CVWW 2010

Proceedings of the 15th Computer Vision Winter Workshop

February 3–5, 2010
Nové Hrady, Czech Republic

Libor Špaček and Vojtěch Franc (eds.)

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

Prague 2010
Proceedings of the Computer Vision Winter Workshop 2010

Editor
Libor Špaček and Vojtěch Franc
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
xfrancv@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 15th CVWW 2010 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

Libor Špaček and Vojtěch Franc (editors).
Czech Society for Cybernetics and Informatics.
Prague, February 2010. 115 pages.
Design, ConTeXt and \LaTeX styles by Vít Zýka (http://typokvitok.com)
\TeX Gyre fonts.

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 15th CVWW 2010 was organized jointly by the Czech Society for Cybernetics and Informatics and the Center of Machine Perception at CTU Prague. It was held in Nové Hrady, Czech Republic, February 3–5, 2010.

Besides papers selected by the review process, three invited talks were also included. We would like to express our thanks to Francis Bach, INRIA, Anthony Cohn, University of Leeds, and Manfred Prantl, Alicona Imaging GmbH, 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.

Libor Špaček
Vojtěch Franc
Prague, January 2010
High-Dimensional Non-Linear Variable Selection through Hierarchical Kernel Learning
*Bach Francis* (invited talk) ................................................................. 5

Learning Functional Object Categories via Mined Event Classes
*Cohn Anthony* (invited talk) .............................................................. 6

Alicona—a (Computer) Vision Became Reality
*Prantl Manfred* (invited talk) .............................................................. 6

3D Camera Tracking in Unknown Environments by On-line Keypoint Learning
*Wohlhart Paul, Roth Peter M., Bischof Horst* ........................................ 7

An Algebraic Approach to Upgrade a Preliminary 3D Reconstruction Based on Distance Constraints
*Schilling Tanja, Pajdla Tomáš* ................................................................. 15

Co-occurrence Bag of Words for Object Recognition
*Bhatti Naeem A., Hanbury Allan* .......................................................... 21

Efficient Dimensionality Reduction Using Random Projection
*Sulić Vildana, Perš Janez, Kristan Matej, Kovačič Stanislav* ..................... 29

Extended Set of Local Binary Patterns for Rapid Object Detection
*Trefný Jiří, Matas Jiří* .............................................................................. 37

Extrinsic Autocalibration of Vehicle Mounted Cameras for Maneuvering Assistance
*Ruland Thomas, Pajdla Tomáš, Krüger Lars* ............................................ 44

Facade Segmentation from Streetside Images
*Wendel Andreas, Bischof Horst* ............................................................... 52

Geodesic Adaptive Support Weight Approach For Local Stereo Matching
*Hosni Asmaa, Bleyer Michael, Gelautz Margrit, Rhemann Christoph* ........... 60

Irregular Laplacian Graph Pyramid
*Shao Dan, Kropatsch Walter G.* .............................................................. 66

Multi-View Integration for a Rotating 3D Object
*Mateos Luis A., Kropatsch Walter G.* .................................................... 72

Multivariate Online Kernel Density Estimation
*Kristan Matej, Leonardis Aleš* ............................................................... 77

Representing the Surface of Objects by Combinatorial Pyramids
*Antúnez Esther, Molina-Abril Helena, Kropatsch Walter G.* ....................... 85

Taking in Shape: Detection and Tracking of Basic 3D Shapes in a Robotics Context
*Richtsfeld Andreas, Mörwald Thomas, Zillich Michael, Vincze Markus* ........ 91

Using Outliers in Structure and Motion Analysis to Reconstruct Foreground Motion
*Holzer Peter, Pinz Axel* ........................................................................... 99

Weighted Skin Color Segmentation and Detection Using Graph Cuts
*Khan Rehanullah, Hanbury Allan, Stöttinger Julian* ................................. 107

Author Index
High-Dimensional Non-Linear Variable Selection through Hierarchical Kernel Learning

Bach Francis

INRIA—WILLOW Project-Team
Laboratoire d’Informatique de l’Ecole Normale Supérieure
(CNRS/ENS/INRIA UMR 8548)
23, avenue d’Italie, 75214 Paris, France
francis.bach@inria.fr

We consider the problem of high-dimensional non-linear variable selection for supervised learning. Our approach is based on performing linear selection among exponentially many appropriately defined positive definite kernels that characterize non-linear interactions between the original variables. To select efficiently from these many kernels, we use the natural hierarchical structure of the problem to extend the multiple kernel learning framework to kernels that can be embedded in a directed acyclic graph; we show that it is then possible to perform kernel selection through a graph-adapted sparsity-inducing norm, in polynomial time in the number of selected kernels. Moreover, we study the consistency of variable selection in high-dimensional settings, showing that under certain assumptions, our regularization framework allows a number of irrelevant variables which is exponential in the number of observations. Our simulations on synthetic datasets and datasets from the UCI repository show state-of-the-art predictive performance for non-linear regression problems.
Learning Functional Object Categories via Mined Event Classes

Cohn Anthony

School of Computing, University of Leeds
Leeds, LS2 9JT, United Kingdom
a.g.cohn@leeds.ac.uk

In this talk I will present ongoing work at Leeds, in collaboration with Krishna Sridhar and David Hogg on inducing functional object categories from (qualitative) spatio-temporal descriptions of the participating objects. First we mine event classes by building a complete activity graph of the video, represented as a layered graph of qualitative spatial and temporal relations, and then induce event classes from this, in an unsupervised manner. This makes use of a generative event model which will be described. Having formed these event classes, object categories can be formed by clustering those objects which take the same roles in a particular event. We have experimented with these techniques in two domains: a kitchen scenario, and aircraft turnovers.

Alicona—a (Computer) Vision Became Reality

Prantl Manfred

Alicona Imaging GmbH—Optische 3D Mess und Prüftechnik
Teslastraße 8, 8074 Grambach/Graz, Austria
manfred.prantl@alicona.com

We will talk about the way from doing a Ph.D. in computer vision to running Alicona, our experiences, milestones, successes, hurdles and so on.
Abstract

In this paper we present an efficient algorithm for camera tracking applicable for mobile devices. In particular, the work is motivated by the limited computational power and memory, precluding the use of existing methods for estimation of the 6-DoF pose of a mobile device (camera) relative to a previously unknown planar object. Similar to existing methods, we introduce a keypoint based approach. We establish a relationship between the object and its image by selecting keypoints on the object, preferably such with a distinctive appearance, and identifying their location within subsequent images. In contrast to existing works, we solve the problem of re-identifying such feature points by robustly learning their appearance with an on-line learning algorithm. We demonstrate the proposed algorithm, hence not limited to this application, in the context of AR. In particular, we give several qualitative and quantitative evaluations showing the benefits of the proposed approach.

1 Introduction

For many applications in computer vision and graphics the exact position and orientation (together called the pose) of a moving camera, relative to the depicted objects, has to be estimated. This process of continuously recovering the camera’s pose is known as visual camera tracking. The majority of successful methods are based on local feature points, i.e., they try to identify points that can be robustly re-identified in different views. In this way, typical problems such as partial occlusions or specular reflections, that make detection of parts of the object impossible, can be handled by redundancy.

The first steps are identifying interesting feature points and describing them such that a re-identification under slightly different conditions (viewing angle, lighting, etc.) is possible. Next, when a set of correlations of 2D points in the current image and 3D points in the world is established and the geometrical constellation of the points (i.e., their coordinates) on the object is known, the correct pose (matching those two sets of points) can be estimated. This is called the Perspective-n-Point problem.

Essentially, there are two backgrounds from which most of the camera tracking approaches emerged, namely Structure from Motion (SfM) and Simultaneous Localization and Mapping (SLAM). SfM is mainly targeted at reconstructing a perfect model in an off-line manner, where all images are available and processing time is only of limited importance. Thus, bundle adjustment can be used to achieve very accurate results [18, 2, 13, 4, 24]. Such systems are used, for instance, in the post production of movies. This line of work is clearly not suited for applications in real-time, low resources domains. SLAM, on the other hand, was originally developed for robot self localization. Thus the goal is to build and constantly refine a rough model of the environment, just detailed enough to fulfill tasks such as navigation planning with obstacle avoidance, and at the same time always have a good estimate of the current pose. As pointed out in [1], there are still differences in the prerequisites for traditional SLAM algorithms used in robot navigation and those of camera tracking for hand-held mobile devices. The main difference is that the movement of a hand-held camera can be very fast and unpredictable and thus no reliable motion models can be constructed. Additionally, motion blur and occlusions might degrade the image, thus, one has to deal with situations where tracking fails and has to be recovered without any external support.
This problem gets more difficult, if we want to track a camera in a previously unknown environment. Hence, the system has to identify interesting local feature points and to learn their appearance for robust re-identification. Additionally, the initially unknown virtual coordinate system used to render virtual objects, should be oriented and scaled relative to the target. Thus, we have to specify the initial virtual coordinate system at startup. In our case this is realized by user interaction. Furthermore, since no predefined geometrical model of the target object is available, the use of model based approaches (e.g., [7, 11, 20, 17]) is prohibited. Instead the location of chosen keypoints on the object with respect to the virtual coordinate system (i.e., 3D world coordinates) must be calculated. For the first frame, with the initial pose given by the user, this is straight forward. But during operation, when certain points fail to be tracked or become invisible, new keypoints have to be initialized for tracking and their 3D coordinates have to be calculated from estimates of the current pose (subjected to noise, in a robust manner).

This process of continuously estimating the pose, and at the same time constructing a model of the environment, is known as simultaneous localization and mapping (SLAM) or as parallel tracking and mapping (PTAM) [3, 9, 27, 1]. The method combines the advantages of accurate model building with a probabilistic framework, constructing a full covariance matrix of all model points and optimizing a robust cost function with bundle adjustment, while still tracking in real-time. This is achieved by splitting the two processes apart. Tracking is performed for every frame on a small subset of the 3D map constructed up to this point. Map building is deferred to a background thread, only triggered for keyframes, which are selected with a certain distance in time and only if the current pose is robust and shows a significantly different view of the scene.

One application that requires highly accurate camera pose estimation is mobile augmented reality (AR). In general, augmented reality systems allow the user to experience the real environment augmented by artificially computer generated objects. The user can see the virtual objects as if they were present in the real world. To give the user the consistent impression of a virtual object really living in the natural space, it is absolutely crucial that the images of the virtual objects are well aligned with the environment (i.e., well registered), depending on the user’s current point of view. Thus, a strong requirement on the algorithms in this field is real-time capability under limited computational resources. Although mobile hand-held devices evolve rapidly and components for fast floating-point operations and even dedicated graphics hardware with good performance are integrated into these systems, the power of a fully equipped desktop computer is not at hand.

Hence, the goal of this work was to develop an efficient camera tracking method that can be applied within the context of mobile AR. Figure 1 illustrates such a system in action. An arbitrary planar target defines the ground plane and is augmented by a virtual interactive soccer field. The small cubes show the locations of tracked object features. The color indicates if the feature was found in the current image and its position was geometrically consistent with the others (green/light) or not (blue/dark); only the green cubes are used for pose estimation.

Even though recent advances in mobile device hardware make it possible to use systems such as PTAM for mobile AR, we want our system to run on platforms lacking the power to run background jobs and to perform affine warps of keypoints for template matching. To overcome these limitations, more efficient trackers can be applied. For instance, in [25] the authors show that by using a combination of a modified version of SIFT and FERNS [16] very fast and stable tracking results can be achieved on standard low cost hand-held devices like mobile phones. However, the target objects have to be known in advance for off-line keypoint classifier learning. Similarly, recently an algorithm for tracking planar textured objects by its natural features has been proposed [6]. It defines tracking as a classification problem, learning classifiers for each point to track on-line. Compared to methods utilizing image patches or other keypoint descriptors such as SIFT [12, 10] this method is especially interesting because of its low performance and memory requirements.

Since the tracker presented in [6] is computationally very efficient even on low performance hand-held devices, it is highly interesting for our target application, i.e., mobile AR. However, to meet the requirements of our application, several extensions are required. First, a mapping of tracked keypoints to a virtual coordinate system to deliver a global camera pose estimate relative to the target object has to be established. Additionally, amendments were made to the process of searching for object features and the policy of exchange of features. Finally, the whole system was integrated into the OpenTracker framework, to serve as pose estimation module in the Studierstube Augmented Reality framework.

The rest of the paper is organized as follows. Section 2 gives an overview on the algorithms building the basis for our approach. Section 3 gives a short overview of the complete system’s design and a detailed explanation of the tracking process. In Section 4 qualitative and quantitative experiments analyze the system’s capabilities, and finally, Section 5 gives a short discussion and conclusion.

## 2 Camera Tracking and Pose Estimation

In the following, we give an overview on the algorithms building the basis for our work, i.e., on-line learning for feature tracking and stable and efficient pose estimation.

### 2.1 Tracking

The work at hand is built on the tracker presented in [6], which is able to robustly track a large variety of textured planar objects. In our case, the re-identification of feature points can be formulated as a classification problem between successive frames. As a first step, interest points are extracted from the current frame (e.g., Harris corners [8]) and a subset of fixed size is selected (preferably uniformly distributed over the target object). For each of them a classifier is trained to distinguish a small patch around the keypoint from the rest in the image (i.e., other keypoints on the object.

---

as well as in the background). For training the classifiers Online Boosting for Feature Selection [5] is used. Tracking is then performed by classifying all interest points in the new frame and selecting the one with highest confidence as a candidate match for the associated keypoint in the last frame. These candidates are verified for geometric consistency by robustly estimating a homography between the constellations in the two successive frames (using RANSAC). This allows for robust re-detection of points with slightly changed appearance in the next frame and subsequently incorporation of the new appearance by on-line learning, thus adapting to context specific transformations of the patch. In contrast to [14] the matching of keypoints is more robust and faster and can therefore be applied on the whole image instead of just a small surrounding. Thus, allowing for very fast camera movement and fast recovery from tracking failure due to strong motion blur or full occlusion.

Additionally, a feature exchange mechanism increases the adaptivity. If one of the initially chosen features cannot be re-detected reliably over time, the system can discard such a feature and initialize a new classifier for a different interest point. For that purpose, a simple confidence measure is calculated, estimating the probability to re-detect each feature in the next frame from its detection history:

$$P_{i,t+1} = \beta \cdot P_{i,t} + (1 - \beta) \cdot \delta_i,$$

where $P_{i,t+1}$ is the probability to re-detect feature point $i$ in the next frame; $\delta_i$ is 1 if it was detected in the current frame; 0 otherwise; $\beta \in \mathbb{R}$, $0 \leq \beta \leq 1$, determines the influence of the feature point's history.

## 2.2 Pose Estimation

The natural feature tracker described above successfully tracks the location and orientation of planar objects throughout a video, but it does not establish a virtual coordinate system and thus also does not calculate a 6-DoF camera pose relative to it. The frame to frame homographies implicitly encode the camera motion between frames and concatenating the homographies would thus give a pose relative to the first frame. But since every homography is estimated, such a scheme would sum up the individual errors and make it susceptible to drift. Furthermore simple plane to plane homography calculation does not take into account internal camera parameters available from calibration, but includes them in the estimation procedure. Altogether recovering the pose directly from a summed up homography would not deliver the accuracy and stability of registration needed in AR.

When internal camera parameters are available, image points can be transformed into the coordinate system of an ideal pinhole camera, and finding the pose that matches the correspondences of 3D points in the world and their 2D projections on the current camera’s image plane is known as the Perspective-n-point (PnP) problem. Formally, the problem is stated as follows: Points on the object with the three dimensional coordinates $M_i = (X_i, Y_i, Z_i)$ are being projected perspective to the points $m_i = (u_i, v_i)$ on the image plane ($i = 1, \ldots, n$) by the $3 \times 4$ projection matrix $P$

defined up to scale $s$:

$$s \tilde{m}_i = PM_i.$$

In the case of a perspective camera $P$ can be further decomposed into

$$P = K [R|t],$$

where $[R|t]$ is the composition of a $3 \times 3$ rotation matrix $R$ and a $3 \times 1$ translation vector $t$, describing the rotation and translation of the world coordinate system to the camera coordinate system, i.e., the camera’s pose. They are also called the camera’s external parameters.

$K$ is the camera calibration matrix, holding the camera’s internal parameters, describing how points expressed in the camera coordinate system are projected onto the image plane:

$$K = 
\begin{pmatrix}
  f_u & s & u_0 \\
  0 & f_v & v_0 \\
  0 & 0 & 1
\end{pmatrix},
$$

where $f_u$ and $f_v$ are the camera’s focal length multiplied by the pixel resolution along the u- and v-axis of the image; $u_0$ and $v_0$ define the image origin’s offset to the camera’s principal point (the point where the optical axis intersects the image plane); $s$ is the skew factor usually being 0 unless the camera’s u- and v-axis are not perpendicular. In our case, these parameters were estimated using publicly available software².

Consequently the task is to find $P$ (i.e., given $K$, find $[R|t]$) that satisfies Equation (2) for a given set of point correspondences $m_i \leftrightarrow \tilde{m}_i$. Since in real world measurements we always have to deal with noise, in general, there will be no solution fitting all points perfectly, and one has to search for a $P$ that fits best, i.e., minimizes an error function. Essentially, either the image space error $E_{ias}$ or the object space error $E_{ios}$ can be taken. $E_{ias}$ describes the total residue of projections of the $M_i$ onto the image plane by $P$, to the actual measured $m_i$. $E_{ios}$ sums up the perpendicular distance of $M_i$ to rays coming from the camera defined by $P$ and going through $m_i$. Minimizing $E_{ias}$ would be preferable for AR because it directly defines how well the final rendering is aligned to the image, unfortunately, it is very hard to efficiently formulate the PnP problem in terms of $E_{ias}$, resulting in that most methods optimize $E_{ios}$.

Solving the PnP problem has been investigated thoroughly and various methods exist with different computational complexity, speed and stability of convergence, robustness, and accuracy. For our purpose, we are especially interested in the case where the world coordinates of the object are coplanar. Since this is a special case, leading to singularities in most mathematical formulations. For many algorithms there exist special extensions for this case (e.g., [15]). Particularly challenging effects in the planar case with noisy measurements are pose ambiguities arising in certain configurations (distant camera or steep viewing angle). In [22], an algorithm is presented that pays special attention to those cases and basically iteratively refines both possible solutions until one is more probable and thus avoids

²Camera Calibration Toolbox for MATLAB: http://www.vision.caltech.edu/bouguetj/calib_doc/
randomly choosing one solution for each frame, making the final outcome jump around. Since this method delivers very accurate results and an efficient implementation is available, we use it to solve the pose estimation task in this work.

3 NaturalFeatureTracker for AR

In the following, we give a detailed overview of our natural feature tracker built on the algorithms discussed in Sections 2.1 and 2.2 and show how it can be included into an AR system.

3.1 NaturalFeatureTracker

The following is an outline of the process of tracking and pose estimation within the NaturalFeatureTracker (NFT). In the beginning, as mentioned above, the system needs to be initialized with a definition of the target object to track and the ground plane of the virtual coordinate system relative to it in the first frame. When the user decides to start, the current frame is frozen, and the user clicks the four corner points of the rectangular planar target region. The rest of the system does not rely on a rectangular target area, but it gives a convenient way to enter the initial pose and scale. The aspect ratio of the target area is not predefined, but can be calculated along with the initial pose. Every new frame arriving is first preprocessed by a conversion to gray-scale and smoothing with a Gauss kernel for noise reduction. Subsequently Harris-corners [8] are extracted as interest points (however, any other interest point detector for which fast implementations exist could be used).

Feature Point Matching On the first frame, classifiers are trained to re-detect feature points in follow-up frames. Using the target region the keypoints found in the image are split into object and background keypoints. For a fixed number of keypoints on the object, chosen such as to cover most of the area, classifiers are initialized by learning the patch around the respective location as positive; patches around all other object keypoints or those in the background are learned as negative. Starting from the second frame, classifiers for a set of feature points have already been initialized and can then be used to re-identify them in the current image. This is done by evaluating every classifier on all candidate keypoints in the new frame and choosing the one with the highest confidence.

Since this mapping might include false matches, a (RANSAC style) verification step is introduced. A series of random subsets of 4 point correspondences is created and for each of them a homography is calculated, mapping the locations of the points in the previous frame to the current ones. By applying each homography to all points in the last image and comparing the distance of the mapped points to the locations of the re-identified matches, inliers supporting the homography can be found. The homography with most inliers and smallest total error (distances) is chosen as the correct one. Only inliers of the best homography are considered as true matches and used further on.

Feature Classifier Update The classifiers of all positively re-identified feature points are updated with a positive patch around the new location and a random negative one. Thus, scene specific appearance changes of feature points are incrementally learned on-line. The others are not updated, since they might not have been found, because they currently are occluded.

Pose Estimation Taking all features for which the world coordinates and the position on the current frame are known, a pose that projects the keypoints from world coordinate system to the respective current keypoint coordinates on the image can be found by solving the PnP problem with the method described in Section 2.2. For the first frame world and image coordinates are only known for the corners of the initial object region defined by the user. For all consecutive frames matched and verified feature points for which world coordinates have been calculated before are included.

Map Building Using the current pose, world coordinates for new object features are calculated by projecting the image coordinates back onto the object plane. This calculation is repeated on a series of frames for which poses have been recovered to have a set of world coordinate estimates over time. These can be averaged (again with outlier removal) to reduce the influence of noisy pose estimates. Only if a new object feature is considered stable enough, it is included in the pose estimation.

Feature Exchange As a final step the quality of the tracked object features is evaluated. Features that have not been found in a certain number of frames are dropped and replaced by new ones initialized on a new interest point on the object. Since for accurate pose calculation it is beneficial to have matched feature points not clustered on one spot in the image but spread over the entire object, the target object region is split into a 3x3 grid and new keypoints are selected preferably in cells with few tracked feature points.

One improvement to speed up the process of feature point recognition is to reduce the search area. In this way, not every classifier has to be matched to every interest point. A common way is to use an EKF (Extended-Kalman-Filter) framework with a motion model to predict the locations of the object features in the next image. This works especially well for robot navigation tasks, where accurate motion models can be constructed and often some kind of control and odometry data is available. Since camera motion of hand-held cameras cannot be estimated very well, it is a safe bet to take the simplest model, which is to just assume small motions with equal probability for all directions. Thus, we define a small radius around the locations of feature points in the previous frame as the search area. Only if no match can be established or too few matches remain as inliers of the best homography, the search radius is increased until, if necessary, the whole image is searched again (tracking loss recovery).

