \in \mathcal{D}^4$, where p= d, q= $\alpha(d)$ r=$\sigma(d)$ and s=$\sigma(\alpha(d))$.

Proposition 2.

1. *The triplet ($\mathcal{D}'$, α, σ') form a valid combinatorial map(G').*
2. *In an empty self-loop:*

$$r = q \tag{11}$$

3. *In a pending edge:*

$$s = q \tag{12}$$

Proof. (1) The proof that G' is a valid combinatorial map without $\alpha^*(d)$ can be found in [2] □

Proof. (2) $r = \sigma(d)$, $q = \alpha(d) \Rightarrow \sigma(d)$= $\alpha(d)$ is an empty self-loop. □

Proof. (3) $s = \sigma(\alpha(d))$, $q = \alpha(d) \Rightarrow \sigma(\alpha(d))$= $\alpha(d)$ is a pending edge. □

Tab. 2 summarize the contraction and removal operations.

Example. Remove Empty Self-Loop: Fig. 4 gives an example of an empty self-loop removal. It shows the combinatorial map before the removal (Fig. 4 on the left) and after the removal (Fig. 4 on the right). Where we can see that the value of $\sigma'(3)$ changes according to Tab. 2, $\sigma'(\sigma^{-1}(d)) := \sigma(\alpha(d))$ (eq. 3) ($\sigma(3)$=d, $\sigma'(3)$=1).

Table 2: Removal and Contraction operations.

case	REDUCE (d, $\alpha(d)$)
pending edge	if $\sigma(\alpha(d)) = \alpha(d)$ then $\sigma'(\sigma^{-1}(d)) := \sigma(d)$ (eq. 9)
empty-self-loop	if $\sigma(d) = \alpha(d)$ then $\sigma'(\sigma^{-1}(d)) := \sigma(\alpha(d))$ (eq. 3)
remove	$\sigma'(\sigma^{-1}(d)) := \sigma(d)$ (eq. 1) $\sigma'(\sigma^{-1}(\alpha(d))) := \sigma(\alpha(d))$ (eq. 2)
contract	$\sigma'(\sigma^{-1}(d)) := \sigma(\alpha(d))$ (eq. 7) $\sigma'(\sigma^{-1}(\alpha(d))) := \sigma(d)$ (eq. 8)

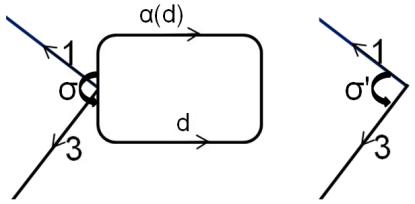


Figure 4: Remove Empty-Self-Loop(d,$\alpha(d)$).

Definition 9. ***The canonical encoding****: The canonical encoding of the combinatorial pyramid is an ordered sequence of darts which fully encodes the combinatorial map G at any level and its construction history. The darts are encoded by even and odd integers; in a way that the involution α could be implicitly encoded by eq. 13.*

$$\alpha(d) = \begin{cases} d+1 & \text{if } d \text{ odd} \\ d-1 & \text{if } d \text{ even} \end{cases} \tag{13}$$

The pyramid is built by a sequence of contraction and removal operations. Each one producing a smaller combinatorial map and the quadruplet of the Lo (Left over). The canonical encoding reorder the darts of the Lo in the chronological order. The surviving darts constitute the active part of the canonical encoding. Meanwhile, the ordered Lo constitute the passive part.

We assume that we have executed n operations in total, then we have n Lo (Tab. 3). The Π function (Π: $\mathcal{D} \rightarrow \mathcal{E}$) (eq. 14, 15) gives the new order of this darts in the passive part of the canonical encoding. The unique identifier of each p_t and q_t is assigned to its position in the passive part, therefore we only need to store $\Pi\ (r_t)$ and $\Pi\ (s_t)$ (such that $\mathcal{E}_1$=$\Pi\ (r_1)$, $\mathcal{E}_2$=$\Pi\ (s_1)$, etc.).

$$\Pi(p_t) = 2t - 1 \tag{14}$$

$$\Pi(q_t) = 2t \tag{15}$$

Table 3: Left over table.

t	Left over (Lo)
1	p_1, q_1, r_1, s_1
2	p_2, q_2, r_2, s_2
3	p_3, q_3, r_3, s_3
4	p_4, q_4, r_4, s_4
n	p_n, q_n, r_n, s_n

Property. Canonical Encoding (Folding the pyramid): The canonical encoding of the combinatorial pyramid encodes the whole pyramid with the same memory as in the base level -it does not increase with the height of the pyramid. Traditional methods (also called explicit encoding) store the initial combinatorial map at the base level; but they also need additional memory every new level -they store a new combinatorial map per level. The implicit encoding [5] also needs additional memory every new level to describe the construction history of the pyramid.