3.2 AR System

The overall AR system including the previously introduced tracker, based on the Studierstube framework, is shown in Figure 2. The video capturing module OpenVideo acquires the camera’s current picture, which is sent to the Studierstube core and forwarded into its event system, which is responsible for all interactions, and holds an instance of OpenTracker for tracking all moving artefacts relevant to the AR application. On the arrival of the image, OpenTracker notifies all
subcomponents that registered for video events. The natural feature tracker developed for this work was implemented as a plug-in module for OpenTracker. It receives the image, calculates the pose and returns it to OpenTracker, from where it is propagated to the Studierstube core, which makes it possible to use it in the scene graph as the pose of the virtual camera, used to render the augmented scene on top of the camera’s image.

Figure 2: Dataflow within the Studierstube system with NaturalFeatureTracker for camera pose estimation: from image acquisition to pose calculation and final output of the augmented image

4 Experimental Results

Since the target application of this work is mobile augmented reality, we first present some qualitative results, showing the natural feature tracker working in two AR scenarios. Figure 3 shows the system, augmenting an ortho-photo of the Jakominiplatz at the center of Graz with a 3D model of the place’s subsurface infrastructure (gas and water pipelines). The data was taken from the Vidente project, targeted at helping construction workers in planning projects [21]. Figure 4 shows an interactive soccer field, rendered on top of any kind of planar surface. Users can investigate the scene, plant trees and make balls jump by using the tracked camera as an input device.

The first quantitative evaluation investigates the accuracy of the resulting pose. We compare the pose coming from our tracker with the one returned by the marker-based tracker ARToolkitPlus [26] on a specifically designed target shown in Figure 5. The ARToolkitPlus module calculates the pose from the markers, whereas the NFT is initialized to track the image. Figure 6(a) shows the path of the camera as recorded by our tracker, while Figures 6(b) and 6(c) show the differences between the positions and orientations for each frame. The results are very similar, demonstrating the capability of the classifiers to locate the feature points and the robustness and accuracy of the pose estimation. The differences in translation remain below 0.5%. The angle between the cameras’ orientations has a mean value of 0.45° with a standard deviation of 0.27°.

In addition, to have a steerable environment for the pose and to have ground-truth, synthetic scenes were constructed. One particular challenge for pose estimation from natural features on planar targets are situations where the viewing angle gets steep. The features are perspectively maximally distorted and in the case of a 90° angle reduced to a line. We want to investigate how our on-line learners can adapt to continuous perspective distortion of the features. Figure 7(a) shows the results of the first synthetic scene, in which the target object initially faces the camera and is then rotated to the side. Multiple clips were recorded with different rotation speeds, defining the x-axis of the figures. The blue line (dots) shows the mean value and the standard deviation of the last angle at which the target was successfully tracked in a series of trial runs. However, the tracker might have failed before and recovered, indicated by the red line (crosses) showing the range of angles at which the tracker failed the first time. Figures 7(b) and 7(c) show the according errors in position and orientation for successfully tracked frames.

With the second experiment, we tested the system’s capability to track objects continuously rotating in front of the camera, around its viewing axis. If there is too much rotation, feature points get lost and feature exchange will have to kick in to continue operation. Results are shown in Figure 8. As the results show, tracking works fine up to an angle of about 60° for both kinds of rotations and rotation speeds up to 2° per frame, which makes 30° per second at a frame rate of 15 fps. One conclusion that can be drawn from both
3D Camera Tracking in Unknown Environments by On-line Keypoint Learning

Figure 3: Tracking sequence of an ortho-photo of Jakominiplatz augmented with sub-surface infrastructure (gas and electricity pipelines): (a) initial frame showing the aligned sub-surface infrastructure models and the feature points as cubes, (b) fast camera movement causes the tracker to fail on one frame, (c) tracking is recovered in one of the subsequent frames, (d) continuous tracking even when several feature points cannot be detected in the current frame (zoomed in and occluded), (e) when zooming out again after a longer period of close-up frames, the feature exchange mechanism placed all the features in the formerly visible area (feature points clustered in top right), (f) some frames later the feature points spread out again to cover the entire object.

Figure 4: Rendering an interactive animated virtual soccer field on top of different grass-textured objects: (a) and (b) different grass textures, robust tracking of the whole region or a sub-area; (c) additional virtual objects (bouncing balls and trees) can be added interactively by pointing the cross-hairs at the desired location on the ground and pressing a key.

Figure 6: Comparison of the camera poses calculated by NaturalFeatureTracker and ARToolkitPlus on a short video sequence.
experiments is that the Haar-features, forming the base of the classifiers’ decisions, perform badly on increasingly rotated patches. When pure rotation exceeds about 30° lots of feature points are lost. For slow rotations, adaptation to the rotated appearance of the patches is possible, but the discriminative power is lost and hence, the effect of learning has a smaller impact on the angle up to which the target can be tracked, than the initialization of new features gradually replacing old ones that cannot be found anymore.

The second observation is that the accuracy of the resulting pose drops as soon as the tracker lost the object in one frame, even if the tracker finds enough correct matches again. This is illustrated in Figures 7(b), 7(c), 8(b) and 8(c). The green lines (dots) show the mean and standard deviation of errors in position and orientation for all frames, from the beginning to the point when tracking starts to fail. For small rotation speeds the translation error is usually around 1% of the target object’s width and the error of the rotation angle is about 1°. The blue lines (crosses) however show the errors in position and orientation when tracking was recovered after failing on one frame. The poses returned in those cases are not really suitable for AR purposes any more. The same applies for fast camera rotation (about 10° per frame) already on consistently tracked frames.

Finally, we evaluate the performance in terms of computation time. As the primary target platform an ultra-mobile PC (UMPC): Sony Vaio VGN-UX50 (1.06 GHz) was chosen. The built-in rear camera’s resolution was set to 320x240. Additional experiments were conducted on the development platform: Sony Vaio VGN-FE11M, Intel Core2 Duo (1.66 GHz). Apart from the slightly faster CPU clock speed, the laptop has the advantage of the dual core architecture, allowing the multi-threaded Studierstube framework to basically dedicate one core to the tracker only. Here the camera is a standard WebCam and with a resolution of 640x480. Table 1 shows the profiling data of two experiments on both platforms. The most essential observation is, that without restriction of the search area, the matching of all classifiers to all interest points is the dominating performance bottleneck. With search area restriction in place, it becomes a minor factor and the remaining costly tasks are the extraction of interest points and the update of the classifiers. For the former, faster methods could be considered (e.g. [19]), the latter is constitutive to this approach, but could be sped up a little, by working on some implementation issues. Summing it up, one can see that interactive frame rates can be achieved even on the UMPC, if some form of search area restriction is applied.

<table>
<thead>
<tr>
<th>Task</th>
<th>UMPC</th>
<th>Laptop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Img. cap. &amp; prep.</td>
<td>7 (ms)</td>
<td>15 (ms)</td>
</tr>
<tr>
<td>Interest Points</td>
<td>15 (ms)</td>
<td>31 (ms)</td>
</tr>
<tr>
<td>Feat. Matching</td>
<td>108 (ms)</td>
<td>46 (ms)</td>
</tr>
<tr>
<td>Homography</td>
<td>5 (ms)</td>
<td>2 (ms)</td>
</tr>
<tr>
<td>Feat. Update</td>
<td>22 (ms)</td>
<td>10 (ms)</td>
</tr>
<tr>
<td>Feat. Exchange</td>
<td>0.5 (ms)</td>
<td>0.5 (ms)</td>
</tr>
<tr>
<td>Pose est.</td>
<td>3 (ms)</td>
<td>1.2 (ms)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>166 (ms)</td>
<td>111 (ms)</td>
</tr>
</tbody>
</table>

Table 1: Timing of the algorithm’s steps per frame.

5 Discussion and Conclusion
For many applications in computer vision, the exact estimation of the 6-DoF pose of the camera, has to be estimated, which is referred to as camera tracking. One prominent application is mobile AR (in unknown environments). However, due to limited computational power and memory on mobile devices state-of-the-art approaches such as SLAM or SfM can not be applied. Hence, in this paper we proposed an efficient camera tracking approach that is highly applicable for mobile devices. The key idea is to apply an on-line learning method to robustly identify keypoints by adaptively learning scene specific representations. Hence, a real-time capable natural feature tracker can be derived, which together with a robust pose estimation builds the basis for the camera tracker. Since our target application was
mobile AR, the tracker was integrated into the Studierstube AR framework, where OpenTracker and our NaturalFeatureTracker submodule are used to determine the position and orientation of the tracked camera. The experimental results show promising results of the working system under realistic conditions. Interactive frame rates can be reached and the registration of virtual content to the scene shown in the camera’s image show good visual results. Future work will include finding better mechanisms for feature exchange and including new features which can handle rotations considerably better.

Acknowledgement

The work was supported by the FFG project MDL (818800) under the Austrian Security Research Programme KIRAS. In addition, the authors want to thank Helmut Grabner and Gerhard Schall for their support.

References

An Algebraic Approach to Upgrade a Preliminary 3D Reconstruction Based on Distance Constraints

Tanja Schilling and Tomáš Pajdla

Center for Machine Perception, Department of Cybernetics
Faculty of Electrical Engineering, Czech Technical University in Prague
schilling@cmp.felk.cvut.cz, pajdla@cmp.felk.cvut.cz

Abstract In this paper, we propose an approach to compute a transformation matrix algebraically that upgrades a preliminary 3D reconstruction. Constraints are derived from segments of equal lengths in the images and yield a set of polynomial equations which we try to solve by means of Gröbner bases. We show how this can be achieved for a special case and explain why the same procedure is supposed to be applicable to the general problem.

1 Introduction

Since the introduction of commercially available robotic surgery systems about ten years ago, robot assisted minimally invasive surgery has become increasingly popular. With the widespread use of such systems, pre- and intraoperative image guidance has gained importance and poses challenging problems to computer vision. Within that scope, accurate and robust 3D reconstruction and tracking of deformable tissue is a fundamental but difficult task, because such surfaces often lack distinctive features and detailed texture or they are partially covered by liquids.

Various approaches to 3D tissue deformation recovery have been published [12, 14, 11]. Some of them apply optical markers and thus require special surgical equipment to cope with the mentioned issues. But as space is strongly limited in this environment, methods without the need of extra instrumentation are preferable.

On the other hand, the instruments used for the particular surgical procedure are almost all the time present in the images. They are rigid and mostly well detectable. Consequently, it seems useful to benefit from these properties in order to improve the tissue reconstruction. However, relying on exact dimensions of the tools is not advisable, because detailed specifications of commercial items are hardly to get and the instruments are changed quite frequently. But often, various scissor-like instruments are applied whose two legs have equal lengths. From this fact a constraint could be derived in order to upgrade a preliminary reconstruction or the projection matrix.

The approach in [10] is based on that idea. Therein, the authors introduce a method to reconstruct dynamic articulated structures from multiple uncalibrated views assuming affine cameras. In particular, they aim to track human motion in sports broadcasts. An affine 3D reconstruction is computed first. Constraints are derived from the fact that the lengths between rotational joints of the body remain constant over time. As parallel projection is assumed, these equations are linear and can be solved via singular value decomposition. Although the affine camera model is a reasonable assumption in recordings of sport events where long focal lengths are used, it is not applicable to the case of endoscopy where the observed surface is located very close to the camera.

In this paper, we present ongoing work to find algebraically a transformation to upgrade a reconstruction based on constraints derived from equally distant pairs of points. The following section gives a description of the problem. Section 3 briefly introduces the notion of Gröbner bases. In section 4, we explain how this can be used to solve a special case and what that implies to the general problem. We give an overview of the experiments we have performed so far in section 5 and discuss it in section 6.

In addition, appendix A illustrates the problem by a simple numerical example and appendix B clarifies some frequently used terms in the paper.

2 Problem formulation

Let the image projection of a scene point $X_i$, represented by its homogeneous coordinates, be

$$\alpha_i x_i = PX_i\quad (1)$$

Let us assume that we have measured image coordinates $\hat{x}_i, \hat{\alpha}_i \hat{x}_i \neq \alpha_i x_i$, and computed $\hat{P}, \hat{P} \neq \beta P$, as well as $\hat{X}_i$, $\hat{X}_i \neq \gamma_i X_i$, from those points, such that

$$\hat{\alpha}_i \hat{x}_i = \hat{P} \hat{X}_i\quad (2)$$

Now, we wish to find a non-singular transformation matrix $H \in \mathbb{R}^{4 \times 4}$ which upgrades $\hat{X}_i$ and $\hat{P}$, such that

$$PX_i \propto \hat{P}H^{-1}X_i\quad (3)$$

and

$$X_i \propto H\hat{X}_i.\quad (4)$$

In order to fix the reference frame, $H$ should map the point of origin to itself, $(0, 0, 0, 1)^\top \propto H(0, 0, 0, 1)^\top$, points on the $x$-axis to the $x$-axis, $(x', 0, 0, 1)^\top \propto H(1, 0, 0, 1)^\top$, and points in the $xy$-plane the $xy$-plane again, $(x', y', 0, 1)^\top \propto H(1, 0, 0, 1)^\top$.
An Algebraic Approach to Upgrade a Preliminary 3D Reconstruction Based on Distance Constraints

\( \mathbf{H}(x_j, y_j, 0, 1) \). Consequently, in such a coordinate frame, the transformation matrix has to be of the form

\[
\mathbf{H} = \begin{pmatrix}
    h_1 & h_2 & h_3 & 0 \\
    0 & h_4 & h_5 & 0 \\
    0 & 0 & h_6 & 0 \\
    h_1 & h_7 & h_8 & h_9
\end{pmatrix}
\]

and since it has to be invertible, it must fulfill

\[ 0 \neq \det(\mathbf{H}) = h_1 h_4 h_5 h_6. \]

Let us now assume that there are two pairs of points, \((X_{i_1}, X_{i_2})\) and \((X_{i_3}, X_{i_4})\), with equal Euclidean distances between the points of each pair \( \|X_{i_1} - X_{i_2}\| = \|X_{i_3} - X_{i_4}\| \), and the indices \( i_1, i_2, i_3, i_4 \), where \( i_1 \neq i_2, i_1 \neq i_3, i_1 \neq i_4 \) and \( i_3 \neq i_4 \) are, known to us. Replacing \( X_i \) by \( \mathbf{H} \hat{X}_i \), yields the following constraint on \( \mathbf{H} \):

\[
0 = (\mathbf{H}_4 \hat{X}_{i_3})^2(\mathbf{H}_4 \hat{X}_{i_4})^2,
\]

\[
\sum_{k=1}^{3} (\mathbf{H}_k \hat{X}_{i_3} \mathbf{H}_4 \hat{X}_{i_2} - \mathbf{H}_4 \hat{X}_{i_3} \mathbf{H}_k \hat{X}_{i_2})^2 - (\mathbf{H}_4 \hat{X}_{i_3})^2(\mathbf{H}_4 \hat{X}_{i_4})^2,
\]

\[
\sum_{k=1}^{3} (\mathbf{H}_k \hat{X}_{i_3} \mathbf{H}_4 \hat{X}_{i_2} - \mathbf{H}_4 \hat{X}_{i_3} \mathbf{H}_k \hat{X}_{i_2})^2
\]

where

\[
\mathbf{H}_k \hat{X}_i \neq 0, \quad \forall i \in \{i_1, i_2, i_3, i_4\}
\]

and \( \mathbf{H}_k \) denotes the \( k \)-th row of \( \mathbf{H} \).

Equation 7 constitutes a homogeneous polynomial of degree 8 in 9 variables, with 808 terms in the general case. Therefore, at least 9 quadruples \((X_{i_1}, X_{i_2}, X_{i_3}, X_{i_4})\) are required, to obtain the 9 equations which determine \( \mathbf{H} \). Introducing two more variables \( h_{10} \) and \( h_{11} \), we can rewrite the inequalities 6 and 8 as equalities [5]

\[
0 = 1 - h_1 h_4 h_5 h_6 h_{10}
\]

\[
0 = 1 - h_{11} \prod_{i=1}^{n} \mathbf{H}_4 \hat{X}_i
\]

Now, the problem is to solve the system of \( m = m' + 2 \) algebraic equations, \( m' \geq 9 \), in 11 variables.

\[ 0 = f_1(\mathbf{x}) = \cdots = f_m(\mathbf{x}) \]

3 Gröbner bases

Systems of polynomial equations can be solved by means of Gröbner bases [7]. \( F \) denotes the set of \( m \) polynomials \( F = \{ f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}) \} \) in \( K[x_1, \ldots, x_n] \) in \( n \) variables \( \mathbf{x} = (x_1, \ldots, x_n) \) over a field \( K \). The ideal \( I = \langle F \rangle \) generated by \( F \) is the set of all polynomial linear combinations: \( I = \left\{ \sum_{i=1}^{m} f_i(\mathbf{x}) p_i(\mathbf{x}) \mid p_i(\mathbf{x}) \in K[x_1, \ldots, x_n] \right\} \). A Gröbner basis is a special set of generators with desirable algorithmic properties. In particular, a Gröbner basis of an ideal \( I \) has the same set of solutions as \( I \). But similar to a system of linear equations after Gaussian elimination, the solutions of \( I \) can be easily identified in the corresponding Gröbner basis w.r.t. a lexicographical monomial ordering [7].

Theoretically, the Gröbner basis can be computed from any generating set of \( I \) by a method called Buchberger’s algorithm [7]. The basic mechanism is to take each pair \((f_i(\mathbf{x}), f_j(\mathbf{x}))\) from \( F \), \( f_i(\mathbf{x}) \neq f_j(\mathbf{x}) \), compute its \( S \)-polynomial, reduce it by \( F \) and add the remainder to \( F \) if it is not zero. This is done until the \( S \)-polynomials of all pairs in \( F \) reduce to zero.

Several methods exist to improve this basic algorithm, for instance by trying to use a minimal number of critical pairs \((f_i(\mathbf{x}), f_j(\mathbf{x}))\) or optimizing the reduction algorithm. Despite all of those improvements, the computation can take very long or may be even infeasible because of memory limitations [9]. Our attempts to solve the system of polynomial equations with standard methods for Gröbner bases computations in Maple 12 [3], Macaulay2 [2] and SINGULAR [4] were not successful due to a lack of memory after some time of computation.

4 Solving a special case

Our strategy to tackle this difficult problem is based upon four ideas: Firstly, we reduce the number of unknowns in \( \mathbf{H} \) by choosing a suitable coordinate frame. Secondly, we avoid undesired solutions that lead to points at infinity. Thirdly, we use a version of Buchberger’s algorithm and a reduction algorithm that is appropriate for the given set of polynomials. The fourth idea is to analyse the given system of algebraic equations, to find a simpler case for which the computation is feasible while it still has properties needed to later extend the procedure to the general case.

The first two ideas were already formulated in section 2 and incorporated in the constraints. Considering the third idea, an approach called slimgb that was introduced in [6] and implemented in SINGULAR proved to be especially useful for the given problem. It combines several criteria to detect and avoid unnecessary pairs of polynomials \((f_i(\mathbf{x}), f_j(\mathbf{x}))\). In addition subsets of \( S \)-polynomials are reduced in parallel which avoids effects of a disadvantageous order of reduction.

Regarding the fourth idea, the first step to simplify the general problem is to use small integers instead of real numbers as point coordinates. Since equation 7 involves only multiplication, addition and subtraction, all coefficients in the resulting polynomials are integers, too. But even small integer coordinates within \([-9, 9]\) may lead to big 7-digit coefficients on average. Those values grow further during the procedure and quickly exceed the capabilities of standard computer hardware.

Hence, it is advisable to restrict the coefficients further to a finite field \( \mathbb{Z}_p \). This has consequences for selection of appropriate points from which the polynomials will be constructed.

In order to set the reference frame as mentioned in section 2, the pairs of points \(((0,0,0)^T, (d,0,0)^T)\) and \(((0,0,0)^T, (e,f,0)^T)\) need to be in the set of
points on which the constraint 7 is based. As the Euclidean distances between the points of each of the pairs have to be equal, it must be \( d^2 = e^2 + f^2 \). For any other pair \( ((x_1, y_1, z_1)^T, (x_j, y_j, z_j)^T) \), we write \((x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 = a^2 + b^2 + c^2 \). Then, \( d^2 = e^2 + f^2 = a^2 + b^2 + c^2 \) and since \( a, b, c, d, e, f \in \mathbb{Z}_p \), the pairs of points must be chosen in such a way that their difference together with the distance form Pythagorean triples \((e, f, d)\) and quadruples \((a, b, c, d)\), respectively.

The leading thought for that narrowing of the original problem is not only to be able to find a solution for a very special and simple case. Beyond that, we assume that the sequence of operations to compute the Gröbner basis is basically identical for different sets of polynomials whose coefficients may differ substantially, given that the equations in those sets \( F \) are generated in the same way and thus contain the same monomials [15]. With this assumption, we rely on the fact that Buchberger’s algorithm and its improved variants do not consider coefficients for the choice of critical pairs, the detection of unnecessary pairs or the selection of reductors.

However, coefficients make an impact on the cancellation of terms during the construction of \( S \)-polynomials and during reduction, thereby influencing the set of monomials contained in the resulting equation. This means, that two sets of polynomials which have the same monomials but different arbitrary coefficients may produce different Gröbner bases procedures. The important fact is that for the given problem, the coefficients for different sets \( F \) must originate from points that fulfill certain conditions in order to result in the same procedure.

For our special case of Pythagorean triples and quadruples, it can be easily shown that every triple of rational numbers \((q_1, q_2, q_3), q_i \in \mathbb{Q}\) can be transformed into a Pythagorean triple by multiplying with an appropriate integer factor. Although, there are triples of real numbers that cannot be transformed into Pythagorean triples, those triples can be sufficiently precisely represented by triples of rational numbers, for which Pythagorean triples exist [13]. In other words, each triple of positive rational numbers has a sufficiently precise representation as a Pythagorean triple. Of course, this applies analogously to the case of quadruples.

Hence, we conclude that the procedure of the Gröbner basis computation, and in particular the choice of critical pairs, should be identical for all sets of polynomial equations that were constructed in the same above explained way and differ only in the coordinates they originate from and the fields to which they belong.

Another issue is the comparatively long time it still takes to compute even the simplest relevant case. Notwithstanding the application of criteria to detect unnecessary pairs, construction and time-consuming reduction of many \( S \)-polynomials that are subsequently reduced to zero and thus do not contribute to the result is inevitable in the current algorithms. But if the computation sequence is basically the same for different sets of initial points then it should be possible to record the details of the procedure for such a special case that can be computed, i.e. particularly which pairs are used to construct \( S \)-polynomials and which of them are finally added to the intermediate basis. This could be used as a template for the computation of any other set of polynomial equations that has the appropriate structure, while all unnecessary \( S \)-polynomials could be omitted.

5 Experiments

We conducted our first experiments with Maple 12 which computes Gröbner bases using Faugère’s F4 algorithm [8]. It failed on the general problem with coefficients in \( \mathbb{Z} \) after some time because the required memory exceeded the available capacity.

Consequently, we simplified the problem even more than explained in the previous section, assuming known (possibly unequal) distances, setting \( h_0 = 1 \), thus omitting equation 9, reducing the number of unknowns to 9 and the maximum degree of the polynomials 7 to 4, and computing coefficients in \( \mathbb{Z}_p \). For several configurations of points, this was possible to compute within approximately 10 minutes. Because of the limited possibilities to monitor the intrinsic computation procedure in Maple, we were looking for an alternative program to compute Gröbner bases which allows to analyse the method in greater detail.

For this purpose, we tried the open-source computer algebra system SINGULAR. The default Gröbner basis method took about 15 minutes, while the slimgb algorithm computed the result in less than one minute. Hence, we extended the experiments to sets of polynomial equations as explained in section 4. In each sample, we used 17 pairs of points with equal distances between the points of all pairs, yielding 136 polynomials according to equation 7. Together with equations 9 and 10, we had 138 polynomials in 11 variables. With coefficients in \( \mathbb{Z}_p \), slimgb computed the Gröbner basis in about 5 minutes.

As SINGULAR’s source code is freely available, we have in principle the opportunity to adapt it to our needs, i.e. to log which critical pairs are chosen, which reductors are used, which \( S \)-polynomials are finally reduced to zero and the like. In practice, this is rather cumbersome because SINGULAR is a quite bulky framework and its code is only sparsely documented.

Finally, we decided to implement our own version of the proposed slimgb algorithm in C++ using CoCoALib-0.99 [1], a library for computations in commutative algebra. The motivation was on the one hand to be able to influence the computation procedure as well as to record the desired data more easily, and on the other hand to better understand the algorithm and how it actually performs the computation for the given problem. Thus, eventually verifying whether the assumption from section 4 turns out to be true.

During our experiments, we modified the original slimgb algorithm slightly. For instance, we found that the so-called exchange trick, that exchanges longer polynomials for shorter ones in order to have a reductor as short as possible, has no significant influence on the computation time in our implementation, and hence skipped that part of the method. Other changes refer to the order of the polynomials in the intermediate basis which has no impact on the result or computation time but seemed to be less suitable in the original version if
the intended purpose is to log the successively executed operations of a procedure in order to use it afterwards to compute the basis for another set of equations.

We generated 30 sample polynomial systems, each from 10 pairs of points in \( \mathbb{Z}^3 \), obtaining 138 equations as mentioned above. Since our implementation is not at all as optimized as the one in SINGULAR, it took about 75 minutes to compute one example in \( \mathbb{Z}_p \). We evaluated only a small set of examples while recording which critical pairs were used to form \( S \)-polynomials in each iteration and which of those were added to the intermediate basis. Then, this log file was postprocessed in order to solely contain those critical pairs of which the \( S \)-polynomials are added to the basis after reduction.

Then, the other 29 examples were run according to the script generated by the first one. The computation was done with coefficients in the same finite field \( \mathbb{Z}_p \) as in the generating first example. That led to correct results for all examples.

CoCoALib provides means to compute with rational numbers, and we consequently tried to run the examples according to the script in \( \mathbb{Q} \). But in this case the intermediate coefficient swell slows the computation down to such an extent that already the second of a total of 118 iterations took about 6 minutes, the third about 20 minutes and the fourth iteration was not yet finished after an hour. Anyway, the leading monomials and the number of terms of the polynomials that were added to the basis in the first 3 iterations were equal for the examples with coefficients in \( \mathbb{Z}_p \) and in \( \mathbb{Q} \). This suggests that the reductions during the procedure would further lead to the same leading monomials and therefore finally lead to the correct result.

In order to avoid this coefficient growth and to get closer to a practicable method, we tested the same procedure using floating point coefficients of limited precision. Doing this, we have to face another difficulty, namely the accumulation of approximation errors during the calculation. In our simple implementation, these errors lead very quickly to a severe distortion of the polynomials and hence prevent the further computation according to the template.

Due to these errors, terms do not cancel as in the integer case. Longer intermediate polynomials with different leading monomials, potentially having higher degrees, result from that. They are added to the basis and used for the construction \( S \)-polynomials and reduction in subsequent iterations. As a consequence of this, the reduction takes more and more time in each following iteration. We finally stopped this experiments after a few hours of computation since no expedient result could be expected. The issue of accumulating rounding errors has to be addressed next.

6 Conclusion

In this paper, we proposed an algebraic way to compute a transformation to upgrade a preliminary reconstruction by means of constraints derived from segments of equal lengths. We think, this could be useful in environments where distinct features are rare and hence 3D reconstruction by standard methods is error-prone.

This work is unfinished but we have shown that though the general problem is very tough and infeasible with standard methods, it is possible to solve it for some special cases. We assume that the actual computation procedure found with this special cases can be generalized to real world cases. Our experiments indicate that this is possible. For this purpose, a numerical stable version of the proposed method has to be developed.

Appendix A: A simple numerical example

We generate a sample set of 17 points \( X_i \) (see figure 1(a)) from the combination of the smallest Pythagorean triple \((e, f, d)\) and quadruple \((a, b, c, d)\), for which \( d^2 = a^2 + b^2 + c^2 = e^2 + f^2 \) holds, i.e.

\[
(e, f, d) = (5, 12, 13), \\
5^2 + 12^2 = 13^2 = 169 = 25 + 144, \\
(a, b, c, d) = (3, 4, 12, 13), \\
3^2 + 4^2 + 12^2 = 13^2 = 169 = 9 + 16 + 144.
\]

The resulting points are

\[
(X_0, X_1, X_2, X_3, X_4, X_5, X_6, \ldots) = \\
\begin{pmatrix}
0 & 13 & 5 & 12 & 3 & 4 & 12 & \ldots \\
0 & 0 & 12 & 4 & 12 & -12 & -3 & \ldots \\
0 & 0 & 0 & 3 & 4 & 3 & -4 & \ldots \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & \ldots
\end{pmatrix},
\]

where the distances \( \|X_i - X_0\| = d = 13 \) for all \( i, 0 < i \leq 17 \) are equal. Next, we choose an invertible matrix \( H' \)

\[
H' = \begin{pmatrix}
1 & 0 & -2 & 0 \\
0 & 1 & 2 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}, \quad H'^{-1} = \begin{pmatrix}
1 & 0 & 2 & 0 \\
0 & 1 & -2 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix},
\]

to obtain the points \( \hat{X}_i \) (see figure 1(b)) which correspond to a preliminary reconstruction as \( \hat{X}_i = H'^{-1} X_i \), such that

\[
(\hat{X}_0, \hat{X}_1, \hat{X}_2, \hat{X}_3, \hat{X}_4, \hat{X}_5, \hat{X}_6, \ldots) = \\
\begin{pmatrix}
0 & 13 & 5 & 18 & 11 & 10 & 4 & \ldots \\
0 & 0 & 12 & -2 & 4 & -18 & 5 & \ldots \\
0 & 0 & 0 & 3 & 4 & 3 & -4 & \ldots \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & \ldots
\end{pmatrix}.
\]

According to equation 7, we construct polynomials for all possible pairs \((i, j)\), \( 0 < i < j \leq 17 \), as \( \|X_i - X_j\| = \|X_j - X_0\|, \forall i, j, 0 < i < j \leq 17 \), and get 136 homogeneous polynomial equations of degree 8 in 9 variables with coefficients in \( \mathbb{Z} \) and at most 90 terms (because \((0, 0, 0, 1)^\top\)
is contained twice in each constraint), for instance
\[ f_k = 251944 h_1^7 h_2 h_3^6 - 75665 h_1^2 h_2^2 h_3^4 - 12320 h_1^3 h_3 h_9^4 + 103496 h_1^7 h_2 h_3 h_9^6 - 4880 h_1^2 h_3^2 h_9^4 - 75665 h_1^2 h_2^2 h_9^4 + \ldots + 88128 h_2 h_3 h_9^6 - 2736 h_3^2 h_9^6 - 61236 h_2^2 h_9^6 + 88128 h_4 h_9 h_9^6 - 2736 h_3^2 h_9^6 - 2736 h_2^2 h_9^6, \]
in addition from equation 9
\[ f_{137} = h_1 h_9 h_9 h_9 h_9 - 1 \]
and according to equation 10
\[ f_{138} = 1631193006432000000 h_1^{17} h_9^{17} h_{11} - 17089630165689600000 h_1^{16} h_9 h_9^{17} h_{11} + 59234872518267240000 h_1^{15} h_9^{17} h_9^{17} h_{11} - \ldots - 60043261279921920000 h_9^{34} h_{11} - 1. \]

Running the slimgb algorithm on that set of 138 polynomials, using the finite field \( \mathbb{Z}_{63223} \), yields a Gröbner bases of 284 equations from which the following reduced basis can easily be computed:
\[
\begin{align*}
&f_1 = h_9^{10} h_{11} - h_9^6 \\
&f_2 = h_9^3 h_{10} + 16611 h_9 h_6 \\
&f_3 = h_9^{10} + 16611 h_3 h_9 h_9 h_{11} \\
&f_4 = h_5 h_9 h_{10} - 2 h_6 \\
&f_5 = h_9 h_9 h_{10} + 16611 h_5 \\
&f_6 = h_5 h_9 h_9 h_{10} - 2 \\
&f_7 = h_9^2 - 4 h_9^2 \\
&f_8 = h_9^6 - h_9^9 \\
&f_9 = h_1 - h_9 \\
&f_{10} = h_2 \\
&f_{11} = h_3 + 2 h_9 \\
&f_{12} = h_4 + 16611 h_5 \\
&f_{13} = h_7 \\
&f_{14} = h_8 .
\end{align*}
\]

From those equations, we obtain the resulting matrix \( \mathbf{H} \)
\[
\begin{pmatrix}
h_9 & 0 & -2 h_9 & 0 \\
0 & s h_9 & 2 s h_9 & 0 \\
0 & 0 & t h_9 & 0 \\
0 & 0 & 0 & h_9
\end{pmatrix}, \quad s, t \in \{-1, +1\}
\]
and
\[
h_{10} = s t (h_9^5)^{-1} \\
h_{11} = (h_9^5)^{-1} .
\]

The scaling of \( \mathbf{H} \) as well as the sign of its second and third row are undetermined. Figure 1(c) illustrates the upgraded points \( \mathbf{H} \hat{X}_i \) that correspond to the different signs in \( \mathbf{H} \). For \( h_9 = s = t = 1 \), the solution corresponds to the initially generated matrix \( \mathbf{H} \) and points \( X_i \).

However, this particular example has very limited usefulness because it is hardly applicable to a more general case as in real-world problems the segments of equal length are unlikely to originate all from the same point.
Appendix B: Notation

We use the notations term and monomial as it is explained in [7], i.e. given a polynomial ring $K[x_1, x_2, \ldots, x_n]$, a monomial is a product of the form $x_1^{\alpha_1} \cdot x_2^{\alpha_2} \cdots x_n^{\alpha_n} = x^\alpha$, with non-negative integer exponents $\alpha_1, \alpha_2, \ldots, \alpha_n$. A term denotes then the product $a_\alpha x^\alpha$ of a monomial and a non-zero coefficient $a_\alpha \in K$.

An S-polynomial of a pair of polynomials $(f, g)$ is computed as

$$S(f, g) = \frac{x^\gamma}{\text{LT}(f)} f - \frac{x^\gamma}{\text{LT}(g)} g$$

where $x^\gamma = \text{LCM}(\text{LM}(f), \text{LM}(g))$ is the least common multiple of $\text{LM}(f)$ and $\text{LM}(g)$. $\text{LM}(f)$ denotes the leading monomial and $\text{LT}(f)$ the leading term of $f$ w.r.t. a monomial ordering.

Acknowledgement

This work was supported by the EC project MEST-CT-2005-021024 WARTHE and the project MSM6840770012 - Transdisciplinary Biomedical Engineering Research II.

References

Co-occurrence Bag of Words for Object Recognition

Naeem A. Bhatti\textsuperscript{1} and Allan Hanbury\textsuperscript{2}

\textsuperscript{1}Vienna University of Technology
Pattern Recognition and Image Processing Group
Favoritenstr. 9/1832, 1040 Vienna
bhatti@prip.tuwien.ac.at

\textsuperscript{2}Information Retrieval Facility
Operngasse 20b, 1040 Vienna
a.hanbury@ir-facility.org

Abstract In the original ‘bag of words’ model image features are seen as independent and orderless. In the real world, relative co-occurrence of image features defines the spatial order in which they appear. This means the spatial relationship between image features has potential information to be exploited in a classification task. Our model attempts to discover the usefulness of the co-occurrence information of image features with its extent ranging from local to global level targeting the image classification task. To establish the pair-wise relations between the features, we use the neighborhood of each image feature. To define the extent of neighborhood of an image feature at a local level and to achieve scale invariance, the model uses a circular area based on the elliptical scale at which each image feature is detected. To range from local to global level, the model explores the extent of the neighborhood spatially at integral multiples of the circular area radius. In turn, each image is described by a histogram of pair-wise visual words and a concatenated histogram of independent visual words with pair-wise visual words. We develop a stop word removal technique to eliminate noisy redundant visual words from the collection. Experimental results show that the pair-wise co-occurrences of visual words alone and their augmentation with independent occurrences leads to a positive improvement in the classification accuracy.

1 Introduction

The relative co-occurrence of local image features plays a significant role in the recognition of objects. If local features of an object are displaced, it becomes difficult to recognize the same object. The exploitation of spatial co-occurrence information can facilitate object recognition, categorization as well as image classification tasks. Modeling of this spatial co-occurrence information at inter-pixel level, object level, region level and local image feature level has been under consideration in computer vision literature [1, 5, 6, 9]. This work considers spatial co-occurrence of local image features and investigates its usefulness in image classification task.

Representation of images by local features is currently state of the art. It is possible to explore the effectiveness of co-occurrence information at the local features level. In this work, the goal is to investigate the usefulness of co-occurrence information of local image features targeting an image classification task. To exploit local spatial co-occurrence information of local image features, we expand the bag of words model. In this bag of words model, the visual words are seen as independent and orderless. The dependencies among visual words are ignored, as the representation of images is made by a vector of frequency of occurrence of visual words independently. Hence this representation alone cannot be said to be complete as the correlation or relationship among visual words is not being captured. What is the role of dependence among visual words in pattern recognition? It is the question under consideration in this work.

Our inspiration to explore the role of co-occurrence information of image features in terms of their relative co-occurrence to each other comes from text retrieval techniques [2, 3]. The Bag of Words model is well established in text retrieval for its state of the art performance, in which only single term patterns are explored in a document. In an attempt to go beyond the Bag of Words paradigm in text retrieval, a shift in exploitation of token occurrence of single term to term-pairs within a sliding window has been explored [2]. In the past, different techniques of text retrieval have been applied successfully in image retrieval as well as object recognition. The well established techniques and theories in the text retrieval field are applicable to object recognition and retrieval as they share the same objective of information retrieval but have to deal with different data formats [4].

We hypothesize that the visual words have spatial dependencies with each other, which can be seen as their relative co-occurrence. To capture the relative co-occurrence of features in a local neighborhood, as an analogy to a fixed window size in text retrieval to capture term-pairs co-occurrence, we define a circular area based upon elliptical scale size of each feature. In this spatial area, we establish pair-wise relations of the features which lie in this
neighborhood. To range from local neighborhood to global neighborhood, we extend the extent of the neighborhood area by taking integral multiples of the circular area radius. In this way, we explore the dependence of image features on each other in a neighborhood with an extent of local level extending towards global level in a stepwise fashion. To model feature co-occurrence, we develop a relational features codebook out of the independent features codebook. This relational codebook represents all possible pairwise relations between the visual words. In turn we represent each image in terms of the new relational codebook. We describe the full details of developed approach in Section 3. In Section 2, we summarize current approaches presented in the literature to exploit context of image features. We present the experimental results in Section 4 and conclude in Section 5.

2 Overview of Current Approaches

In the computer vision literature, several approaches have been developed to capture and investigate the effectiveness of spatial co-occurrence information of local image features in recognition tasks.

Savarese et al. [10] developed an approach to capture the spatial co-occurrences of features using correlograms of visual words. The approach captures the relationship between visual words as a function of distance in the image and constructs a correlogram matrix. Each element of the correlogram matrix is called correlaton. In turn, image representation is made in terms of the correlaton histogram. An improvement in results is shown by an augmentation of the correlaton representation with the visual words representation (joint appearance and shape model). In [7] Lazebnik et al. investigate part-based and feature-based relations in image data. The aim is to investigate whether incorporating feature relations into the image representation improves image classification results. However the work states that feature relations alone perform worse and augmentation with a part-based model does not improve the performance.

To explore the spatial relationship of visual words, Sivic et al. [11] establishes pair-wise relations between visual words and called the pairs as doublets. The approach discards the visual words which have significant overlap of their elliptical scales. So in the formation of doublets the words having no overlapping ellipses are considered. The usefulness of the approach is shown in obtaining clear segmentation of objects in images. The work of Galleguillos et al. [9] captures spatial co-occurrence information at object level by constructing co-occurrence matrices for four types of pairwise spatial relationships among objects (above, below, inside and around). Their work uses the Conditional Random Field model to incorporate spatial and co-occurrence information in maximizing object label agreement. A considerable improvement in accuracy is shown on the MSRC and PASCAL 2007 datasets. The Zhang et al. [12] approach is based on the bag of words model. Their approach exploits local features spatial relationship. In the first step, all the keypoints are ranked on the basis of their occurrence frequency and then the frequent keypoints are selected from training images. The less frequent keypoints are discarded on the assumption that they represent background clutter and image noise. The spatial relations of these selected keypoints are established as doublets and triplets with keypoints that lie within a predefined radius. To reduce the number of overall features, the top 100 single keypoints are selected from each class and top 100 doublets. Results are shown on CALTECH 101 and 15 categories dataset, which are not better than the state of the art. To capture local shape structure, Ferrari et al. [8] presents local contour features in small groups of connected segments called k adjacent segments. They showed that adding the information of spatial organisation of spatially localized k adjacent segments improves the classification accuracy as compared to the standard orderless bag of words.

In the above mentioned approaches, to incorporate co-occurrence information of local image features with simple occurrence information, the Bag of Words model and Conditional Random Fields model (CRF) are used. To investigate co-occurrence information at the semantic level, object level and part level CRF are mostly used. Whereas to investigate co-occurrence interactions at local features level mostly the BoW model is employed. The contribution of this paper is to take a co-occurrence Bag of Words model from the text retrieval field, and modify it to apply to visual words in images. As in the text retrieval, the model is parametrized by a window size in which visual word pairs are counted. This proposed model is simpler than other proposed models to exploit co-occurrence information of image features, as it involves an expansion of the feature vectors by adding information on visual word pairs. As these visual word pair feature vectors have length proportional to the square of the number of visual words, a further contribution is the proposal of a technique to remove non-essential visual word pairs.

3 Co-occurrence Bag of Words

Spatial co-occurrence information is to explore the co-occurrence of local image features in a specified locality. In a local feature based image representation, to explore co-occurrence information is to quantify the relationship of one local feature with other local features. In this scenario, we have two tasks at hand: 1) to define the relationship level between image features 2) to specify the spatial extent of the locality to explore the relationship. Considering the image plane and local image features representation, the relationship between image features can be seen at some spatial extent. If we consider a local image feature as reference, there can be different possible relations of the reference image feature with other image features lying in a neighborhood. This relationship paradigm can be studied in terms of distance, correlation, dependence, and co-occurrence with each other. In this work, we explore image feature relations in terms of their co-occurrence at a spatial extent. To model co-occurrence of local features, we select and extend the Bag of Words model. The reason
behind this selection is the simplicity of the model so that it is easy to incorporate the co-occurrence information in it.

### 3.1 Relational Features Codebook

The vector quantization step converts an image into the codebook feature space representation. In this feature space, each image feature has the representation by a feature (Visual Word) from the set of previously determined codebook features. To represent an image as a histogram of visual words in the classic Bag of Words representation, we count the number of occurrences of each visual word. To represent each image as a histogram of pairs of visual words, we have to count the number of occurrences of each pair of visual words. To establish pairs of visual words, we look for all possible and distinct pairs of visual words. In order to implement the relational features codebook, we develop a matrix $M$ of $n$ rows and $n$ columns. Each element $M_{i,j}$ of this matrix represents the pair of one independent feature with another feature. In this way, we have all possible pairs represented in this matrix. If we consider only diagonal elements and upper diagonal elements of matrix $M$ we can get distinct pairs out of all possible pairs of independent features as shown in Figure 1. To maintain a linear index of distinct pairs, we store linear indices to the locations of diagonal and upper diagonal elements of matrix $M$ starting from the first element to the last element. In this way, we establish distinct pairwise relations between independent codebook features and index them linearly. A standard codebook of $n$ features produces a codebook of $n(n+1)/2$ relational features instead of $n(n-1)/2$ pairs in [3], as we also consider a visual word co-occurrence with itself as shown in Figure 1. We therefore have:

$$IC = n$$

and

$$RC = n(n + 1)/2$$

where $IC$ is Independent Features Count and $RC$ is Relational Features Count.