4.2 Unfolding the Pyramid

In order to retrieve the initial combinatorial map (G_0) from the combinatorial map at any level, the darts in the passive part are moving to the active part. In the folding previous to the unfolding operation, the darts have been ordered with respect to the construction history. Therefore, we only have to shift the boundary between the active and the passive part; and to apply -re-insertion or de-contraction.

Definition 10. ***Re-insertion Operation:*** *We add the LoR(t)=$\{p, q, r, s\}$ to G'=($\mathcal{D}'$,α,σ'), $\mathcal{D}''$= $\mathcal{D}' \cup \{p, q\}$. And in addition, we modify the permutation σ' and we obtain σ''. The values of σ'' for the all the d" $\in \mathcal{D}''$ are:*

- *if $r \neq q$(eq. 5) and $s \neq q$ (eq. 6):*

$$\sigma''(\sigma'^{-1}(r)) := p \tag{16}$$

$$\sigma''(\sigma'^{-1}(s)) := q \tag{17}$$

- *if $r = q$ (eq. 5):*

$$\sigma''(\sigma'^{-1}(s)) := p \tag{18}$$

- *For the rest of darts, which are not included in the cases above:*

$$\forall d'' \in D'' \setminus \{\sigma'^{-1}(p), \sigma'^{-1}(q)\}\sigma''(d'') = \sigma'(d'') \tag{19}$$

The values of α'' for the all the d" $\in \mathcal{D}''$ are:

$$\forall d'' \in D'' \setminus \{p, q\}\alpha''(d'') = \alpha'(d'') \tag{20}$$

$$\{p, q\} \in LoR(t)''\alpha''(p) = q; \alpha''(q) = p \tag{21}$$

Proposition 3.

1. *The triplet ($\mathcal{D}''$, α'', σ'') form a valid combinatorial map(G").*
2. *G"(Def. 10)= G (Def. 7).*

Proof. (1)

- $\mathcal{D}''$= $\mathcal{D}' \cup \{p, q\}$ is a finite set of darts.
- $\forall d'' \in \mathcal{D}' \Rightarrow \alpha''\ (d'')$= $\alpha'\ (d'')$ (eq. 20) is an involution.
- $\forall d'' \notin \mathcal{D}' \Rightarrow d'' = p \in LoR$ and $\alpha''\ (d'') = q \in LoR$ $\Rightarrow q = \alpha''(p)$ (eq. 21) is an involution.

- $\forall d''$ such that $\sigma'(d'') \neq r$ and $\sigma'(d'') \neq s \Rightarrow \sigma''(d'')$= $\sigma'(d'')$ (eq. 19) is a permutation.
- if $\sigma'(d'') = r$ and $r \neq q$ (eq. 5) and $s \neq q$ (eq. 6) $\Rightarrow \sigma''(d'') = p$ (eq. 16) and $\sigma''(p) = r$ (Def. 7) is a permutation.
- if $\sigma'(d'') = s$ and $r \neq q$ (eq. 5) and $s \neq q$ (eq. 6) $\Rightarrow \sigma''(d'') = q$ (eq. 17) and $\sigma''(q) = s$ (Def. 7) is a permutation.
- if $\sigma'(d'') = s$ and $r = q$ (eq. 5) $\Rightarrow \sigma''(d'') = p$ (eq. 18) and $\sigma''(q) = s$ (Def. 7) is a permutation.