![Figure 1: The distinct pairs out of all possible pairs are shown for 3 visual words.](image)

### 3.2 Modeling the Spatial Co-occurrence of Visual Words

In the developed approach, the main step is to model the co-occurrence information of Visual Words. In our employed method to explore the effectiveness of the captured spatial co-occurrence information, the main steps are analogous to the Bag of Words model. Now we discuss the development at each step in more detail:

#### 3.2.1 Feature Extraction and Description

The first step is the extraction of features in the image and to describe these features. To cope with illumination variations and geometrical transformations, a number of scale and affine invariant feature detectors and descriptors have been developed [13]-[17]. We use the Hessian affine detector presented in [17] because of its scale and affine invariance. Besides being able to cope with illumination and geometrical variations, scale and affine invariant point detectors have their own limitations. In an attempt to capture the invariance property, they become less discriminative. To normalize the detected affine regions for affine transformations, the detected affine regions are mapped to circular regions as in [20]. To describe the normalized circular regions, we use the Scale Invariant Feature Transform (SIFT) descriptors [14]. They split the regions in $4 \times 4$ grids and compute Gaussian derivatives in 8 orientation planes giving a 128-dimensional vector for each interest point region. Our preference for SIFT descriptors is due to their stability towards noise and their rich and potentially more discriminative representation. Mikolajczyk et al. [18] performed an evaluation of several descriptors and showed that SIFT descriptors have the best performance.

#### 3.2.2 Codebook Construction and Quantization

After going through the interest point detection and description of each image in the data corpus, the next step is to construct a Codebook of the image features. The motive behind constructing a codebook is to learn an optimal number of the features from the training images and to then construct a feature vector of each image capturing statistics of these codebook features in terms of their frequency of occurrence. This feature vector representation is used in the classification of a query image using knowledge of feature vectors of the training images. The codebook formation is done through clustering. A number of clustering techniques including square-error partitioning algorithms to hierarchical techniques are available in the literature. We opt to use the k-means clustering algorithm which employs a simple square-error partitioning method [19]. In the k-means algorithm, one difficulty is to determine the parameter $k$. We determine this parameter experimentally and select the value of $k$ which confirms minimum empirical risk in the classification task. After several k-means algorithm iterations, we get the cluster centers which represent the codebook features (visual words).

Once the visual words are obtained, the next step is the quantization or assignment of each local feature of each image to one of the visual words. This step is achieved by assigning a feature to a visual word with which it has minimum distance. The frequency of each visual word constitutes the feature vector of each image, called the histogram. In the Bag of Words model, it is assumed that all these visual features appear independently from each other. Whereas our assumption is that visual features ap-
Co-occurrence Bag of Words for Object Recognition

Figure 2: The local neighborhood of the interest points shown in magenta. The interest point for which the neighborhood is defined is shown in red with its elliptical scale of detection, whereas the interest points lying in the local neighborhood are shown in yellow. A few interest points are shown for clarity.

3.2.3 The Extent Range of the Neighborhood

Natural images appear in multiple spatial resolutions. As the spatial resolution changes, the spatial extent of the appearance of the objects changes accordingly. To cope with the unseen changes in the spatial resolution during modeling of the co-occurrence information, we have to consider the scale invariance of the neighborhood spatially. To achieve scale invariance, we define neighborhood based upon the elliptical scale of detection of the image features. In other words, the extent of neighborhood is a circular area of radius equal to the minor axis plus major axis of the elliptical region of interest point

\[ R = l_1 + l_2 \]  

(3)

This circular area is analogous to the sliding window established to take into account term-pair co-occurrence in text retrieval [2]. However, in our case the extent of circular area is variable for each image feature as its radius directly depends upon the minor axis and major axis of the elliptical scale as shown in the Figure 2.

To find the image features which are in a spatial relationship with each other at a certain extent of the neighborhood, starting at local level, we move towards global level. We traverse all the image features and establish the spatial relation of one image feature with other features in the defined neighborhood. Denote the image feature under consideration as \( IF \), and the feature in spatial relationship as \( RF \). Make a circular area of radius \( r \) which marks the neighborhood of the image feature \( IF \) under consideration at a local level. In this local neighborhood, we look for the other related image features \( RF \) which appear in this circular area. An image feature is said to be \( RF \) of an image feature \( IF \), if it lies in the neighborhood of \( IF \), such that:

\[ |dx| + |dy| < r \]  

(4)

where \( dx \) is the absolute distance of the \( RF \) feature with respect to the \( IF \) feature in the \( x \) direction and \( dy \) is the absolute distance in the \( y \) direction. A row vector of spatial pairs denoted as \( SP_k \) where \( k = 1...IC \) is formed which contains the indexes of all the \( RF \) features for each \( IF \) feature. The length of \( SP_k \) depends upon the number of the \( RF \) features for a particular image feature.

At the end of this step, we have \( IC \) row vectors of variable lengths. In all of these row vectors, there are possible repeatable pairs in a sense that: one \( IF \) feature has a particular image feature as its \( RF \) feature, which in turn when
considered as IF feature can probably have the first IF feature as its RF feature. We develop a technique to avoid this repeatability of pairs to have distinct and unrepeatable pairs in $SP_k$. We experiment with the feature vectors of images formed from $SP_k$ with repeatable pairs and $SP_k$ without repeatable pairs. It is found that the effect of removing repeatable pairs on classification accuracy is minor. We explore the effect of the extent range of the neighborhood at a local level and extending towards a global level of the image plane. We define the local extent of the neighborhood by the circular area $R$ defined in Equation 3. To increase the extent of the neighborhood spatially from local level towards global level, we use the incremental extent range of the neighborhood by integral multiples of circular radius $R$.

$$R_n = n \cdot R$$

where $n$ is an integer and $R_n$ is the radius of the $n^{th}$ extent of the neighborhood. The upper bound of extent range prescribed by integer value $n$ varies for different databases to experiment on. However, it is determined experimentally and bounded by the integer value at which the effect of co-occurrence information in classification accuracy can be well analyzed. The extent levels for a range of $4R$ are shown in Figure 3 in an image of the category “Bikes and Persons” of the Graz01 database.

### 3.2.4 Image Description in terms of the Relational Codebook

After we have developed the spatial relations of the image features in a neighborhood with an extent defined in Equation 5, the next step is to describe each image in terms of the Relational Feature Codebook, which has all the possible pairs of visual words already defined. As in the simple Bag of Words model, the image features are projected to visual words obtained by clustering. After establishing the pairwise relations between the image features locally, all pairs are to be projected to visual word pairs in the Relational Feature Codebook. In this way, it forms a histogram representation of size $RC$ for each image. For each feature vector, $n$ in Equation 5 is constant. Each histogram bin represents the number of occurrences of each pair of visual words in the image. In other words, each histogram bin represents the number of co-occurrences of two visual words in the image. This histogram, as it captures co-occurrence of visual words, is named Co-occurrence Histogram abbreviated as $H_c$. The histogram which is obtained from the standard bag of words approach is named the Non-co-occurrence Histogram abbreviated as $H_{nc}$. Another way to exploit the usefulness of co-occurrence information of visual words is to augment the non-co-occurrence information with co-occurrence one. This augmentation is obtained by concatenating $H_{nc}$ with $H_c$ in serial fashion and abbreviated as $H_{enc}$, also seen as early fusion of the two feature representations.

### 3.2.5 Word Pair Removal Technique

To explore the usefulness of the co-occurrence information and its augmentation with non-co-occurrence information, we also develop a word removal technique to remove noisy, less frequent as well as more frequent word pairs. We calculate the number of non zero occurrences of each visual word pair in all the images in the matrix of feature vectors. From this vector of non zero occurrence, the maxima are obtained. We calculate two threshold values from this maxima, the upper threshold $Tu$ and lower threshold $Tl$ by

$$Tu = \max (NV) \times c1$$

$$Tl = \max (NV) \times c2$$

where $NV$ is the vector containing the number of nonzero occurrences of each visual word pair and $c1$, $c2$ are constants whose values are in the range from 0 to 1. The values of $c1 = 0.9$ and $c2 = 0.1$ are found best in our experiments. The values are once found and set for all the experiments. We apply the technique on the word-image matrix obtained from by the feature vectors $H_c$ and $H_{enc}$. The visual words pairs which have non zero occurrences greater than $Tu$ and less than $Tl$ are removed from the word-image matrix. In turn, the visual word dimension of the word-image matrix is reduced. The feature vectors obtained after the word removal are named as $H_{cwr}$ and $H_{encwr}$. The usefulness of the developed word removal technique and feature vector representations is shown and discussed in the Experiments section.

### 4 Experiments

To explore the effectiveness of the developed approach in image classification, we perform experiments on the challenging Graz01 image database and present the classification average accuracy. The classification is performed by the Naive Bayes classifier and SVM classifier. To test our developed approach, we adopt the strategy of Object vs. No-Object as adopted in [21, 22] for the Graz01 database. To formulate our classification task as binary (two-class) classification, for each category in the database, we split the images randomly into two parts: 50% training and 50% testing. To train the classifier, we use all the images from the training part where the images which belong to the Object category are taken as positive examples and the images which belong to the No-Object category are taken as negative examples. We test our developed approach with all the images in the testing part where the images from Object category are taken as true positives while the images from No-Object category as true negatives.

Our approach incorporating the co-occurrence information and its augmentation with non-co-occurrence information shows comparable performance with the non-co-occurrence approach while showing the importance of spatial relation information as improvement in the classification accuracy in the specific categories.

#### 4.1 Performance Evaluation

We formulate our experiments for binary classification using the One-Against-All strategy in the Object vs No-Object scenario. The number of classes which belong to the No-Object category is often more than one, which makes the overall number of images in the No-Object category greater than the number of images in Object category. The greater number of images in one class makes the greater influence in the classification results. To have unbiased results, we
use the measure of average accuracy of classification obtained as:

\[ AC = \frac{TP}{TP + TN} \times 100 \]  

where \( AC \) is the average accuracy of the classification, \( TP \) are the true positives, \( TN \) are the true negatives, \( TP \) is the total number of positive images and \( TN \) is the total number of the negative images. We perform the experiments for simple BoW with the \( H_{nc} \) histogram of each image and take the results as a reference to compare the results of our developed approach. The experiments for the developed approach are performed with the \( H_{c} \) and \( H_{cnc} \) histograms of each image. Also we explore the effect of the developed word removal technique by performing experiments with the \( H_{ncw} \) and \( H_{ncncw} \) feature vectors of each image. As to explore the effectiveness of the co-occurrence information, we extend the extent of the neighborhood spatially from local level towards global level given by Equation 5 and 6. We present the results for the different extents of the neighborhood for each class of the input image databases so that the effect of co-occurrence information travelling from the local level to the global level can be analyzed. The results for the simple BoW feature vector representation \( H_{nc} \) are presented under label \( NC \), whereas the results for the developed feature vectors representations \( H_{c} \), \( H_{ncw} \) and \( H_{ncncw} \) are presented under the labels \( C \), \( CW \), \( C \), \( C \) and \( C \) respectively in the tables as well as graphs.

In our experimental setup, we explored the effect of codebook size parameter \( k \) on the classification accuracy for a range of values \( k = 500, 1000, 1500, 2000 \) experimentally. It is found that the value of \( k = 1000 \) performs well for the database as explained in [20].

### 4.2 Results and Discussion

In this section, the classification results obtained from the Naïve Bayes classification and SVM classification for Graz01 are presented. We use the Joachim’s binary classification SVM software\(^1\) and use linear kernel for classification. The linear kernel is used in text retrieval techniques where the number of features is significantly large. As the number of visual word pairs given by Equation 2 and their augmentation with independent visual words is large, we prefer the linear kernel for SVM experimentation.

In the Graz01 database, there are three categories named “Bikes” (B), “Persons” (P) and the “Bikes and Persons” (BaP) comprising two major objects bike and person and a fourth category containing none of these objects which is “No Bike No Person” (N). The investigation for the effect of the co-occurrence information in the classification accuracy is made at an extent range of \( 4R \) i.e. \( 1R, 2R, 3R \) and \( 4R \) of the neighborhood. The extent range of \( 4R \) is found experimentally, where the effectiveness of co-occurrence information can be well analyzed. To achieve the Object vs. No-Object strategy, we formulate binary classification experiments as:

- In the first set of experiments, we test each of the category “B”, “P” and “BaP” against the category “N”. For the category “B”, the co-occurrence information captured at local neighborhood level \( 1R \) improves the classification accuracy to 5.80% as shown in Figure 4(a), however as we move further towards global level, the results start decreasing. This confirms that at the local level the co-occurrence information captured belongs to the object “Bikes” and improves the classification significantly, whereas at increasing levels of the extent of neighborhood, co-occurrence information overlaps with background and the effect of co-occurrence information decreases as it is clear from Figure 4(a). For the category “P”, results are shown in Figure 5(a). The co-occurrence information improves the classification accuracy in an increasing fashion from local extent level to global level of the neighborhood. At extent level of \( 1R \), accuracy remains low as compared to non-co-occurrence case, at \( 2R \) it becomes equal to the non-co-occurrence case, however at extent level of \( 3R \), the accuracy improves to 3% and shifts to 2% increase at extent level of \( 4R \) as shown in Figure 5(a). For the object “Persons”, the co-occurrence information at local level does not capture the structure of the object well however increasing the extent of the neighborhood brings structure into account which shows an improvement in classification accuracy. The significant increase in results at extent level of \( 3R \) confirms the advantage of exploration of the co-occurrence information at an extent range of various

---

\(^1\)http://svmlight.joachims.org/
levels. The same behavior, we see in the case of category “BaP”, where we have 1.2% increase at an extent level of 3R and 1.33% increase at an extent level of 4R as shown in Figure 6(a). This shows that the extent level of the neighborhood should be more explored to capture the structure of the object well in the category “BaP” to have a significant improvement in the classification. The developed word removal technique shows its benefit for the category “BaP” where it improves the results by 2% for CNCWR as compared to NC and 4% improvement as compared to CNC. The Naive Bayes classifier outperforms the SVM classifier in the C and CNC cases, whereas the SVM classifier performs better than the Naive Bayes classifier in case of NC as shown in Figures 4(b), 5(b) and 6(b).

- We perform the second experiment for the category “B” against categories “P” and “N”. The results in Figure 7(a) show that the co-occurrence information improves results by 3% at extent level of 1R but no further improvement at further extent levels of the neighborhood, however the results remain always better than NC. The fall in results of C at 1R as compared to the results in “B” vs “N” is due to the inclusion of more complex backgrounds by category “P” in the No-Object category. The SVM classification is shown in Figure 7(b) has no significant improvement.

- The third experiment we perform is for Category P against categories B and N. The results are shown in Figure 8(a) for the Naive Bayes classifier and in Figure 8(b) for the SVM classifier. The Naive Bayes results show a 5% of increase in classification accuracy at extent level of 3R whereas for other levels of extent there is a 4% increase for C. In CNC case, the results are almost the same but we have 5% increase at an extent level of 4R. The SVM results also show improvement for C and CNC cases. The word removal technique has no effect in overall improvement of the results.

5 Conclusion

In this paper, we have presented a novel approach to capture the co-occurrence information of the local image features. The local neighborhood to capture the spatial relations between image features is defined based on the elliptical scale of each feature. To further explore the effect of co-occurrence information in classification accuracy, we explore the neighborhood spatially at different levels of extent ranging from local level to global level. The capture of co-occurrence information is incorporated in the Bag of Words model. We perform experiments on Graz01 database using the Naive Bayes and SVM classifiers.

The results show that the co-occurrence information alone and its augmentation with non-co-occurrence information improves the classification accuracy significantly which shows the potential of the co-occurrence information. Our future direction is to explore the co-occurrence information in other image databases like PASCAL and Caltech. We also plan to increase the model stability and study further methods to explore the co-occurrence information.
Acknowledgement

This research work was supported by Higher Education Commission (HEC) Pakistan under the “Pakistan Overseas Scholarship Program for PhD in Selected Fields”.

References

Efficient Dimensionality Reduction Using Random Projection

Vildana Sulić, Janez Perš, Matej Kristan, and Stanislav Kovačič
Faculty of Electrical Engineering, University of Ljubljana, Slovenia
vildana.sulic@fe.uni-lj.si

Abstract Dimensionality reduction techniques are especially important in the context of embedded vision systems. A promising dimensionality reduction method for a use in such systems is the random projection. In this paper we explore the performance of the random projection method, which can be easily used in embedded cameras. Random projection is compared to Principal Component Analysis in the terms of recognition efficiency on the COIL-20 image data set. Results show surprisingly good performance of the random projection in comparison to the principal component analysis even without explicit orthogonalization or normalization of transformation subspace. These results support the use of random projection in our hierarchical feature-distribution scheme in visual-sensor networks, where random projection elegantly solves the problem of shared subspace distribution.

1 Introduction

In visual systems we usually deal with large amounts of digital image data. Data has to be archived or exchanged between numerous users and systems [3], consuming expensive resources, such as storage space or transmission bandwidth.

In digital imaging the basic unit of image is pixel. Therefore, an image can be represented as a feature vector, where each pixel corresponds to one feature value. Even standard resolution images (e.g., VGA) contain large number of pixels and therefore the resulting feature vector representation has usually high dimensionality. In order to handle real-world data adequately, the dimensionality needs to be reduced [27]. Using dimensionality reduction techniques is especially important when data is distributed across networks with limited bandwidth.

A dimensionality reduction technique that is capable to reduce the data into a lower-dimensional model, while preserving the reconstructive or discriminative properties of the original data can be marked as ideal. However, in practice information is lost as the dimensionality is reduced. Therefore, a method which efficiently reduces dimensionality, while preserving as much as possible information from the original data is needed. One solution is to reduce the dimensionality of data by projecting it onto a lower-dimensional subspace [18].

Dimensionality reduction techniques using linear transformations have been very popular in determining the intrinsic dimensionality of the manifold as well as extracting its principal directions (i.e., basis vectors). The most famous method in this category is the Principal Component Analysis (PCA) [11]. PCA (also known as the Karhunen-Loève transform) is a vector-space transform that reduces multidimensional data sets to lower dimensions while minimizing the loss of information. A low-dimensional representation of the data is constructed in such a way that it describes as much of the variance in the data as possible. This is achieved by finding a linear basis of reduced dimensionality for the data (a set of eigenvectors) in which the variance in the data is maximal [27].

Besides PCA, many other dimensionality reduction techniques exist. Recently, Random Projection (RP) [28, 16, 5] has emerged as a powerful method for reducing dimensionality. The most important property of the RP method is that it is a general data reduction method. Unlike PCA, it does not depend on a particular training data set. Unlike Discrete Cosine Transform (DCT) or Discrete Fourier Transform (DFT) its basis vectors do not exhibit particular frequency or phase properties.

In RP, the original high-dimensional data is projected onto a low-dimensional subspace using a random matrix, whose columns have unit length. If compared to other methods, for instance PCA, which compute a low-dimensional subspace by optimizing certain criteria (e.g., PCA finds a subspace that maximizes the variance in the data), RP does not use such criteria, therefore, it is data independent. Furthermore, it represents a computationally simple and efficient method that preserves the structure of the data without significant distortion [11]. There exist theoretical results supporting that RP preserves for example volumes and affine distances [19] or the structure of data (e.g., clustering) [6].

1.1 Motivation

Dimensionality reduction techniques are especially important in the context of embedded vision systems, such as smart cameras. The reasons for that are the specific characteristics and limitations of smart cameras (i.e., low processing power, limited storage space and limited network bandwidth) [26]. Processing of dimensionality reduced data usually requires far less resources than processing of unmodified data. In our previous work [24, 25] we proposed a framework
of hierarchical feature-distribution (HFD) for object recognition in a network of visual sensors, which utilizes network in a more balanced way than trivial network flooding. HFD for visual-sensor networks (VSNs) is based on hierarchical distribution of the information, where each individual node retains only a small amount of information about the objects seen by the network. Nevertheless, this amount is sufficient to efficiently route queries through the network without any degradation in the recognition performance. The amount of data transmitted through the network can be significantly reduced using our hierarchical distribution.

One of the methods for image feature extraction used in our work was PCA. We used the variant of the PCA, which assumes that the eigenvectors (the subspace) were obtained in advance. We assumed that each sensor has an access to the global subspace. However, distribution of subspace is problem by itself. In practical implementation this subspace would have to be transmitted to each and every node in the network, which would cause significant network traffic. Alternatively, cameras could use fixed subspace, which is built into the camera at the time of manufacture or installation.

Therefore, the problem that we are aiming to solve is as follows. How to distribute common subspace to all cameras in a network without transmitting large amount of data?

Using RP, there is a possibility that all cameras in the network recreate exactly the same subspace with minimum of transmitted information. This is possible, if pseudorandom generator is used to generate the random matrix. The only thing that has to be known to each camera in the network is the pseudorandom state. In practice, it is sufficient that each camera knows only one parameter, called seed of the pseudorandom generator. Since same pseudorandom seed results in exactly the same pseudorandom sequence, recreation of the same random matrix in each camera is possible. This means that each camera in the network projects the input data (images) through the same random matrix into same subspace.

Therefore, since RP method enables reduction of data dimensionality, is computationally simple, preserves the structure of the data and is increasingly used in distributed visual-sensor networks, we decided to examine the recognition efficiency of RP and compare it to the well-known PCA. Other researchers have explored RP in the context of many applications, but to best of our knowledge, a detailed comparison to PCA has not been done in the terms of recognition efficiency. Moreover, RP can be implemented in many ways, and many of them are not appropriate for the resource-constrained embedded cameras. Therefore, our aim is to determine the performance of RP implementation, which can be directly used in embedded cameras. Furthermore, we decided to explore several variant of RP and their effect on recognition performance. Therefore, our main contribution is extensive comparison between RP and PCA (our baseline) in terms of recognition accuracy, in a way that is directly relevant to use of RP in embedded camera systems and VSNs.

The remainder of this paper is organized as follows.

In the Section 2 we provide theoretical background of the RP, including selection of the random matrix and its orthogonalization, and we present different applications of the RP. Experiments and results of tests are reported and discussed in the Section 3. Section 4 provides discussion and conclusion.

2 Random projection

Random projection is a powerful dimension reduction technique that uses random projection matrices to project data from high-dimensional subspace to a low-dimensional subspace [11]. The RP technique has been used for designing algorithms for problems from a variety of areas, such as combinatorial optimization, information retrieval and machine learning [28]. Some theoretical background of RP and a brief review of its applications is presented below.

2.1 Background

The main idea of RP is that using a random matrix whose columns have unit lengths, the original high-dimensional data is projected onto a lower-dimensional subspace [3]. RP has been found computationally efficient and a sufficiently accurate method for dimensionality reduction of highly dimensional data sets (e.g., [14, 16, 4, 6, 1] just to name a few).