□

Proof. (2)

- if $r \neq q$ (eq. 5) and $s \neq q$ (eq. 6):

  Given $\sigma''(\sigma'^{-1}(r)) := p$ (eq. 16). We have $r = \sigma'(\sigma^{-1}(p))$ (eq. 1) thus:
  $\sigma''(\sigma'^{-1}(\sigma'(\sigma^{-1}(d))) = \sigma''((\sigma^{-1}(p))) := p$

  Given $\sigma''(\sigma'^{-1}(s)) := q$ (eq. 17). We have $s = \sigma'(\sigma^{-1}(q))$ (eq. 2) thus:
  $\sigma''(\sigma'^{-1}(\sigma'(\sigma^{-1}(q)))) = \sigma''(\sigma^{-1}(q)) := q$

- if $r = q$ (eq. 5):

  Given $\sigma''(\sigma'^{-1}(s)) := p$ (eq. 18). We have $s = \sigma'(\sigma^{-1}(p))$ (eq. 3) thus:
  $\sigma''(\sigma'^{-1}(\sigma'(\sigma^{-1}(p)))) = \sigma''(\sigma^{-1}(p)) := p$

- $\forall d'' \in \mathcal{D}'' \setminus \{\sigma'^{-1}(r), \sigma'^{-1}(s)\}$ $\sigma''(d'') = \sigma'(d'')$ (eq. 19). We have $\forall d' \in \mathcal{D} \setminus \{\sigma^{-1}(d), \sigma^{-1}(\alpha(d))\}$ $\sigma'(d') = \sigma(d')$ (eq. 4) thus:
  $\sigma''(d'') = \sigma(d'')$

□

Definition 11. ***De-contraction Operation:*** *We add the LoC(t)=$\{p, q, r, s\}$ to G'=($\mathcal{D}'$,α, σ'), D''= $\mathcal{D}' \cup \{p, q\}$. And in addition, we modify the permutation σ' and we obtain σ''. The values of σ'' for the all the d" $\in \mathcal{D}''$ are:*

- *if $r \neq q$ (eq. 11) and $s \neq q$ (eq. 12):*

$$\sigma''(\sigma'^{-1}(r)) := q \tag{22}$$

$$\sigma''(\sigma'^{-1}(s)) := p \tag{23}$$

- *if $s = q$ (eq. 12):*

$$\sigma''(\sigma'^{-1}(r)) := p \tag{24}$$

- *For the rest of darts, which are not included in the cases above:*

$$\forall d'' \in D \setminus \{\sigma'^{-1}(p), \sigma'^{-1}(q)\}\sigma''(d'') = \sigma'(d'') \tag{25}$$

The values of α'' for the all the d" $\in \mathcal{D}''$ are:

$$\forall d'' \in D'' \setminus \{p, q\}\alpha''(d'') = \alpha'(d'') \tag{26}$$

$$\{p, q\} \in LoC(t)''\alpha''(p) = q; \alpha''(q) = p \tag{27}$$

Proposition 4.

1. *The triplet ($\mathcal{D}''$, α'', σ'') form a valid combinatorial map(G").*
2. *G"(Def. 11)= G (Def. 8).*

Proof. (1)

- $\mathcal{D}''$= $\mathcal{D}' \cup \{p, q\}$ is a finite set of darts.
- $\forall d'' \in \mathcal{D}' \Rightarrow \alpha''(d'')$= $\alpha'(d'')$ (eq. 26) is an involution.
- $\forall d'' \notin \mathcal{D}' \Rightarrow d'' = p \in LoC$ and $\alpha''(d'') = q \in LoC$ $\Rightarrow q = \alpha''(p)$ (eq. 27) is an involution.
- $\forall d''$ such that $\sigma'(d'') \neq r$ and $\sigma'(d'') \neq s \Rightarrow \sigma''(d'')$= $\sigma'(d'')$ (eq. 25) is a permutation.
- if $\sigma'(d'') = r$ and $r \neq q$ (eq. 11) and $s \neq q$ (eq. 12) $\Rightarrow \sigma''(d'') = p$ (eq. 23) and $\sigma''(p) = r$ (Def. 8) is a permutation.
- if $\sigma'(d'') = s$ and $r \neq q$ (eq. 11) and $s \neq q$ (eq. 12) $\Rightarrow \sigma''(d'') = q$ (eq. 22) and $\sigma''(q) = s$ (Def. 8) is a permutation.
- if $\sigma'(d'') = r$ and $s = q$ (eq. 12) $\Rightarrow \sigma''(d'') = p$ (eq. 24) and $\sigma''(q) = s$ (Def. 8) is a permutation.

□

Proof. (2)

- if $r \neq q$ (eq. 11) and $s \neq q$ (eq. 12):

  Given $\sigma''(\sigma'^{-1}(r)) := q$ (eq. 22). We have $r = \sigma'(\sigma^{-1}(q))$ (eq. 8) thus:
  $\sigma''(\sigma'^{-1}(\sigma'(\sigma^{-1}(q)))) = \sigma''(\sigma^{-1}(q)) := q$

  Given $\sigma''(\sigma'^{-1}(s)) := p$ (eq. 23). We have $s = \sigma'(\sigma^{-1}(p))$ (eq. 7) thus:
  $\sigma''(\sigma'^{-1}(\sigma'(\sigma^{-1}(p)))) = \sigma''(\sigma^{-1}(p)) := p$

- if $s = q$ (eq. 12):

  Given $\sigma''(\sigma'^{-1}(r)) := p$ (eq. 24). We have $\sigma'(\sigma^{-1}(p)) = r$ (eq. 9).
  $\sigma''(\sigma'^{-1}(\sigma'(\sigma^{-1}(p)))) = \sigma''(\sigma^{-1}(p)) := p$

- $\forall d'' \in \mathcal{D} \setminus \{\sigma'^{-1}(d), \sigma'^{-1}(\alpha(d))\}$ $\sigma''(d'') = \sigma'(d'')$ (eq. 25). We have $\forall d' \in \mathcal{D} \setminus \{\sigma^{-1}(d), \sigma^{-1}(\alpha(d))\}$ $\sigma'(d') = \sigma(d')$ (eq. 10) thus:
  $\sigma''(d'') = \sigma(d'')$

Table 4: Re-insert and De-contract operations.

case	EXPAND (p, q)
	if s = q then $\sigma''(\sigma'^{-1}(r)) := p$ (eq. 24)
	if r = q then $\sigma''(\sigma'^{-1}(s)) := p$ (eq. 18)
re-insert -conditions	if $\sigma'(q) \notin \sigma'^{*}(\sigma'(p))$ or $\sigma'(p) \notin \sigma'^{*}(\sigma'(q))$ if $\sigma'^{*}(p) = \sigma'^{*}(q)$
-operations	$\sigma''(\sigma'^{-1}(s)) := q$ (eq. 16) $\sigma''(\sigma'^{-1}(s)) := q$ (eq. 17)
de-contract -conditions	if $\sigma'(q) \in \sigma'^{*}(\sigma'(p))$ or $\sigma'(p) \in \sigma'^{*}(\sigma'(q))$
-operations	$\sigma''(\sigma'^{-1}(r)) := q$ (eq. 22) $\sigma''(\sigma'^{-1}(s)) := p$ (eq. 23)

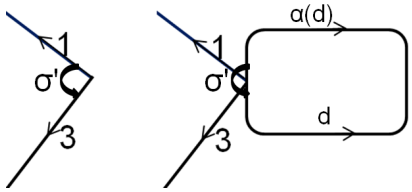


Figure 5: Re-Insert Empty-Self-Loop(d, $\alpha(d)$).

□

Tab. 4 summarizes the re-insertions and de-contractions. It gives the conditions to recognize whether the pair of darts were previously either removed (re-insert conditions) or contracted (de-contract conditions) in the folding procedure.

Example. Re-Insert Empty-Self-Loop: Fig. 5 gives an example of an empty self-loop re-insertion. It shows the combinatorial map before the re-insertion (Fig. 5 on the left) and after the re-insertion (Fig. 5 on the right). Where we can see that the value of $\sigma''(3)$ changes according to Tab. 4, $\sigma''(\sigma'^{-1}(\sigma'(\alpha(d)))) := d$ (eq. 18) ($\sigma'(3) = 1$, $\sigma''(3) = d$).

Property. Canonical Encoding (Unfolding the pyramid): Given the canonical encoding of the combinatorial pyramid at any level, the initial combinatorial map (G_0) can be retrieved -without extra information. Traditional methods store the parent/child relations to be able to unfold the pyramid. In the canonical encoding, we detect if we should either re-insert or de-contract the pair of darts, of the passive part, according with Tab. 4. And we apply the corresponding operation either re-insertion or de-contraction.

5 Proof of concepts

Application (Connected component labeling). The canonical encoding of the combinatorial pyramid has been used for connected components labeling. At the base level G_0, a pair of darts (d, α(d)) connects a pixel of the initial image with its 4-neighbors. Each dart stores the color of its related pixel. In the connected component application,

Figure 6: Input Image.

the contraction kernels are composed of darts all having the same color. The contraction may create redundant edges, which constitutes the removal kernels.