The concept of RP is as follows: Given a data matrix $X$, the dimensionality of data can be reduced by projecting it onto a lower-dimensional subspace formed by a set of random vectors $[16]$.

$$A_{[m \times N]} = R_{[m \times d]} \cdot X_{[d \times N]},$$

where $N$ is the total number of points, $d$ is the original dimension, and $m$ is the desired lower dimension. The central idea of RP is based on the Johnson-Lindenstrauss lemma (JL lemma) [15]:

**Johnson-Lindenstrauss lemma**

*For any $0 < \varepsilon < 1$ and any integer $n$, let $k$ be a positive integer such that $k > 4\left(\frac{\varepsilon^2}{2} - \frac{\varepsilon^3}{3}\right)^{-1} \ln n$. Then for any set $V$ of $n$ points in $\mathbb{R}^d$, there is a map $f : \mathbb{R}^d \mapsto \mathbb{R}^k$ such that for all $u, v \in V$ [7],

$$(1 - \varepsilon)\|u - v\|^2 \leq \|f(u) - f(v)\|^2 \leq (1 + \varepsilon)\|u - v\|^2,$$

where $f(u)$ and $f(v)$ are the projections of $u$ and $v$. Using the above lemma, [7] shows that if we perform an orthogonal projection of $n$ points in a vector space ($\mathbb{R}^d$) onto a selected lower-dimensional subspace, then distances between points are preserved (i.e., not distorted more than a factor of $1 \pm \varepsilon$), for any $0 < \varepsilon < 1$. For complete proofs on the lemma refer to [10, 7]. The JL lemma can be proven for sparse transformation matrices $\mathbb{R}$ as well, for details see [1, 17, 2].

**Selecting the random matrix.** The choice of random matrix $R$ is one of the crucial points of interest.
An efficient method for dimensionality reduction using the JL lemma employs a random matrix $R$, whose elements are drawn independently and identically distributed (i.i.d.) from a zero mean, bounded variance distribution [26]. There are many choices for the random matrix. A random matrix with elements generated by a normal distribution $r_{i,j} \sim N(0,1)$ is one of the simplest in terms of analysis [28]. The problem of this type of RP is its computational complexity due to the dense nature of the projection matrix. Achlioptas [1] suggested the use of two simpler distributions that generate sparse projection matrices with elements drawn i.i.d. as:

$$r_{i,j} = \begin{cases} +1 & \text{with probability } \frac{1}{2} \\ -1 & \text{with probability } \frac{1}{2} \end{cases} \quad (4)$$

or

$$r_{i,j} = \sqrt{3} : \begin{cases} +1 & \text{with probability } \frac{1}{2} \\ 0 & \text{with probability } \frac{1}{2} \\ -1 & \text{with probability } \frac{1}{2} \end{cases} \quad (5)$$

Distributions shown in Eq.(4) and (5) reduce computational time for the calculation of $R \cdot X$. For the second distribution, the speedup is threefold because only one-third of the operations are needed. Authors in [17] further explored the idea of sparse RP and suggested a method for achieving a $\sqrt{n}$ - fold increase in speedup with a small penalty in the preservation of the pairwise distances. In this work, we use the sparse projection matrix, presented in Eq.(5). Usage of sparse RP in a distributed environment is presented in [29].

Orthogonalization If the random vectors were orthogonal, then the similarities between the original vectors would be preserved under RP [16]. Ideally we would want the random matrix to be orthogonal but, unfortunately, orthogonalization is very costly (e.g., Gram–Schmidt method has a complexity in the order of $nm^2$ if $m$ principal eigenvectors of dimension $n$ have to be determined [21]). The cost of the orthogonalization defeats the purpose of using RP on resource constrained embedded cameras. However, Hecht-Nielsen et al. [16, 13] have noted that in a high-dimensional space, there exist a much larger number of nearly orthogonal than truly orthogonal vectors. Therefore, in a high-dimensional space even random vectors might be sufficiently close enough to orthogonal to offer a reasonable approximation of the original vectors [18].

To summarize, RP combines very interesting characteristics, making it ideal for computer vision. First, it tackles the “curse of dimensionality” by projecting the data to a much lower dimensional space. In addition, problems that deal with large amounts of data can be tackled more efficiently by operating on fewer data [26].

In contrast to similar dimensionality reduction techniques, such as PCA, RP is data independent, while still preserving the structure of the input data. RP can be applied on various types of data such as text, image, audio, etc.

2.2 Applications

There are several successful applications of RPs to computer vision problems. Tsagkatakis et al. [26] use RP for object tracking under variable pose and multi-camera views. Han et al. [12] used RP and robust linear discriminant analysis for face recognition. Wright et al. [30] extended their use by using RP and a novel $l_1$-norm minimization for face recognition. An insightful observation shared by many researchers that utilize RP for computer vision applications, as noted in [30], is that the choice of features is not as important as the number of features. This observation could prove to be of great significance for resource constrained environments such as embedded vision systems, where elaborate feature extraction may not be feasible due to the imposed limitations [26]. Kaski [16] presented experimental results using RP in the context of a system for organizing textual documents using Self-Organizing Map (i.e., WEBSOM). In this case, the results were as good as those obtained using PCA, and almost as good as those obtained using the original vectors.

Lin and Gunopulos [18] compared RP with EM to learn geometric object appearance for object tracking under variable pose and multi-camera views. Han et al. [12] used RP and robust linear discriminant analysis for face recognition. Wright et al. [30] extended their use by using RP and a novel $l_1$-norm minimization for face recognition. An insightful observation shared by many researchers that utilize RP for computer vision applications, as noted in [30], is that the choice of features is not as important as the number of features. This observation could prove to be of great significance for resource constrained environments such as embedded vision systems, where elaborate feature extraction may not be feasible due to the imposed limitations [26]. Kaski [16] presented experimental results using RP in the context of a system for organizing textual documents using Self-Organizing Map (i.e., WEBSOM). In this case, the results were as good as those obtained using PCA, and almost as good as those obtained using the original vectors. Lin and Gunopulos [18] compared RP and Latent Semantic Indexing (LSI) in the area of information retrieval. Bingham and Mannila [5] compared several dimensionality reduction techniques, such as PCA (based on data covariance matrix), RP and Discrete Cosine Transform (DCT) on image and text data. Their results indicate again that RP preserves distances and has performance comparable to that of PCA while being faster. Work, which is related to [5] and is focused on using RP for lossy image compression is presented in [3]. Dasgupta [6] described experiments on learning mixtures of Gaussians in high dimensions using RP and PCA. Li et al. [17] used RP with EM to learn geometric object appearance for object recognition. Motivated by the results of [6] Fern and Brodley [8] investigated the application of RP for clustering high-dimensional data. More recently, Fradkin and Madigan [9] evaluated RP in the context of supervised learning. In particular, RP was compared with PCA on a number of different problems using different machine learning algorithms. They concluded that although RP was slightly worse than PCA, its computational advantages might make it attractive in certain applications.

RP has demonstrated good performance in a number of applications, yielding results comparable to conventional dimensionality reduction techniques, such as PCA, while having much lower computational requirements. Feasibility of RP for face recognition is also investigated in [11], where authors compared RPs performance to the performance of PCA. However, even if theirs results suggest that RP is comparable to PCA, they used Gram-Schmidt algorithm for orthogonalization of random matrix, which is computationally wasteful. In this paper we do not orthogonalize random matrix since according to Hecht-Nielsen et al. there exist a much larger number of nearly orthogonal than truly orthogonal vectors [16, 13].
3 Experimental analysis

We performed a series of experiments, including both PCA and RP. The recognition performance of both methods in terms of percentage of false positives and false negatives on a standard database was examined.

Experiments were divided into two parts. In the first part general properties of the RP were tested and compared to the PCA. In the second part we used RP in the VSN simulator in conjunction with our hierarchical feature-distribution (HFD) scheme. In both parts of experiments, PCA was used as a benchmark. The experimental protocol was as follows.

3.1 Experimental procedure

Database We have used the standard COIL-20 database, which consists of images of 20 different objects; each one is rotated with 5 degree angle interval, corresponding to 72 images per object. That sums up to 1,440 images for the whole database [22].

Preprocessing Image dimensions were the same for both methods, i.e., 128 × 128 pixels. First, the database was split into two parts of approximately same size. For every object, every second image was selected for recognition, and the remaining images were used to build PCA subspace. Images selected for recognition were transformed into the PCA and RP subspaces. This way feature vectors for both methods were obtained. Images that corresponded to zero orientation of each object and the corresponding features were used as a reference. This is consistent with the testing protocol, used in our VSN simulator [25].

Protocol All vector projections have been compared to the projections of zero orientation images and Euclidean distance was calculated. If the distance was below the predefined threshold $T$, the comparison resulted in a match between the reference image from the training set and the tested image. If the match was between images of two different classes (one class corresponds to all images of the same object), the number of false positives (FPs) was increased. If there was no match between the two images from the same class, number of false negatives (FNs) was increased. This way, results for a particular threshold $T$ were obtained. This protocol was conducted separately for PCA and RP.

The procedure was repeated for a range of thresholds, which yielded FPs and FNs percentages.

Additionally, to rule out the influence of seed experiments for RP were repeated ten times with different seeds of a pseudorandom generator.

All the experiments were conducted several times with different degrees of dimensionality reduction (i.e., for subspace dimensions of 512, 256, 128, 64, 32, 16, 8, 4, 2, and 1).

Benchmarking First, performance of the PCA was tested to obtain reference values for proportion of FPs and FNs. PCA transforms the data to a new coordinate system in which basis vectors follow modes of greatest variance in training data [23]. Therefore, PCA is a data dependent method. Every second image from the COIL-20 database was used to build a PCA subspace in advance. Properly constructed PCA feature vectors contain feature values that are ordered by decreasing importance in terms of the reconstruction quality of the original data.

The results are shown in Figure 1, which depict number of FPs and FNs in relation to the threshold $T$. We show only the results for subspace dimensions of 512, 16 and 1.

3.2 RP

Major portion of experiments were dedicated to the RP method. We explored the following questions.

- How do results vary with different seeds of used pseudorandom generator?
- Is transformation vector normalization really necessary?
- Does the method benefit from sorting of the transformation vectors based on the preserved variance in data?

Although sorting of the transformation vectors according to the preserved variance in the data is not common practice in RP (as it is in the PCA), we are interested if this would improve recognition performance of RP.

Random projection matrix $R$ was generated following the suggestion by Achlioptas [1] as defined in Eq.(5). The maximum dimensions of the matrix were 512 × 16384 elements. The Mersenne twister [20] was used as a pseudorandom generator. We verified that same seeds result in same pseudorandom sequences and then precalculated RP matrices for ten randomly generated seeds. In practical applications the matrices would be calculated on the fly on each of the cameras. Similar as with the PCA, every second image in the database was transformed to the random subspace. Procedure was repeated for all ten different RP matrices and Figures 2 – 5 were generated by running tests with different thresholds $T$. Results for RP with different random seeds were compiled to single graphs using boxplots, as seen in Figures 2 – 5. Finally, the whole procedure was repeated for the following scenarios.

- RP using normalized vectors (Figure 2).
- RP using non-normalized vectors (Figure 3).
- RP using normalized vectors, sorted according to the highest preserved variance in data (Figure 4).
- RP using non-normalized vectors, sorted according to the highest preserved variance in data (Figure 5).

Observing Figures 2 – 5 we can conclude the following:

With 512 feature vectors used there is practically no difference between the PCA and RP and the scatter around the median value in the RP results is negligible. When 16 feature vectors are used PCA outperforms RP regardless of normalization or RP feature vector sorting. In this case, PCA yields FP/FN rates of 20%, while RP yields FP/FN rates in 25% – 30% range. Scatter in RP results is slightly larger but still small at the point of intersection of FPs and
Figure 1: Recognition rates (FPs and FNs) for PCA depending on the number of features used.

Figure 2: Recognition rates (FPs and FNs) for RP (normalized feature vectors) depending on the number of features used.

Figure 3: Recognition rates (FPs and FNs) for RP (non-normalized feature vectors) depending on the number of features used.

Figure 4: Recognition rates (FPs and FNs) for RP (normalized and sorted feature vectors) depending on the number of features used.
Figure 5: Recognition rates (FPs and FNs) for RP (non-normalized and sorted feature vectors) depending on the number of features used.

FNs curves. With only one feature vector used performance of RP deteriorates significantly and scatter is even larger.

It can also be seen that there is no significant difference if RP vectors are normalized or not. Indeed, if we observe the length of RP projection vectors, shown in Figure 6 it is obvious that they already have similar lengths. This is not surprising due to the dimensionality of input data – each vector has 16384 elements which can take only three possible values. Therefore, by discarding RP vector normalization, only scaling factor is introduced into the transformation.

Figure 8 shows that there are differences in reconstructive power of individual RP vectors. However, those differences are too small to have major impact on method performance. Therefore, this implies that it did not pay off to sort 512 random vectors according to the preserved variance.

In contrast, Figure 7 shows that PCA indeed condenses the information in small number of features.

3.3 Application to simulator for VSNs

To test the performance of RP in our previously proposed hierarchical feature-distribution scheme [24, 25], we used a distributed network simulator. It runs on a standard desktop...
The simulator measures both the amount of traffic transmitted between the nodes and the number of nodes (hops) over which the traffic is transmitted.

For the experiments, we used a network consisting of 99 nodes, arranged in a $11 \times 9$ rectangular, 4-connected grid.

The simulator for VSNs is used for testing the hierarchical feature-distribution (HFD) across the network of visual sensors (cameras). HFD has an important property. It significantly reduces the amount of transmitted data in the task of distributed object recognition (e.g., recognizing the objects that have been seen by other cameras in the network). The efficiency of our method in terms of data transmission is directly related to the recognition efficiency of the used object recognition method, as recognition errors significantly increase amount of transmitted data. In our previous work, we have established that PCA is appropriate for use in our HFD scheme, however, distribution of subspace across the network would be extremely wasteful in terms of network traffic. If RP is used instead, the only thing that we have to distribute is the seed of the pseudorandom generator (maximum eight bytes if Matlab implementation of Mersenne twister pseudorandom generator is used as a reference). The HFD method also ensures that the final decision on the identity of the recognized object is made using all available features. Therefore, in terms of recognition accuracy we are not concerned with the lower performance of RP when small number of features is used. However, this affects the amount of transmitted data.

Experiment was divided in two phases. The first (training) phase measured the performance of the network during training. Twenty nodes, evenly distributed through the network, were injected with images of the twenty different objects from the database. Those images corresponded to the zero orientation in the COIL-20 database. Next, the simulation cycle was started, and, after the network traffic stopped, the statistics on the network load (number of hops and the total network traffic per sample) was recorded. The results for training and recognition for both PCA and RP are shown in Table 1.

It can be seen that substitution of PCA with RP degrades performance of the network. Recognition rate is almost the same, however, both traffic and number of hops in the recognition phase are increased due to the poor performance of RP with small number of features. However, our method still outperforms naive feature-distribution schemes, such as flooding by almost 1:2 [25] in terms of network traffic. Additionally, use of RP does not change performance of the network in training phase, since the format of data is practical identical for both PCA and RP. Nevertheless, considering the huge advantage of RP in distributing the subspace to all cameras, the RP method seems perfectly fit for the use with our hierarchical feature-distribution (HFD) scheme [25].

- It is a good substitute for PCA when dimensionality reduction is needed.
- The combination of HFD and RP still significantly reduces the amount of traffic across the network, even though this reduction is not as high as when HFD and PCA are used.
- A lower recognition rate of RP in comparison to PCA at same dimensionality does not influence the overall recognition rate of our simulated network, and the reduction in network efficiency is relatively small.

### 4 Discussion and conclusion

Focus of our work was random projection in a context of possible use in embedded vision systems. RP projects original high-dimensional data through the random matrix onto the low-dimensional space. Data in low-dimensional space consume less resources, which is usually constrained in embedded systems. RP is data independent, preserves the structure of the data without significant distortion, is computationally simple and it does not require distribution of shared subspace. On the other hand, while principal component analysis is known to give good results and has a lot of useful properties it is also computationally expensive and require distribution of shared subspace, which results in consuming more resources in the network. Therefore, our aim was to determine the performance of the RP implementation, which could be directly used in embedded cameras. Some minor modifications of RP (the absence of

<table>
<thead>
<tr>
<th>Recognition method</th>
<th>Training</th>
<th>Recognition</th>
<th>Recognition rate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hops</td>
<td>Traffic</td>
<td>Hops</td>
</tr>
<tr>
<td>PCA</td>
<td>614</td>
<td>44</td>
<td>614</td>
</tr>
<tr>
<td>RP</td>
<td>614</td>
<td>44</td>
<td>610</td>
</tr>
</tbody>
</table>

Table 1: Experimental results for hierarchical feature-distribution method
row normalization of the random matrix, and additional sorting of the rows of the random matrix considering highest variance in the data – as it is in the PCA) and their effect on the recognition accuracy were systematically explored. Performance of the RP was compared to the performance of the PCA in conjunction with recognition efficiency on the COIL-20 image data set. Results show surprisingly good performance of the RP in comparison to the PCA even without explicit orthogonalization (computationally wasteful) or normalization (important for preserving similarities in the low-dimensional space) of transformation subspace. Moreover, in our case even sorting of feature vectors in accordance to the preserved variance did not pay off. Our results indicate that RP can be used with our hierarchical feature-distribution scheme in visual-sensor networks, where RP can elegantly solve the problem of shared subspace distribution.

Acknowledgement
The work was supported by the research program P2-0095 and research grant 1000-07-310221, both by Slovenian research agency.

References
Extended Set of Local Binary Patterns for Rapid Object Detection

Jiří Třfný\textsuperscript{1,2} and Jiří Matas\textsuperscript{1}

\textsuperscript{1}Center for Machine Perception, Faculty of Electrical Engineering, Czech Technical University in Prague
matas@fel.cvut.cz

\textsuperscript{2}Eyedea Recognition s.r.o.
Pod Hybšmankou 2848/7, 15000 Praha 5
jiri.trefny@eyedea.cz

Abstract The paper presents two new encoding schemes for representation of the intensity function in a local neighborhood. The encoding produces binary codes, which are complementary to the standard local binary patterns (LBPs). Both new schemes preserve an important property of the LBP, the invariance to monotonic transformations of the intensity. Moreover, one of the schemes possesses invariance to gray scale inversion. The utility of the new encodings is demonstrated in the framework of AdaBoost learning.

The new LBP encoding schemes were tested on the face detection, car detection and gender recognition problems using the CMU-MIT frontal face dataset, the UIUC Car dataset and the FERET dataset respectively.

Experimental results show that the proposed encoding methods improve both the accuracy and the speed of the final classifier. In all tested tasks, a combination of the encoding schemes outperforms the original one. No LBP encoding scheme dominates, the relative importance of the schemes is problem-specific.

1 Introduction

Object detectors based on boosted combinations of efficiently computable features such as Haar wavelets or Local binary patterns (LBP) represent the state-of-the-art for a wide range of detection problem. In particular, detectors exploiting LBPs have achieved highly competitive results in areas including texture and dynamic texture classification [14, 15, 28, 29], face detection [4, 7, 26, 23], face recognition [2, 27, 25, 11], gender classification [20] and facial expression recognition [29, 30].

The LBP is a simple local descriptor which generates a binary code for a pixel neighbourhood. Despite its simplicity, a number of LBP modifications and extensions have been proposed. Most of the changes focus either on the definition of the location where gray value measurement are taken or on post-processing steps that improve discriminability of the binary code.

In this work, the power of LBP features is enhanced by introducing two new schemes for generating binary codes, also referred to as “rules”. The new rules are compatible with the original methodology, i.e. the same number of bits is generated. The new rules preserve an important property of the original LBP, the invariance to monotonic transformations of the intensity functions. As a novelty, one of the rules also possesses invariance to gray scale inversion. The new rules are intended to supplement and complement, not substitute, the original LBP coding scheme.

We experimentally show that, in conjunction with the algorithms for feature selection like AdaBoost and WaldBoost, the combination of different encoding rules improves accuracy and speed of the final classifier when compared with a classifier based on a single rule.

The new ensemble of LBP features is compared with the original and Haar-like features on a face detection task using CMU-MIT frontal face test set [17], on a car detection task using UIUC multiscale test [1] and on a gender recognition task using FERET dataset [16].

The paper is structured as follows. Section 2 introduces local binary patterns methodology and its modifications. In this section we also introduce two new encoding rules for binary code generation. Experimental validation and comparison of our extensions are presented in Section 3 and the paper is concluded in Section 4.

2 Local Binary Pattern and its modifications

Local binary patterns have gone through a large number of changes and adjustments, which lead to generalization or improvement of some of their specific characteristics. The changes can be viewed from several perspectives. In Section 2.1, changes from the perspective of the measurement processes are reviewed. Next, in Section 2.2, we look at encoding method for the measurements. Finally, in Section 2.3, two novel encoding methods are introduced.

2.1 What is measured

The local binary pattern [14] operator, also known as census transform [24], is a non-parametric gray-scale descriptor invariant to monotonic transformations of the intensity function. The basic version of LBP considers measurements from a 3x3 pixel square.

The binary code that describes the local texture pattern is obtained by thresholding the eight neighborhood pixel values by the gray value of the center, see Figure 1(a). The operator was extended to rotation symmetric and multiscale version [15], see Figure 1(b). This version of the LBP is
parametrized by the neighborhood size $P$ and the radius $R$ and is defined as

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c)2^p,$$

where

$$s(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

$g_p$ are gray values of pixels regularly spaced on circle and $g_c$ is the gray value of the center pixel. Gray values at non-integer positions are obtained by interpolation. Another encoding, the LGBP, was introduced by Zhang at al. [27], who calculate LBPs on images preprocessed with Gabor wavelets.

All the LBPs described above are commonly used in conjunction with classification of distributions (histograms), calculated in a semi-local neighbourhood. In detection and recognition approaches exploiting spatial appearance of features, single LBP measurements are unstable and sensitive to noise and localization. The problem was addressed by Zhang at al., who introduced a Multi-Block LBP (MB-LBP) [26], which is inspired by Haar features [22]. Instead of comparing pixel values, Zhang compares mean values of 3x3 adjacent rectangular blocks, which can be done in constant time using the integral image [22].

The MB-LBPs enable generating large sets of operators with different scales and aspect ratios, see Figure 1(c). Similarly to Haar features, integrating larger areas makes the measurements more stable and suitable for spatial appearance classification methods. However, this modification does not possess LBP’s invariance to monotonic intensity transformations, only invariance to affine intensity changes is preserved. The MB-LBP feature also appears in the literature as the Locally Assembled Binary (LAB) feature [23].