Example (Connected components labeling). Fig. 6 is the input image of the combinatorial pyramid. Fig. 7 shows the first level of the pyramid, where the pixels which have all their related darts in the passive part have black color. Fig. 8 shows the combinatorial map at the top level of the pyramid. Each connected component is contracted to a single vertex. The combinatorial map encodes the inclusion and adjacency relations among these connected components. It fully describes the topology of the plane partition.

Property (Memory requirements). A combinatorial map with $|\mathcal{D}|$ darts maybe stored in one array of dimensions equal to $|\mathcal{D}| \times \log_2(\mathcal{D})$ bits (assuming that the unique identifier of each dart (d) is assigned to its position in the array, we only need to store $\sigma(d)$). If the reduction factor between any two levels of the pyramid is K. The number of darts at one level of the pyramid G_l is $\frac{|\mathcal{D}|}{k^l}$. Therefore, the bits used to store the pyramid explicitly is $\log_2(D) \sum_{l=0}^{n} \frac{|\mathcal{D}|}{k^l}$. It also needs in addition the parent/child relations to unfold the pyramid. The implicit encoding [16, 5] requires the storage of the combinatorial map at the base level and one integer for each pair darts. The bits used to store the implicit encoding is $|\mathcal{D}| \log_2(D) + \frac{1}{2} |D| (\log_2(n))$. The canonical implementation requires only the storage of the combinatorial map at the base level $|\mathcal{D}| \times \log_2(D)$ bits.

6 Conclusions and Future work

In the present work, we propose a canonical encoding of a combinatorial pyramid. This new structure stores the whole combinatorial pyramid and its construction history in the same memory as the initial combinatorial map. We use the order of the darts to encode the information about the pyramid structure. It allows the full reconstruction of the pyra-

Figure 7: Darts at the first level(D_1).

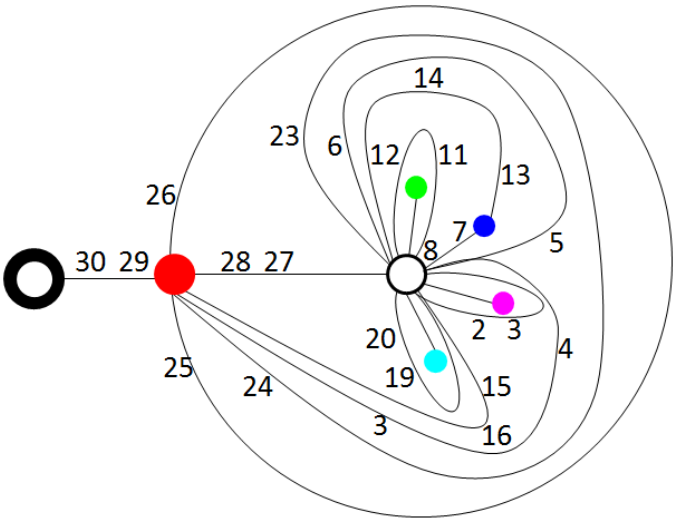


Figure 8: Combinatorial Map at the top level(G_9).

mid in both directions (folding and unfolding the pyramid). Different ways to construct a combinatorial pyramid can be found in the literature. Such methods either store a stack of successively reduced maps; or they store the initial combinatorial map and additional information to describe the construction history of the pyramid. The canonical encoding of the combinatorial pyramid reduces the memory requirements of the previous works without loss the functionality. In our proposed framework to encode the combinatorial pyramid the darts ranked according to its importance on the initial image (i.e. most of the darts of the uniform regions are contracted at the firsts level of the pyramid. They will be in the lasts positions of the canonical encoding). Now, we plan to study this property of our canonical encoding and to build signatures for image matching applications such as in [14, 13].

Acknowledgement

The first author thanks to Doctoral College on Computational Perception (Vienna University of Technology, Austria) for funding.

References

[1] E. Antunez, R. Marfil, J. P. Bandera, and A. Bandera. Part-based object detection into a hierarchy of image segmentations combining color and topology. *Pattern Recognition Letters*, 2013.

[2] L. Brun and W. Kropatsch. Dual contraction of combinatorial maps. Technical report, Pattern Recognition and Image Processing Group. Vienna University of Technology, 1999.