2.2 Encoding methods

Improvements of LBP aimed at modifying the resulting binary code started with the rotation symmetric and multiscale LBPs of Ojala [15]. The rotation invariant encodings, denoted $LBP_{P,R}^{\text{ori}}$ (which can be found also as Advanced LBP - ALBP $P,R$ [10]), are restricted to a subset of so-called “uniform” patterns ($LBP_{P,R}^{\text{uni}}$).

Froba at al. introduced a modified census transform [4], which was adopted also as a modified LBP (mLBP) [21]. The Modified LBP uses the mean value of all measured pixels as a threshold, so the final code then generates $2^8 - 1 = 511$ unique values instead of $2^8 = 256$ of LBP codes. Because of compatibility with the original LBP, we adopted only code generated by eight border pixels with $2^8$ unique values, see Figure 2(b). Heikkilä at al. in [6] introduced a center symmetric LBP (CS-LBP) modification for description of interest regions. Their rule encodes the sign of the difference of two border pixels symmetrically placed due to the center, thus the final code of CS-LBP generates $2^4 = 32$ unique codes.

2.3 The novel encoding methods

To introduce new encoding rules we were motivated by spatial appearance classification models, which enables to effectively combine different features. The evaluation complexity of the model does not increase, provided that the computation cost of each feature is approximately equal. Extension of the feature set from which the features are chosen increase only training time but not the evaluation time. This lead us to propose encoding rules, which should not be competitive with LBP but complementary and extend a set of feature candidates. In order to preserve compatibility with LBP, we set the restriction on dimension of generated binary code to be the same as the original.

Transition Local Binary Patterns (tLBP) - The LBP encoding rule thresholds the neighbor gray values by its center pixel value. This gives rough knowledge of pixel with respect to the center one, but relations between pixels with the same binary value are lost. Binary value of transition coded LBP is composed of neighbor pixel comparisons in clockwise direction for all pixels except the central, see Figure 2(c). Thus this rule encodes relation between neighbor pixels. It can be
also seen as an information about partial ordering of border pixels. Each sequence of the same binary values indicates ordered sequence of pixel intensities.

More precisely, let $g_p$ correspond to gray value $p$-th neighbor of center pixel, then

$$tLBP_{P,R} = s(g_0 - g_{P-1}) + \sum_{p=1}^{P-1} s(g_p - g_{p-1})2^p.$$ (2)

We can see that tLBP is gray-scale invariant and can also benefit from rotation invariant extension and uniform extension of LBP ($LBP_{P,R}^{u,b}$).

**Direction coded Local Binary Pattern (dLBP)** - Motivation of dLBP is to provide better information of local pattern in sense of direction functions similarly to CS-LBP. For simplicity, let us consider the basic LBP operator. We can see that there are four base directions through the center pixel in LBP, see Figure 2(d). We encode intensity variation along these directions into two bits, thus the binary word has the same length as the original LBP. In contrast to the CS-LBP, we also use center pixel information for encoding. The first bit encodes, whether the difference of border pixels due to the center one grows or falls. In Figure 3 we can see comparison of LBP and dLBP rules for a given direction. Both the LBP and the dLBP rules encodes if center pixel is an extrema. Unlike the LBP rule, the dLBP does not encode it as maximum or minimum but encodes if sign of first and second differential is the same. This gives to the dLBP not only gray-scale intensity invariance property, but also the intensity inversion invariance property.

Formally, let $LBP_{P,R}$ have $P = 2^p\rho$ neighbors, then

$$dLBP_{P,R} = \sum_{p'=0}^{P-1} \left(s(g_{p'} - g_c)(g_{p'} + p' - g_c)2^{2p'} + s(g_{p'} - g_c) - |g_{p'} - p' - g_c|2^{2p'+1}\right).$$ (3)

3 Experiments

In all the detection and classification experiments, only the multi-block extensions of LBP were evaluated as they have outperformed the standard LBP. The extended MB-LBP set (EMB-LBP) included the MB-LBP, mMB-LB, tMB-LBP and dMB-LBP, see Figure 2(a-d).

The tests evaluated performance of different LBP types in the process of boosting a detector (or a classifier). The

![Figure 3: Examples of generated codes and schemes of possible pixel intensity values for a given pixel sequence: (a) LBP encoding rule, (b) dLBP encoding rule](image)

![Figure 4: Frontal face detection - The ROC curve on the CMU-MIT data set](image)

![Figure 5: Some detection results on the CMU-MIT data set](image)

EMB-LBP set was tested on face and car detection tasks using the WaldBoost[19] detector and on gender recognition task using AdaBoost classifier. The reason is that for gender recognition, speed of the classifier is not important as only one window per face is classified. On the other hand, in the car and face detection problems, hundreds of thousands of windows are evaluated and speed, the main advantage of WaldBoost over AdaBoost, is a critical parameter.

WaldBoost is an AdaBoost-based algorithm which automatically builds a fine-grained detection cascade of the Viola and Jones type [22] based on Wald’s sequential probability ratio test (SPRT). The training runs in loops, the first iteration is a standard AdaBoost learning search for the best weak classifier. Then the Wald’s thresholds are estimated on a large pool of data (we used $20,10^9$ samples). After that, the pool is pruned and bootstrap strategy is used to collect non-object examples. To speed-up the AdaBoost learning step, a smaller set was sampled from the pool using QWS+ strategy [8]. The weak classifiers are build on MB-LBPs by estimating the weighted error for each code as in the confidence-rated classification approach [18], which enables a fast look-up table based implementation.

In all experiments, three classifiers were trained. The first was learned with Haar features (including six types of features), the second with MB-LBP features and the third one with the EMB-LBP feature set.
3.1 Face detection
The face detectors were trained on 5500 face images and on more than 3000 background images. We set the minimum resolution of the detector to 24x24 pixels and its length to 1000 weak classifiers. SPRT parameters were set to allow 10% false negative rate and no false positives on the training data.

The detectors were tested on standard the CMU-MIT frontal face database [17], which consists of 130 images with 507 labeled frontal faces. Some detection results can be seen at Figure 5. The ROC curves for the three detectors are shown in Figure 4. The detector using the EMB-LBP feature set slightly improves recall for all levels of false positive rates.

3.2 Car detection
The side car detection performance is evaluated on the UIUC car dataset [1], which consists of 550 positive training samples and the multi-scale and the single-scale test sets. We trained the detectors on 16x40 pixels windows and allowed 5% false negative rate on the training set; classifier length was set to 500 features.

For the experiment, we chose the multi-scale test set, which consists of 108 images containing 139 car side views. The set includes instances of partially occluded cars, cars that have low contrast with the background, and images with highly textured backgrounds. Sample detection results are displayed in Figure 7.

As is common for the UIUC Car dataset, we measure the performance by the 1-precision vs. recall curve. Figure 6 shows the curves for different feature sets. The detector using the EMB-LBP feature set improves recall for all levels of precision and dominates both MB-LBP features to Haar features. The difference in performance is impressive for high precisions where a recall of 95% was achieved with 100% precision. Table 1 compares recalls at the point of equal precision and recall with the state-of-art results. The EMB-LBP is highly competitive.

3.3 Gender recognition
The gender recognition experiment was carried out on the Feret data set [16], which is a standard data set for face recognition task and has also been used as a gender recognition benchmark data set. Data set contains several photos of persons with different pose; we used only frontal images labeled "fa" and "fb" in the database. The dataset includes 1006 persons (599 males, 407 females). For evaluation, we adopted Baluja’s methodology [3] which uses 5-fold cross validation. Each partition splits the training and testing data 80:20 in such a way that each individual appears only in the training set or the test set. It is important to note that Moghaddam at

Table 1: Recalls on the UIUC Car dataset at the point of equal precision and recall.

<table>
<thead>
<tr>
<th>method</th>
<th>recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agarwal at al. [1]</td>
<td>39.6%</td>
</tr>
<tr>
<td>Fritz at al.[5]</td>
<td>87.8%</td>
</tr>
<tr>
<td>Mutch at al.[13]</td>
<td>90.6%</td>
</tr>
<tr>
<td>Lampert at al.[9]</td>
<td>98.6%</td>
</tr>
<tr>
<td>WaldBoost, Haar</td>
<td>91.4%</td>
</tr>
<tr>
<td>WaldBoost, MB-LBP1</td>
<td>95.7%</td>
</tr>
<tr>
<td>WaldBoost, EMB-LBP1</td>
<td>97.1%</td>
</tr>
</tbody>
</table>

1 each person in the data set is included only either in the training or test set
2 our implementation

Table 2: FERET dataset - gender classification accuracy.

<table>
<thead>
<tr>
<th>Training algorithm</th>
<th>face size</th>
<th>trn/tst</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdaBoost, pixel comparison [3]</td>
<td>20x20</td>
<td>YES</td>
<td>94.4%</td>
</tr>
<tr>
<td>SVM (RBF) [3]</td>
<td>20x20</td>
<td>YES</td>
<td>93.5%</td>
</tr>
<tr>
<td>SVM (RBF) [12]</td>
<td>20x20</td>
<td>NO</td>
<td>96.6%</td>
</tr>
<tr>
<td>AdaBoost, LBP[20]</td>
<td>120x144</td>
<td>?</td>
<td>95.7%</td>
</tr>
<tr>
<td>AdaBoost, Haar2</td>
<td>20x20</td>
<td>YES</td>
<td>92.4%</td>
</tr>
<tr>
<td>AdaBoost, MB-LBP2</td>
<td>20x20</td>
<td>YES</td>
<td>93.8%</td>
</tr>
<tr>
<td>AdaBoost, EMB-LBP2</td>
<td>20x20</td>
<td>YES</td>
<td>94.6%</td>
</tr>
</tbody>
</table>

3 each person in the data set is included only either in the training or test set
2 our implementation
Figure 9: Examples of correct (a) and wrong (b) classification of sex between men (left) and women on the FERET dataset. Note that gender classification for images showed in (b) is difficult even for humans.

Figure 10: Frontal face detection - the average number of used features per scanning position

Figure 11: Car detection - average number of features used per scanning position

<table>
<thead>
<tr>
<th></th>
<th>Haar</th>
<th>MB</th>
<th>mMB</th>
<th>tMB</th>
<th>dMB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Haar</td>
<td>1.000</td>
<td>1.114</td>
<td>1.117</td>
<td>1.207</td>
<td>1.242</td>
</tr>
<tr>
<td>MB-LBP</td>
<td>0.898</td>
<td>1.000</td>
<td>1.003</td>
<td>1.011</td>
<td>1.114</td>
</tr>
</tbody>
</table>

Table 3: Comparison of the relative feature evaluation time...

3.5 Feature preferences

AdaBoost learning algorithm can be seen also as a benchmark tool for feature strength comparison, if the same classifier is used. It uses greedy approach to minimize training error and at each stage chooses the best weak classifiers. Thus, frequency of feature selection indicates how often a given feature dominates the others. However, it does not show how much better than the other it was. Dependence of feature selection on the length of classifier is shown in Figure 12. It can be seen that for different tasks the ratio of representation of features differs significantly. For face detection the contribution of dMB-LBP features are negligible, but they dominate others for gender recognition. The standard MB-LBP features [26] perform surprisingly poorly and as Figure 12 shows for car side detection they were not used at all.

4 Conclusions

Two new encodings of LBPs have been presented. We have trained spatial appearance models based on multi-block ma-
measurements of LBP. Instead of direct comparison of every new rule with other LBP methodologies, we have used a trained classifiers using an ensemble of different LBP encoding rules. In experiments we have made comparisons with standard LBP encoding rule and traditional Haar features. We have tested detectors based on the extended set of LBP features on the CMU-MIT frontal face data set and on the UIUC car side data set. Experiments on gender recognition task used the Feret dataset. In all cases, the extended set of LBP features dominates both the LBP features and Haar features. For the detection tasks, the proposed LBP set has improved speed of learned detectors, in case of the face detection task almost two times. The price paid for achieved improvements of the detectors and classifiers has been only the increase in the training time. In experiments we have shown that the importance each of the encoding rules depends on the task and there is no dominant rule.

Acknowledgement

The research was supported by EUREKA project OE09009 OLIGOSYNT and by Czech Science Foundation Project 102/07/1317.

References


Extrinsic Autocalibration of Vehicle Mounted Cameras for Maneuvering Assistance

Thomas Ruland\textsuperscript{1,2}, Tomas Pajdla\textsuperscript{1}, and Lars Krüger\textsuperscript{2}
\textsuperscript{1}Czech Technical University of Prague
rulantho@cmp.felk.cvut.cz, pajdla@cmp.felk.cvut.cz
\textsuperscript{2}Daimler AG, Ulm
lars.krueger@daimler.com

Abstract This paper introduces a fast and robust approach to extrinsic autocalibration of vehicle mounted cameras for Bird’s Eye View\textsuperscript{1} applications. We limit ourselves to orientation estimation of the camera with respect to the vehicle coordinate frame due to the application’s higher sensitivity to rotation than to translation. The presented algorithm predicts the movement of image features based on a vehicle movement model. This prediction gets compared with temporal correspondences extracted from image pairs to yield an error measure for camera orientation. It stays robust under bad illumination and texture conditions by accumulating sparse correspondences over time. The presented solution both reduces costs and improves maneuvering safety by replacing end of assembly line calibration with continuous lifetime adaption to changes in camera orientation.

1 Introduction

This paper introduces an extrinsic autocalibration solution for vehicle mounted cameras used in Bird’s Eye View applications. Bird’s Eye Views have become a valuable driver assistance in parking scenarios. They have been studied in various approaches for heavy vehicles such as trucks [4] and vans [5] as well as for passenger cars [8, 12].

For a Bird’s Eye View of the vehicle’s surroundings to be calculated from camera data, the system requires accurate information about each camera’s parameters. Intrinsic parameters (e.g. focal length) are generally available for each camera and determined by construction. Extrinsic parameters, like rotation and translation, have to be calibrated at the end of the vehicle assembly line since the camera mounting procedure is subject to tolerances. Vehicular Bird’s Eye Views are especially sensitive to rotational errors in camera calibration, while being less sensitive to translational errors. Section 3 shows, that with the given camera mounting positions, extrinsic autocalibration can be limited to orientation estimation.

Calibrating extrinsic parameters at the end of the assembly line has several disadvantages. The necessary calibration objects for all four cameras require a lot of space and the actual calibration process takes time. Both space and time are costly in the complex assembly lines. Conventional calibration also has to be repeatable in the service garage where precisely alligned calibration objects are also expensive.

Recently presented models, such as the Nissan Elgrand and the Honda Odyssey, show feasible camera mounting positions for cars. While the front camera is mounted directly to the vehicle body, the side cameras are located in the side mirrors. Our tests with a similarly constructed test vehicle have shown that camera mountings at these locations are subject to constant mechanical stress. Vibrations cause the cameras to gradually change their pose during lifetime. Figure 2 shows the impact of this change on the Bird’s Eye View image.

The change of pose is intensified by mechanically moving camera mountings, like retractable side mirrors or a retractable rear view camera (introduced in Porsche Cayenne). The moving parts’ tolerances result in a total tolerance of camera orientation of up to 1°. The actual camera orientation is thus not exactly known and changes every time the camera is activated. Calibration has to be verified and maybe updated every time the camera moves out of the vehicle body. In case of retracting camera mountings, the camera observes the world only during parking maneuvers. The system must be able to work properly in bad illumination and with little texture on the ground since possible situations for calibration are rare.

\footnotesize{\textsuperscript{1}Also known as Top-View}
Orientation changes, which were caused by small collisions may dislocate the mirror and its camera not gradually but almost instantaneously. Because of this, no assumption about the initial camera orientation is made in this paper, except that it is able to observe the road.

For a Bird’s Eye View system, featuring lifetime autocalibration, factory calibration after construction and repairs can be omitted. This greatly reduces costs and complexity. Adapting calibration to lifetime changes increases the accuracy of the Bird’s Eye View and thus improves safety during parking.

![Figure 2: Bird’s Eye View image of a chess board pattern placed in front of the vehicle (tile side length is 500 mm). The grid overlay represents the vehicle frame. The left image was created using calibration parameters, which were 4 months old. The effect of the lifetime changes in camera pose could mislead the driver while maneuvering. The parameters used to create the right image were estimated by the proposed algorithm.](image)

The remainder of this paper is organized as follows. Section 2 collects previous work, which was done in the field of autocalibration of vehicle mounted cameras, and discusses its applicability. Section 3 presents the proposed algorithm to estimate camera orientation by movement prediction of image features. Section 4 reports from experiments, validating accuracy and robustness of the proposed approach, and Section 5 summarizes the achieved solution as well as open challenges.

2 Previous work

Esquivel et al. [6] presented a solution for estimating relative pose of two rigidly coupled cameras with non-overlapping views. They interpreted the task as a variation of the classical hand eye calibration and relied on a structure from motion algorithm for successive camera pose estimations. As described in their work [6], this approach would theoretically be suited for the targeted scenario of calibrating the extrinsic parameters of a camera against a vehicle. Though in case of retractable cameras, which can observe the world only in the short time frame of a parking maneuver, calibration has to be robust and reliable even in poor illumination and texture conditions. This robustness can hardly be achieved using Esquivel’s approach since camera motion estimation requires numerous correct temporal correspondences over frame pairs. The experiments in this paper show, that the average number of correct correspondences per frame pair may reduce to 1 during sunset.

Klappstein et al. [10] presented an approach to estimate the current road plane and the extrinsic parameters of the camera. It was designed to compensate pitch movements of vehicles. Bird’s Eye View applications would directly benefit from these constantly adapted parameters because pitch movements cause distortions in the Bird’s Eye View image. Klappstein’s approach tracks the parameters of the ground plane using a Kalman filter. But still at every point in time it solely uses the correspondences of a single frame pair for parameter estimation. If a frame’s correspondences are sparse or erroneous, the estimation fails. The presented Kalman filter is able to bridge short time windows of sparse correspondences but works poorly if correspondences are constantly sparse.

Road features like lane boundaries and road markings are often used for autocalibration of vehicle mounted cameras [1, 2, 11, 14]. With the camera mounting positions as described in the previous section, road features can only be used for the front camera. Figure 1 illustrates the field of view of a side camera, which looks downwards on one single side of the road. This perspective makes vanishing point estimation difficult. The calibration of the rear camera also cannot benefit from highway lane markings since it is retracted after parking.

Stereo based approaches are not applicable due to the barely overlapping views of the cameras.

3 Orientation estimation

This section presents an algorithm, which overcomes the limitations of the previous work. It is less sensitive to illumination and especially tailored to the application in a Bird’s Eye View maneuvering assistant system.

To evaluate the quality of a Bird’s Eye View and the used camera parameters, which were either created manually or by an autocalibration algorithm, the back-projection error of the capturing camera is examined. This error is given by the difference of an object’s ground truth vehicle frame location and the location which resulted from back-projecting its position in the image frame onto the ground. The back-projection error is an intuitive quality measure of a Bird’s Eye View because it describes how much obstacles in the world could be nearer or further away than they appear in the Bird’s Eye View image.

Using the back-projection error the special sensitivity of the Bird’s Eye View to rotational errors in camera parameters becomes apparent. While keeping calibration parameters constant, the camera was rotated by 0.8° around its pitch axis. This caused an average back-projection error of 100 mm over the visible area in the Bird’s Eye View. When keeping rotation constant and creating the same back-projection error by pure translation the camera needs to be moved by either 30 mm above its usual location or 100 mm to the sides of the vehicle. Both translations exceed the typical tolerances and lifetime drift. Thus orientation estimation is the important part of camera calibration for Bird’s Eye View applications while translation is known sufficiently well by construction of the vehicle.

The following sections describe the algorithm to perform orientation estimation of vehicle mounted cameras for Bird’s Eye View applications.
3.1 Algorithm principle

The basic principle of the proposed algorithm is predicting the movement of image features, which is induced by the known movement of the vehicle. The prediction of the feature movement is compared to each feature’s actual movement in the image. The algorithm uses the resulting prediction error as an error measure for camera orientation. Figure 3 illustrates the prediction in detail.

![Image Frame diagram](image)

**Figure 3:** Given the camera orientation to be evaluated, the algorithm first projects image features back onto the ground in the ego centric 2D vehicle frame. Second, it transforms each feature in the vehicle frame according to the movement of the world against the vehicle. The world is assumed to be rigid during this step. Third, the algorithm projects the features into the camera image and compares the result with the temporal correspondences measured by a correspondence algorithm in the two subsequent images.

The proposed solution to the problem of sparse temporal correspondences (e.g. in the evening) is to accumulate information triples $(y_t, y_{t+1}, T_{t+1})$ over time. Each triple consists of a feature location $y_t$ at time $t$, a location of the corresponding feature $y_{t+1}$ in the subsequent frame at time $t + 1$ and the movement $T_{t+1}$ of the world against the vehicle frame in-between both time steps. The algorithm accumulates an unordered set of these triples over time, gathering both long tracks of a feature across the image as well as correspondences which could only be established over a single frame pair. The error measure of camera orientation is based on a large set of triples. It exploits the large data base instead of being limited to the correspondences of a single frame pair. This accumulation increases robustness to poor illumination and texture conditions.

3.2 Projection model

Throughout this paper, the world around the vehicle is assumed to be flat and perceptions violating this assumption are regarded as outliers. Under this assumption, points from the flat world plane $x \in \mathbb{R}^2$ can directly be transformed into points in the image frame $y \in \mathbb{R}^2$, using the homography $H \in \mathbb{R}^{3 \times 3}$ as $y \equiv H x$. As described by Hartley and Zisserman [9, p.196], the homography can be decomposed into

$$H = P D = K R \left[[I \mid -t]\right] D$$  \hspace{1cm} (1)

where $K \in \mathbb{R}^{3 \times 3}$ is a pinhole camera calibration matrix, $R \in \mathbb{R}^{3 \times 3}$ and $t \in \mathbb{R}^3$ the rotation and translation of the camera against the vehicle frame respectively. Matrix $D$ adds a zero row to the two dimensional vehicle frame vectors in order to transform them to their three dimensional extension in the ground plane.

$$D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$A = ([I \mid -t] D)$ is an invertible $3 \times 3$ matrix (for $t_y \neq 0$).

$$A = \begin{bmatrix} 1 & 0 & -t_x \\ 0 & 1 & -t_y \\ 0 & 0 & 1 \end{bmatrix} \quad A^{-1} = \begin{bmatrix} 1 & \frac{t_x}{t_y} & 0 \\ 0 & 1 & \frac{t_y}{t_x} \\ 0 & 0 & 1 \end{bmatrix}$$

With $K$ and $R$ being invertible matrices as well, the inverse homography can be given. It transforms from the image to the two dimensional vehicle frame.

$$H^{-1} = A^{-1} R^{-1} K^{-1}$$

with $R^{-1} = R^T$ since $R$ is an orthonormal rotation matrix.

3.3 Ego motion

Ego motion of the vehicle is estimated using the model of Ehlgen et al. [4] for truck trailer combinations. Disregarding its trailer equations yields a standard single track model. Based on steering angle and movement distance of the vehicle, the model estimates the covered relative translation and rotation. This information is encoded in a homogeneous transformation matrix $T \in \mathbb{R}^{3 \times 3}$ in the vehicle frame.

The matrix $T$ is used to transform observations, which are registered relative to the ego centric vehicle frame, over various vehicle positions. To illustrate this, let $x_t \in \mathbb{R}^2$ describe the vehicle frame location of an object observed at time $t$. At time $t + 1$, the world moved relative to the vehicle by $T_{t+1}$. The previously observed point can now be located at $x_{t+1}$. Both measurements are related by

$$x_{t+1} = T_{t+1} x_t .$$

Using the wheel revolution sensors and the wheel circumference, movement distances can be acquired in 25 mm steps. This leads to discretization errors in some frames when the actual movement distance noticeably deviated from the discretized measurement. This effect was also discernable when looking at movement distances over time. While these values should be rather stable over short time windows, there was a high frequency noise with amplitudes of about 25 mm. Applying the assumption of smooth temporal velocity changes, movement distances are averaged within a sliding window of 3 frames both into past and future of each measurement.

3.4 Correspondence extraction

Temporal correspondence extraction is based on the work of Stein [15] with pixel precision. This method was selected because of its capability of handling long feature displacements. Correspondences are extracted from the original fish-eye images. Afterwards the algorithm rectifies the correspondences to a pinhole camera using the two parameteric nonlinear model for fish-eye lenses of Micusik and Pajdla [13]. Its intrinsic parameters $a$, $b$, $r$ and $c$ have been estimated previously and are assumed to be known. Their marginal change during lifetime does not affect the Bird’s Eye View significantly and is ignored.
3.5 Error measure

In this paper, the basic idea of the error measure for a given camera orientation is to use it for prediction of the movement of image features and compare the prediction with measured temporal correspondences. The camera orientation is encoded by the parameter vector \( \Theta \in \mathbb{R}^l \). The resulting homography from the vehicle to the image frame \( H_\Theta = K R_\Theta ( [I | -t] D ) \) of Equation (1) depends on this orientation.

In practice, the comparison of predicted and measured correspondences in the image favors features on distant objects because short feature displacements create small prediction errors in the image. This is disadvantageous because the signal-to-noise ratio during correspondence extraction decreases on short correspondences. To prevent this effect, the comparison is carried out in the vehicle frame instead. The vehicle frame location \( x_t \) of an image feature \( y_t \) can be determined using the inverse homography \( H_\Theta^{-1} \).

\[
x_t = \alpha_1 H_\Theta^{-1} y_t
\]

\[
x_{t+1} = \alpha_2 H_\Theta^{-1} y_{t+1}
\]

where \( \alpha_1 := \frac{1}{(H_\Theta y_t)} \) and \( \alpha_2 := \frac{1}{(H_\Theta y_{t+1})} \) normalize the scale.

The prediction of the feature movement in the vehicle frame is based on the ego motion \( T_{t+1} \) of the vehicle.

\[
\tilde{x}_{t+1} = \alpha T_{t+1} H_\Theta^{-1} y_t
\]

with \( \alpha := \frac{1}{(T_{t+1} H_\Theta y_t)} \).

Starting from an image location \( y_t \in \mathbb{P}^2 \) at time \( t \), \( H_\Theta^{-1} \) projects \( y_t \) back onto the flat ground plane in the vehicle frame. \( T_{t+1} \) then applies the vehicle motion and results in the predicted vehicle frame location \( x_{t+1} \) at time \( t+1 \).

Assuming a rigid and flat world as well as an ideal camera orientation \( \Theta \), both the predicted \( x_{t+1} \) and the measured feature location \( x_{t+1} \) should be identical. Comparing them yields an error measure for camera orientation \( \Theta \) in the vehicle frame.

\[
E_\Theta := \|x_{t+1} - \tilde{x}_{t+1}\|_2
\]

It is possible to incorporate multiple information triples \( (y_{(i)}, y_{(i+1)}, T_{(t+1)}) \), \( i \in [1, N] \), \( N \in \mathbb{N} \). Each triple can be recorded at a different point in time \( t (i) \). With the corresponding prediction

\[
\tilde{x}_{t(i)+1} = \alpha T_{t(i)+1} H_\Theta^{-1} y_{t(i)}
\]

the error measure is given by

\[
E_\Theta := \sum_{i=1}^N \| \alpha_1 H_\Theta^{-1} y_{t(i)+1} - \alpha_2 T_{t(i)} H_\Theta^{-1} y_{t(i)} \|_2^2
\]

with

\[
\alpha_1 := \frac{1}{(H_\Theta^{-1} y_{t(i)+1})} \quad \alpha_2 := \frac{1}{(T_{t(i)} H_\Theta^{-1} y_{t(i)})}
\]

This formulation enables the gathering of correspondences over time and thus increases robustness to sparse correspondences. The parameter search becomes a batch process and can rely on a much larger data base. This error measure is used in the subsequent sections for camera orientation estimation.

3.6 Robust optimization

Preliminary test have shown, that a well balanced distribution of correspondences over the image and the range of steering angles of the vehicle is crucial. The algorithm calculates a three dimensional histogram over these three parameters with two bins for both image dimensions and 3 bins for the steering angles. Then the algorithm randomly selects an equal amount of correspondences from each bin for optimization.

The algorithm performs an optimization based on RANSAC [7]. In each run, the optimization algorithm creates a tentative set of information triples. It minimizes the error measure from Equation 2 based on the correspondences in the tentative set over the parameter space of possible rotations, which results in \( \Theta^* \). It selects the inliers by applying a threshold of 10 mm to \( E_\Theta \), calculated on each element of the complete set of information triples individually. This procedure is repeated 20 times. The final result is the parameter vector \( \Theta^* \), which yields the largest number of inliers. The individual steps are described detailedly in the following.

The tentative set is composed of one correspondence randomly selected from each histogram bin, which results in a 12 correspondences total. This approach ensures, that optimization does not over adapt to a small subset of the gathered correspondences.

Rotations are parametrized using their axis \( n \in \mathbb{R}^3 \) with \( \|n\|_2 = 1 \) and angle \( \Theta \in \mathbb{R} \) measured in radians. Both are encoded and stored as \( r = \Theta n \), thus resulting in a three dimensional parameter space \([-\pi, \pi] \). The corresponding rotation matrix \( R \) can be calculated from \( n \) and \( \Theta \) as described by Craig [3, p.441].

The parameter space is initially limited to orientations which allow the camera to observe the ground plane. This constraint is enforced by requiring the principle axis of the camera to intersect the ground plane in front of the camera.

To minimize the error measure \( E_\Theta \) from Equation 2 over the parameter space, the algorithm performs a grid search with equidistantly spread samples. The sample distance is 0.0349 in all three dimensions. In case of a rotation around a coordinate axis, this equals angular steps of \( 2^\circ \). Even though being time consuming, the grid search approach has been selected because of its simplicity and robustness to local minima. The computation time also does not vary depending on the input data, which is an advantage in the targeted automotive application.

The RANSAC optimization is executed every 5 s during accumulation of correspondences. The resulting parameter vectors are averaged component wise over time to yield the overall result.

4 Experiments

Fig. 4 shows the testing vehicle used for the Bird’s Eye View and the orientation estimation. The experimental camera locations are similar to the ones used in Honda’s Odyssey or Nissan’s Elgrand. Two cameras are mounted to the side mirrors, which get retracted when the vehicle is switched off. The front and rear cameras are mounted directly to the ve-
Extrinsic Autocalibration of Vehicle Mounted Cameras for Maneuvering Assistance

Each camera features a resolution of \(640 \times 480\) pixels and perceives a field of view of \(185^\circ\) using fish-eye lenses. The overlapping regions between the views are very small and located in image regions where resolution is low. The region of interest on the ground, which is displayed by the Bird’s Eye View, spans \(3\ m\) to the front and side of the vehicle.

![Image of vehicle](image.png)

**Figure 4:** Four cameras are mounted to the test vehicle to capture the input imagery for the Bird’s Eye View. The side cameras are mounted to the side mirrors, which get retracted when the vehicle is switched off. The front and rear cameras are mounted directly to the vehicle body.

The cameras of the Bird’s Eye View system deliver frames at \(30\ Hz\). The error measure requires both temporal correspondences and the vehicle ego motion. Since ego motion of on-board odometry sensors arrives at \(10\ Hz\) only every third image needs to be searched for correspondences. At parking velocities of approximately \(5 – 10\ km/h\), this results in typical correspondence vector lengths of above 4 pixel in image regions viewing the ground within the interesting \(3\ m\) around the vehicle. A lower pixel threshold for correspondence length of 4 pixel, limits the processed correspondences to this region of interest around the vehicle.

The back-projection error, which Section 3 describes, is used to assess the quality of orientation estimation results in this evaluation. The chess board pattern illustrated in Figure 5 was placed at a known location in front of the testing vehicle. The known vehicle frame locations of the chess board corners and their corresponding manually labeled image locations are the input data for the back-projection error evaluation. This error is then averaged over the corners in potential viewing areas of the Bird’s Eye View, covering \(2\ m\) and \(3\ m\) of ground around the vehicle.

During the experiments, orientation optimization was performed every \(5\ s\) while accumulating correspondences. With the increasing amount of correspondences, the orientation estimations improve in quality. The results of the optimization runs are averaged over a sliding time window of the last 10 optimizations. Limiting the average to this short history prevents the first, usually bad estimations on a small dataset, from disturbing later results on larger amounts of data.

The following experiments evaluate performance and certain characteristics of the presented orientation estimation algorithm. First, they show its general suitability for the given problem statement in three maneuvering situations at low vehicle speeds and in bright daylight. Second, experiments with varying environment conditions analyze the algorithm’s sensitivity to illumination.

### 4.1 Maneuvering

In the maneuvering experiments the authors assumed that the Bird’s Eye View activates automatically at vehicle speeds slower than \(20\ km/h\). Thus the orientation estimation is already able to perform a first estimation during approach of the parking spot. This way, the driver can use a well calibrated Bird’s Eye View, when maneuvering into the actual spot. Including approaching and leaving the parking spot, each maneuver took approximately \(2\ min\) time.

Figures 6, 7 and 8 show results of the orientation estimation during three typical maneuvering scenarios. Driving straight along a parking lot with vehicles on both sides, parallel parking and reversing into a parking spot, which is orthogonal to the street. The dash dot line shows the number of inliers of the optimized camera orientation in multiples of 1000 inliers. The dashed line illustrates the average back-projection error in the Bird’s Eye View image covering an area, which extends the vehicle by \(2\ m\). The solid line shows the same error for an area of \(3\ m\) around the vehicle.

In all three situations, the back-projection error in the \(3\ m\) area reaches a value below \(100\ mm\) after less than \(50\ s\). The error in the \(2\ m\) area, which is of special importance when maneuvering close to obstacles, even goes below \(50\ mm\). In the two parking situations (Figures 7 and 8) the parking spot was reached after approximately \(60\ s\). In both situations, the driver can already use a sufficiently precise Bird’s Eye View when maneuvering into the spot.

In Figure 6, the back-projection errors of both regions feature an exceptional, temporary minimum at \(t \approx 100\ s\) and increase afterwards back to \(100\ mm\) and \(50\ mm\) respectively. The exact reason for this behavior is still not known for sure. A detailed analysis of the sequence unveiled the road to feature a slight hill with the peak being reached by the vehicle at this point in time. The hill is barely noticeable by the driver. A hypothesis for the following increase of the
Figure 6: Orientation estimation while driving slowly over a parking lot with vehicles on both sides. The number of inliers is given in multiples of 1000 inliers.

Figure 7: Orientation estimation while reversing into a parking spot, which is parallel to the street.

Figure 8: Orientation estimation during maneuvering into a parking spot, which is orthogonal to the street.

back-projection error could be the integration of correspondences before and after passing the hill. 50 mm seem to be a quality limit of the current configuration for roads, which are not noticeably hilly. A solution might be to reduce the inlier threshold during RANSAC optimization. This behavior has to be addressed in future to overcome this deficiency.

Also the sudden drop of the back-projection error towards the end of the sequence in Figure 8 is still not known. It may be caused by the significant expansion of the accumulated data base due to dense correspondences, which is also visible in the increasing number of inliers.

In the straight driving experiment, the number of inlier correspondences increases linearly during the complete maneuver. In both parking maneuvers the number of inliers stops growing for a time between 60 s and 100 s into the maneuver. This time coincides with the period of maximal steering angles at the corner of the parking spot. In this situation, the feature movement prediction does not model the measured temporal correspondences as good as when driving at lower steering angles. This could be caused by the ego motion model, which does not take the tire slippage at high steering angles into account. This effect will be subject to future research.

4.2 Sensitivity to illumination

In the illumination experiments, the testing vehicle drove along a parking lot on approximately the same course at three successive points in time during sunset. Figures 6, 9 and 10 show the results of orientation estimation in the three experiments in order of increasing darkness.

Note: The unit of number of inliers has been reduced to multiples of 10 inliers to enhance visibility.

With the reducing sunlight, the camera images got darker and the noise increased. As a result, the temporal correspondences became more and more sparse. This effect can be
observed well in the plot of inlier correspondences, which increases slower. It is important to note, that the plot of the number of inliers in Figure 10 has been scaled to multiples of 10 inliers to enhance visibility. On average, the algorithm accumulates a single inlier per frame. Because of the slower growth of the data base, the back-projection error of the Bird’s Eye View declines slower but still reaches a value below 50 mm in the close area after 100 s.

In this experiment the algorithm exploits its design of batch processing accumulated correspondences. The first few seconds of the experiments show, that optimization would not be as accurate if it was limited to the small data set of correspondences from a single frame pair.

4.3 Violations of the flat world assumption

A central assumption of the camera orientation estimation in this work is a flat world plane, which is visible to the camera. This assumption might be violated in different situations such as elevated objects (e.g. cars, traffic signs), a sidewalk or a wrong known camera translation with respect to the vehicle.

Only points which move according to the vehicle model in a plane, which is similar to the road plane but maybe rotated around the camera, would support a specific orientation model of the camera during RANSAC optimization. Depending on the inlier threshold, points which are located outside the road plane would not support one single model or maybe no model at all. Thus scenes containing non-flat objects or a wrong known camera translation would be detectable by an optimization result featuring low support. In this case the system had to repeat calibration until it finds a model with sufficiently large support.

Based on this theoretical analysis, the system is expected to be robust against violations of the flat world assumption. A detailed, experimental analysis will be subject of future work.

4.4 Resource requirements

The memory requirement of the correspondence accumulation scales linearly in the number of correspondences. Since the correspondences do not have to be dense, the time consumption of the extraction is low. The RANSAC optimization of Equation (2) has a time consumption which scales linearly in the number of correspondences and cubic in the sample distance of the rotation parameter space. It’s memory consumption is constant. In the application of a Bird’s Eye View system both tasks can be executed in the background whenever the main task of calculating the Bird’s Eye View leaves unused resources.

5 Conclusion

End of assembly line calibration of vehicle mounted cameras for Bird’s Eye View applications is a costly process, which does not cover changes of camera orientation during lifetime. Inaccurate camera parameters distort the Bird’s Eye View image and could mislead the driver while maneuvering.

This paper presents a simple and robust solution to estimate camera orientation, which is the most sensitive parameter for construction of Bird’s Eye Views. The algorithm does neither require any artificial calibration objects, which increase costs in the service garage, nor does it depend on special road features like lane boundaries. It performs well in general maneuvering environments like parking lots or the side of a street. The approach also creates accurate results when scenery conditions degrade. It does not suffer from sparse correspondences even if this situation is persistent. Under low illumination conditions accumulation time increases but accuracy remains stable.

The approach is especially well suited for multi-camera systems, where cameras feature non-overlapping field of views. Since every camera can be calibrated against the vehicle coordinate frame separately, no correspondences have to be established between images of different cameras.

The algorithm showed to be well suited for the use in Bird’s Eye View maneuvering assistant. With its abilities, it can cope with mechanically moved camera mountings like retractable mirrors and is able to compensate minor damages, which slightly turned the camera out of its designed orientation. By providing the driver an accurate Bird’s Eye View it improves maneuvering safety.

Future work will expand the presented extrinsic autocalibration by estimating translation as well. This would enable the usage of this approach in applications, were there is no previous knowledge about the camera’s translation against the vehicle frame. A more elaborate ego motion model, which includes special vehicle dynamics during maneuvering, could enable the system to exploit correspondences which were accumulated at large steering angles as well.

References


Facade Segmentation from Streetside Images

Andreas Wendel and Horst Bischof
Institute for Computer Graphics and Vision (ICG), Graz University of Technology
{wendel, bischof}@icg.tugraz.at

Abstract Procedural modeling of facades has become a topic of high interest in computer graphics and vision. It allows to generate high-quality, three-dimensional (3D) city models for usage in geospatial mapping platforms such as Microsoft Bing Maps. However, state-of-the-art approaches for 3D modeling require orthorectified images of a single facade as input. Since streetside data is in general acquired by cameras on top of a moving car, images are not rectified and may show multiple facades. Within this paper, we introduce a novel approach for separating and segmenting individual facades. Our algorithm incorporates prior knowledge about arbitrarily shaped repetitive regions, which are detected using intensity profile descriptors and a voting-based matcher. In the experiments we compare our approach to extended state-of-the-art matching approaches using more than 600 challenging streetside images, including different building styles and various occlusions. Our algorithm outperforms these approaches and allows to correctly separate 94% of the facades. Pixel-wise comparison to our ground-truth yields a segmentation accuracy of 85%. According to these results our work is an important contribution to fully automatic building reconstruction.

1 Introduction

Large-scale image acquisition for geospatial mapping platforms such as Microsoft Bing Maps or Google Maps requires appropriate data processing methods. While early systems just showed raw photographies, more recent visualization techniques allow to superimpose data obtained from satellite imagery, aerial photography, and road maps. Modern systems use multiple-view images to analyze the geometric properties of objects, resulting in three-dimensional (3D) visualizations.

State-of-the-art methods [21] for 3D reconstruction are able to automatically extract simple building models from aerial images. However, photographs taken from an airplane offer limited view of building facades and therefore the reconstruction lacks details. This is depicted to the left in Figure 1 using a 3D model obtained from Microsoft Bing Maps. In contrast, an image of the same location taken from street-level can be found to the right. The advantages are obvious: Streetside images, as we call such images throughout our work, can be obtained with much higher spatial resolution and from a more natural point of view than aerial images.

Recently, there has been increasing interest in using streetside imagery for automatically deriving 3D building models [14, 9]. Müller et al. [14] introduced an approach on image-based procedural modeling. Given single facade images as input they first determine the structure of a facade. Once they know about the hierarchical subdivisions of the facade it is possible to replace architectural elements by parameterizable models. This representation has several advantages: First, the visual quality of the image is improved. Whether we scale or change the view within the model, there is no limit in spatial resolution anymore. Second, the approach assigns semantical meanings to the parts of the facade. This is important for future applications using city models, for instance if the entrance has to be located. Third, the huge amount of image data required to visualize an entire city can be reduced to a predictable number of parameters. In other words, this technology is required by geospatial mapping systems for providing high-quality data because it would not be possible to transmit the textured 3D models in a reasonable time across the web.

The availability of procedural modeling algorithms motivates the implementation of fully automatic streetside modeling pipelines. However, the gap between real-world data and assumptions in algorithms is big: State-of-the-art approaches for 3D modeling [14, 9] require orthorectified images of a single facade as input. Since streetside data is in general acquired by cameras on top of a moving car, images are not rectified and may show multiple facades. The goal of our work is to close this gap by detecting and extracting single facade segments from streetside images. By our definition, two facades should be separated if a significant change in color or building structure can be detected. A single facade segment is therefore a coherent area in an image, containing repetitive patterns which match in color and
texture. An optimal result is depicted in Figure 2. The algorithm should be invariant to different architectural styles, occlusions, and illumination variations.

Figure 2: The goal of our work is to detect and extract single facade segments automatically, as depicted in this panoramic image with four manually labeled facades.

The successful realization of this task does not only have an impact on automatic procedural modeling workflows but also supports other computer vision algorithms that cope with urban environments. For example, window detection is strongly simplified if applied to single facades because the appearance of windows is often similar. Our work contributes to this goal in two areas:

1. **Repetitive patterns.** We analyze repetitive patterns by using contextual information rather than directly comparing features or raw image data. Besides, we compare our algorithm against various state-of-the-art feature matching approaches which we adapt for our purpose.

2. **Facade separation and segmentation.** Building upon repetitive patterns discovered in streetside images, we show how to separate and segment facades. The approach is evaluated using 620 high-resolution streetside photographs, acquired by cameras on top of a moving car. The images offer total coverage of a city as seen from roads, but also include difficulties such as various building styles and occlusions.

The remainder of this work is organized as follows. Section 2 gives a short overview of the streetside dataset and Section 3 provides information about related work. Our approach to facade segmentation is explained in detail in Section 4, and results of the evaluation are presented in Section 5. Finally, a summary and ideas for future work are given in Section 6.

## 2 Streetside Data

We apply our algorithms to streams of high-resolution streetside photographs. We are able to use images of cameras with different orientations, taken from a car to both sides in different cities. Consecutive images are guaranteed to overlap, leading to full coverage of the scene. Processing the streetside dataset is very challenging, as algorithms have to cope with different architectural styles, occlusions, and illumination variations. An example for such an image stream can be seen in Figure 3. As long as the facade plane is parallel to the image acquisition plane, we can assume that buildings in streetside images are separated by vertical structures such as edges, gutters, or various types of decorations. Buildings with common architecture allow to find repetitive structures in horizontal and sometimes vertical direction within a single facade. Those repetitive elements are most likely windows, but could also be ornaments or decorations.