[3] L. Brun and W. G. Kropatsch. Dual contraction of combinatorial maps. *Workshop on Graph-based Representations*, 1999.

[4] L. Brun and W. G. Kropatsch. Introduction to combinatorial pyramids. *Lecture Notes in Computer Science 2243.*, 2001.

[5] L Brun and W. G. Kropatsch. Combinatorial pyramids. *International Conference on Image Processing (ICIP), Barcelona, Spain*, 2003.

[6] L. Brun and W. G. Kropatsch. Construction of combinatorial pyramids. *Hancock, E.R., Vento,M. (eds.) GbRPR 2003. LNCS, vol. 2726, pp. 112. Springer*, 2003.

[7] L. Brun and W. G. Kropatsch. Contraction kernels and combinatorial maps. *Pattern Recognition Letters*, 2003.

[8] L. Brun, M. Mokhtari, and F. Meyer. Hierarchical watersheds within the combinatorial pyramid framework. *Discrete Geometry for Computer Imagery. Springer Berlin Heidelberg*, 2005.

[9] L. Brun and J. H. Pruvot. Hierarchical matching using combinatorial pyramid framework. *Lecture Notes in Computer Science, vol. 5099. Springer, pp. 346355*, 2008.

[10] X. Feng, Y. Wang, Y. Weng, and Y. Tong. Compact combinatorial maps in 3d. *Computational Visual Media. Springer Berlin*, 2012.

[11] A. J. Gareth and D. Singerman. Theory of maps on orientable surfaces. *Proceedings of the London Mathematical Society*, 1978.

[12] R. Goffe, L. Brun, and G. Damiand. Tiled topdown combinatorial pyramids for large images representation. *International Journal of Imaging Systems and Technology*, 2011.

[13] S. Gosselin, G. Damiand, and C. Solnon. Efficient search of combinatorial maps. *Unknown Journal*, 2011.

[14] S. Gosselin, G. Damiand, and C. Solnon. Frequent submap discovery. *Combinatorial Pattern Matching, Springer Berlin Heidelberg*, 2011.

[15] P. Lienhardt. Topological models for boundary representation: a comparison with n-dimensional generalized maps. *Computer-Aided Design 23(1)*, 1991.

[16] F. Sbastien and Luc Brun. Efficient encoding of nd combinatorial pyramids. *International Conference on Pattern Recognition (ICPR)*, 2010.

[17] T. Wang, G. Dai, B. Ni, D. Xu, and F. Siewe. A distance measure between labeled combinatorial maps. *Computer Vision and Image Understanding*, 116(12):1168 – 1177, 2012.

Author Index

Albouy-Kissi Adélaïde *6*
Ali Abder-Rahman *6*
Aoki Terumasa *27*

Bischof Horst *81*
Boben Marko *110*
Boire Jean-Yves *6*
Borovec Jiří *14*
Bukovec Marko *75*
Busch Wolfgang *43*

Cehovin Luka *59*
Chen Yunjin *19*
Cuceloglu Ilkhan *89*

Donner Rene *35*

Fernandez Gustavo *59*

Goto Shinichi *27*
Grand-brochier Manuel *6*

Hegedic Matjaz *95*
Heipke Christian *51*
Hlavac Vaclav *103*
Holzer Markus *35*

Janusch Ines *43*

Klinger Tobias *51*
Kristan Matej *59*, *95*, *110*
Kropatsch Walter *43*, *118*
Kybic Jan *14*

Lenc Karel *67*
Leonardis Ales *59*, *110*
Likar Bostjan *75*

Matas Jiri *59*, *67*
Mehle Andraž *75*
Mishkin Dmytro *67*

Nebehay Georg *59*

Oberweger Markus *81*
Ogul Hasan *89*

Pernuš Franjo *75*
Pflugfelder Roman *59*
Pock Tomas *19*
Porikli Fatih *59*
Prusa Daniel *103*

Racki Domen *95*
Ranftl René *19*
Rottensteiner Franz *51*

Skocaj Danijel *95*
Stria Jan *103*

Tabernik Domen *95*, *110*
Tomaževič Dejan *75*
Torres Fuensanta *118*

Vacavant Antoine *6*
Vojir Tomaš *59*
Vrecko Alen *95*

Wendel Andreas *81*