Our evaluation is based on 620 images in total, which can be grouped in two single-frame datasets and one dataset with stitched images (panoramic images). Panoramic stitching is important because it increases the field of view in the direction necessary for finding repetitive patterns. However, common algorithms such as Autostitch [2] (uses SIFT descriptors [12], RANSAC [6] matching, and bundle adjustment) regularly fail due to occlusions by vegetation and significant depth changes. Therefore, we cannot rely on panoramic images in automatic workflows. To show the applicability of our approach, we manually selected 20 series (each of about 30 consecutive frames) where Autostitch worked. The stitching process resulted in the Panorama dataset (various cities, 20 images). The difficulty is to find multiple separations and segmentations despite the different facade structures.

For single-frame images, we roughly correct the perspective distortion of the up-looking cameras using a constant transformation (which is estimated in the first frame). While our results show vertically stitched images, this is not necessary for our algorithm to work and is just done for visualization purposes. We distinguish two datasets: First, the NYC dataset (New York City, USA, 220 frames) contains images of the up-looking camera which have been transformed to correct perspective distortions. While we do not encounter any problems with occlusions and clutter here, the main difficulties of the dataset are very similar facades and low image quality in the top third of the image. The second dataset called Graz (Graz, Austria, 380 frames) consists of images from the up- and the side-looking cameras, which enlarges the visible part of the facade (buildings are much lower than in New York City). However, this introduces some problems: Cars and trees occlude the facade and large parts of road and sky can be seen. Furthermore, facades are heavily structured and it is hard to obtain good segments for most of them.

## 3 Related Work

Facade separation and segmentation are tasks which have not received much attention in the past. Classical segmentation approaches try to group pixels with similar visual properties, whereas our approach is to find repetitive regions and combine those into a diverse facade segment. On the one hand the huge amount of data (several terabytes per city) requires algorithms to be fast, but on the other hand the variety of scenes and facades expresses the need for robustness.
3.1 Facade separation
Müller et al. [14] introduced an algorithm which is able to automatically subdivide images of single facades into floors and tiles. After removing vertical and horizontal symmetries they obtain the Irreducible Facade, which summarizes redundant parts of the facade. Clearly, such a representation would help to separate individual facades. As already mentioned, a major limitation is the dependency on single facade images. Additionally, automatic processing fails for scenarios with blurry texture, low contrast, chaotic ground floors, and occlusions caused by vegetation, hence the approach is not useful for our task.

3.2 Facade segmentation
The easiest way to address our problem would be to use general-purpose approaches to segmentation. Popular methods include Normalized Cuts [18] and Mean Shift [3]. We also tried two more recently published approaches, namely efficient graph-based segmentation [5] and ROI-SEG [4]. However, the results in Figure 4 do not satisfy our needs.

Most application-driven approaches therefore incorporate some kind of prior knowledge. Korah and Rasmussen [11] assume that the pixels located around a window belong to the building wall. Given a color Gaussian Mixture Model of the entire image, the cluster which fits the distribution of near-window pixels best is selected as facade plane. This method would fail for our datasets, as we cannot assume knowledge about the window grid and a significant part of facades does not have a homogeneous texture.

In this work we present an idea to incorporate prior knowledge about the repetitive areas of an image into the segmentation process. Felzenszwalb and Huttenlocher [5] introduced an efficient graph-based segmentation method, which can be extended to suit our needs. It neglects detail in high-variability image regions while preserving it if the region is pretty much homogeneous. The algorithm has already been used successfully in other works [10] for creating superpixels and proved to be fast, having a complexity linear to the number of image pixels.

3.3 Finding repetitive patterns
As we have seen before we need to acquire knowledge about repetitive patterns, which can be done by matching of local features. This has been widely investigated but hardly ever applied to a single image. Most algorithms were originally developed for object recognition or wide-baseline matching where the task is to find the single best match to a descriptor in a second image. We demonstrate how such approaches can be extended for finding repetitive patterns by taking into account non-perfect matches as well.

The simplest approach we can think of is to extract fixed-size raw patches of pixels, grayscale or in any color space, and match them within the image. Clearly, the computation time for exhaustive matching is very high and the invariance to rotation, scaling, illumination, and occlusions close to zero. However, we can use interest points such as Harris corners [8] to speed up the process.

Shechtman and Irani [17] presented an approach to match complex visual data using local self-similarities. They correlate a patch centered at the point of interest with a larger surrounding region and use the maximal correlation values within log-polar bins as descriptor. Features are computed densely on a 5 pixel fixed grid. The benefit of this approach is its speed and the independence of representation, meaning that just the spatial layout or shape is important. However, this poses a problem for our needs: The most common repetitive patterns are windows and their shape is often similar in different facades. While the texture within the window often stays the same, the texture outside changes and should definitely influence the result.

The most popular method for local feature-based matching in computer vision applications is Lowe’s Scale-Invariant Feature Transform (SIFT) [12]. While other approaches with similar properties have been proposed (see [15] for an overview), we chose SIFT to represent this category of methods. It has been widely used and several evaluations show that it is one of the best descriptors currently available [13]. SIFT has been designed for finding correspondences among local feature sets from different images. For single image operation a range of valid matches needs to be defined. However, finding a proper threshold turned out to be difficult as the descriptor either matched with structures across facade boundaries or it did not find enough matches within a facade.

Tell and Carlsson [19, 20] developed a robust approach for wide-baseline matching. The basic idea is to extract intensity profiles between pairs of interest points and match them to each other. If these profiles lie on a locally planar surface such as a facade, any scale-invariant descriptor is also invariant to affine transformations. Tell and Carlsson compute six Fourier coefficients as features, namely the dot product of the profile with the first three sine and cosine basis vectors. Descriptors of the first image are then matched to descriptors of the second image, and votes are casted for the respective start- and endpoints of the matching intensity profiles. Maxima in the voting table are then considered as the best matches for the interest points of both images.

Within this work, we adapted the approach of Tell and Carlsson for finding repetitive patterns within a single image. Given locally planar facades and accurate endpoints (Harris corners) the approach is invariant to affine transformations.
4 Facade Segmentation Algorithm

Our algorithm for facade segmentation consists of three major steps: First, we detect repetitive patterns in streetside images by extending a method designed for wide–baseline matching. The resulting pairs of interest points are then used in a bottom–up manner to separate facades. Finally, we combine the knowledge about repetitive areas with state–of–the–art segmentation methods to obtain individual facade segments. These steps are described in detail in the following paragraphs. A visualization can be found in Figure 6 and 7.

4.1 Finding repetitive patterns

Our approach to repetition detection is similar to Tell and Carlsson’s work on wide–baseline matching [19, 20]. First we detect interest points in the image, namely Harris corners [8]. Instead of computing descriptors at these locations, we form pairs of interest points and extract the image content on a straight line between them. This results in a graph which connects all interest points (nodes) to each other. To limit the complexity, we reduce the degree of every node to 30 by taking only the closest neighbors into account.

The next step is to describe the image content in a compact way, while remaining invariant to scale and small illumination changes. We smooth the image and then extract color intensity profiles on a straight line between pairs of Harris corners. Every RGB color channel (red, green, blue) contributes 20 values to the descriptor, sampled using bilinear interpolation in regular intervals along the line. Finally, the 60–dimensional descriptor is normalized. We achieve scale invariance by extracting a fixed amount of coefficients from interpolated, one–dimensional data. While illumination changes of small areas are also corrected by the interpolation, larger areas can be handled with the tolerance of the matching approach. Figure 5 visualizes some extracted intensity profiles and their origin in the image.

Tell and Carlsson [19] propose the application of a kd–tree [7] for efficient matching of descriptors from one image to descriptors of another image. We apply the same technique to find matching descriptors within an image. However, in our case the best match would be the descriptor itself, so we tolerate ±5% deviation off the descriptor prototype for finding repetitive patterns. We do not consider matches with more than 10 descriptors involved because these features are not discriminative enough.

The robustness of the approach is based on a voting step following the matching of intensity profiles. All profiles within the specified range vote for the similarity of the respective pair of start- and endpoints of the profiles. Using this method, interest points which are actually in similar regions get more votes than two random points. We store a list of contributing profiles for every possible pair and increase the vote count only if a descriptor has not contributed to that correspondence so far. This is in a similar manner described in [20] and ensures that no bias is introduced in the voting matrix.

Once the matching process is finished and votes have been casted, we need to locate repetitive patterns in the vote matrix. While Tell and Carlsson search maxima in rows and columns to find the single best match, we allow multiple matches for every interest point. We threshold the number of votes a correspondence received. The minimal number has been found empirically and is set to 3 of 30 profiles (10%), i.e. the correspondence of interest points is supported by at least three intensity profiles. One of the advantages of this descriptor is the ability to match arbitrary areas of the image, as visualized in Figure 6(a).

For the purpose of our work, we can further restrain the previously found matches. Repetitive patterns on facades are unlikely to occur across the entire image, but also very close matches are not valuable. We therefore restrict the horizontal or vertical distance of the matches to avoid outliers. The final result of our algorithm for finding repetitive patterns in a single image can be seen in Figure 6(b). Note that only a small amount of interest point correspondences cross the boundary between the two facade segments, while we can find a large number within a segment.

4.2 Facade separation

The majority of images in streetside datasets show facades where repetitions occur in horizontal direction and separations between facades in vertical directions. Our algorithm assumes that facade separations can be approximated by a vertical line. Clearly, this approximation does not hold for perspective views, so roughly correcting this distortion is the only preprocessing step necessary (cp. Section 2).

The first stage of our facade segmentation algorithm provides interest point correspondences. We can now use these results to detect clusters of repetitive patterns. To obtain a repetition likelihood, we project the lines between all pairs of matching interest points onto the horizontal axis and obtain the match count for every position on the horizontal axis.
Facade Segmentation from Streetside Images

Figure 6: From streetside data to separation: (a) Arbitrary areas of the image can be matched to each other. (b) Repetitive patterns (color-coded lines) occur frequently within facade segments, but hardly connect them. (c) The connection line between matching interest points is projected onto the horizontal axis. Summing up the projections results in a match count for every position on the horizontal axis. (d) The repetition likelihood (normalized match count) can be thresholded using the uniform repetition likelihood. (e) As a result, separation areas mark the position where one facade ends and another starts (green). Unknown areas occur close to the image boundary if not enough matches are available (red), and repetitive areas in between are used for facade segmentation.

This step is illustrated in Figure 6(c). Vertical matches are summed up as well to support the overall match count in regions where not enough horizontal matches are available. However, to avoid false separations by the resulting sharp peaks we filter the projected matches using a median filter. Finally, the match count is normalized by the total number of matches to obtain a repetition likelihood. It shows the percentage of all matches at a given place on the horizontal axis and is a more general measure of repetitiveness.

Simply detecting minima on the repetition likelihood is not suitable for finding separation areas, as the global minimum would fail for panoramic images with multiple splits and local minima occur regularly between rows of windows. If all parts of the facade would contribute the same amount of repetitive patterns to the likelihood, we would get a uniform repetition likelihood. This value is an intuitive threshold, because areas where the likelihood is higher are more repetitive than average (repetitive areas) and areas where it is lower are less repetitive (separation areas). This fact is visualized in Figure 6(d). While repetitive areas are used for segmentation later on, separation areas mark the position where one facade ends and another starts. We also have to cope with the problem of very narrow fields of view, meaning that few or no repetitions are visible in images that actually show a separation. We solve this by defining an unknown area on both sides of the image, starting at the image boundary and ending at the point where the first repetitive area is detected. Separation examples for Graz are given in Figure 6(e). The algorithm does not depend heavily on the quality of repetitive patterns but works well with several approaches, as the evaluation in Section 5.3 shows.

4.3 Facade segmentation

For the task of facade segmentation, it is important to consider the continuity of segments and the underlying structure of facades. While most segmentation algorithms just analyze the image locally, we want to incorporate repetitive areas as prior knowledge. Based upon the results of the separation stage, we process repetitive areas individually. Based upon the results of the separation stage, we process repetitive areas individually. An interest point is called repetitive point if it has some correspondence to another point. We assemble a set of all repetitive points within a repetitive area and compute a convex hull \[1\] for this set of points, as can be seen in Figure 7(a). The graph-based segmentation approach of \[5\] can be adjusted to deliver oversegmentations, often called superpixels: Relatively small, homogeneously colored segments, visualized in pseudo–colors in Figure 7(b). We further combine prior knowledge and superpixels to the binary masks in Figure 7(c). For all repetitive areas, we include those superpixels into the respective facade segment which overlap with the convex mask. Although the raw segmentation is already quite good, it can be improved by using morphological operations. We morphologically close gaps in the segmentation, then fill holes in the mask, and finally morphologically open the mask to cut elongated facade parts such as cornices. The final output of our algorithm are individually segmented facades. Figure 7(d) shows an example of the Graz dataset. We chose the data–driven visualization with arbitrary shapes for our application and evaluation, but it is also possible to further refine the segments to rectangular shapes by median–filtering the boundary.

5 Experiments and Results

Experimental evaluation focuses on analyzing the influence of different profile descriptors and of different methods for point matching on the quality of facade segmentation. We apply the algorithms to the three datasets introduced in Section 2, with images scaled to a height of 1180 pixels. Our approach also depends on the parameters of the Harris cor-
Figure 7: From separation to segmentation: (a) The convex hull spans around all repetitive points in the repetitive area. The resulting convex binary mask, visualized in red, is used to introduce prior knowledge to the segmentation process. (b) Felzenszwalb’s algorithm [5] delivers relatively small, homogeneously colored segments (superpixels). (c) The prior knowledge in form of a convex mask can be combined with appearance–based segmentation. (d) Typical results for facade segmentation after morphological postprocessing.

Table 1: The evaluation of profile descriptors on all datasets shows that intensity profiles in RGB color space perform best.

<table>
<thead>
<tr>
<th>Profile Descriptor</th>
<th>620 images, 3 datasets</th>
<th>$PR_{\text{match}}$</th>
<th>$F_{1,\text{sep}}$</th>
<th>$F_{1,\text{seg}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intensity (RGB)</td>
<td>72.8%</td>
<td>94.0%</td>
<td>85.4%</td>
<td></td>
</tr>
<tr>
<td>Intensity (Lab)</td>
<td>68.8%</td>
<td>89.6%</td>
<td>83.8%</td>
<td></td>
</tr>
<tr>
<td>Intensity (gray)</td>
<td>67.2%</td>
<td>90.9%</td>
<td>83.6%</td>
<td></td>
</tr>
<tr>
<td>Fourier (gray)</td>
<td>67.1%</td>
<td>89.8%</td>
<td>84.6%</td>
<td></td>
</tr>
<tr>
<td>Gradient (gray)</td>
<td>51.0%</td>
<td>82.6%</td>
<td>80.3%</td>
<td></td>
</tr>
<tr>
<td>Histogram (RGB)</td>
<td>51.8%</td>
<td>83.6%</td>
<td>81.3%</td>
<td></td>
</tr>
</tbody>
</table>
segmentation algorithm and differ only by the approach to repetitive pattern detection. We impose the same restrictions on distance, horizontal, and vertical offset (cp. Section 4.1) for all approaches. For fast matching we approximate the Euclidean distance by the ratio of angles between two descriptors. Table 2 visualizes the results in terms of matching precision, separation effectiveness, and segmentation effectiveness on the combined datasets.

According to these experimental results our method performs better than any extended state–of–the–art method. Although the distance to its competitors is small there are two main reasons for using our method: First, it achieves the best matching precision combined with a number of correct repetitive patterns which is three times higher than for other methods. This increased support makes the repetition likelihood defined in Section 4.2 more robust. Second, our method is most tolerant regarding the appearance of repetitive patterns. Corresponding areas can be arbitrarily shaped and do not depend on scale or rotation.

The reference algorithms we extended for our purpose achieve good results as well. The most simplistic method, raw patches of pixels, performs better than SIFT or self–similarities for most datasets. The weak point though is the separation of facades in panoramic images, which is more than 10% less effective than our approach. The main reason is the missing scale invariance caused by extracting fixed–size patches. The description strength of the SIFT approach limits its applicability for finding repetitive patterns. We optimized the matching threshold for good precision, but this results in a low number of detected point correspondences. If we increase the threshold to get more matches, matching precision quickly deteriorates. The self–similarity descriptor is worst when evaluated using all datasets, but it shows good performance for the NYC dataset. The reason for this behavior is the large amount of sky and road areas in the Graz and Panorama datasets. Other approaches are not effected by those self–similar areas because they use interest points, but the self–similarity approach uses a fixed grid for descriptor extraction.

### 5.4 Illustration of results

After comparing the performance of various approaches using objective measures we want to summarize and illustrate the results. When applying the facade segmentation algorithm to three datasets containing 620 images, we achieve on average a matching precision of 72.8%, a separation effectiveness of 94.0%, and a segmentation effectiveness of 85.4%. Figure 8 and 9 show the separation and segmentation of four consecutive images for the Graz and NYC datasets, respectively. For better visualization, videos are available online. Second, the Panorama dataset contains 20 panoramic images with 3–5 facades each. A typical result for separation and segmentation can be found in Figure 10.

![Figure 8](image1.png)  
**Figure 8:** Typical results for facade separation (top) and segmentation (bottom) in the Graz dataset. The separation ground–truth is marked by red lines. It overlaps either with separation areas (green), or with unknown areas (red) if not enough repetitive matches can be found.  

![Figure 9](image2.png)  
**Figure 9:** Typical results for facade separation (top) and segmentation (bottom) in the NYC dataset.

Separation problems occur if the field of view is too small. In such a case, a different column of windows is enough to indicate a different facade. Segmentation problems are mainly caused by repetitive patterns which are not part of the facade, such as power lines in the sky. The resulting segment is therefore too large and includes parts of the sky. Missing repetitive patterns at image boundaries lead to wrong segmentations as well.

---

1. [http://www.icg.tugraz.at/Members/wendel](http://www.icg.tugraz.at/Members/wendel) (Jan 19, 2010)
6 Conclusion

This work is concerned with the fully automatic segmentation of facades from streetside images. We propose an algorithm which closes the gap between real-world data and state-of-the-art procedural modeling approaches [14, 9].

Our contributions are two-fold: First, we developed a novel approach for finding pairs of interest points within a single image. We compare contextual information using pairwise intensity profile descriptors and an intermediate step of vote casting. As a result, corresponding areas can be arbitrarily shaped and the matches are invariant to small illumination changes and affine transformations. Second, we present a novel approach for facade separation and segmentation. Our algorithm is evaluated using 620 high-resolution streetside photographs, which offer extensive coverage of a city as seen from roads. We achieve excellent results of 94.0% separation and 85.4% segmentation effectiveness overall, making our work an important contribution to fully automatic building reconstruction.

Future research should focus on detecting occlusions such as vegetation and cars, as avoiding separations in these areas would improve the performance. It would also allow to take advantage of the redundancy. Our results could also be incorporated by other computer vision algorithms, as knowledge about individual facades simplifies window detection, and matching of arbitrarily-shaped repetitive areas could be applied to texture segmentation and symmetry detection. Finally, Figure 11 shows the results of our algorithm on different input data. The matching and segmentation results are good, thus encouraging further work on this topic.

Acknowledgement

This research has been supported by the Austrian Research Promotion Agency (FFG) project FIT-IT CityFit (815971/14472-GLE/ROD).

References

Geodesic Adaptive Support Weight Approach For Local Stereo Matching

Asmaa Hosni, Michael Bleyer, Margrit Gelautz, and Christoph Rhemann

Institute for Software Technology and Interactive Systems, Vienna University of Technology
 Favoritenstr. 9-11/188/2, A-1040 Vienna, Austria
 [asmaa, bleyer, gelautz, rhemann]@ims.tuwien.ac.at

Abstract
Local stereo matching has recently experienced large progress by the introduction of new support aggregation schemes. These approaches estimate a pixel’s support region via color segmentation. Our contribution lies in an improved method for accomplishing this segmentation. Inside a square support window, we compute the geodesic distance from all pixels to the window’s center pixel. Pixels of low geodesic distance are given high support weights and therefore large influence in the matching process. In contrast to previous work, we enforce connectivity by using the geodesic distance transform. For obtaining a high support weight, a pixel must have a path to the center point along which the color does not change significantly. This connectivity property leads to improved segmentation results and consequently to improved disparity maps. The success of our geodesic approach is demonstrated on the Middlebury images. According to the Middlebury benchmark, the proposed algorithm is the top performer among local stereo methods at the current state-of-the-art.

1 Introduction

Local stereo matching algorithms center a support window on a pixel of the reference image. This window is then displaced in the second view along the corresponding epipolar line in order to find a matching point of maximum correspondence.

The major challenge in local stereo is to find a well-suited size for the typically square support window. A window should thereby be large enough to capture sufficient intensity variation for handling regions of poor texture. At the same time, the window should be small enough to not include pixels of different disparities in order to avoid the well-known edge fattening effect at disparity discontinuities. In practice, there is no golden middle between these conflicting requirements. This has led to a long track of research that varies window sizes and shapes depending on the image position (e.g., [5]). However, the moderate results of these early approaches are in general not able to compete with state-of-the-art algorithms that typically rely on global optimization [8].

Local stereo matching has recently experienced a renaissance with the upcome of novel segmentation-based support aggregation schemes. These methods can deliver results close to the quality of global approaches. They assume that pixels of homogeneous color share the same disparity value. Hence, it is reasonable to find a pixel’s support region via the use of color segmentation. In the original adaptive weight approach [11], weights are computed for all pixels within a square window. These weights regulate a pixel’s influence in the matching process. A pixel’s weight is thereby inversely proportional (i) to its color dissimilarity to the window’s center pixel and (ii) to its spatial distance from the center.

Alternative methods for computing these weights have been proposed recently. The work of [2, 9] suggests using a precomputed (mean shift-based) color segmentation in the weight calculation. While this approach seems to work slightly better, it takes away some elegance by “outsourcing” the segmentation into a preprocessing step that also consumes additional processing time. In the context of prior work, also the approach of [6] is mentioned. Here, weight computation is accomplished via energy minimization. For more details and evaluation of related aggregation methods, the reader is referred to two recent studies [3, 10].

2 Algorithm

Our proposed algorithm is summarized in the pipeline shown in figure 1. Details on the processing steps are given in the following subsections.

2.1 Local Matching with Geodesic Support

The basic idea of this paper is to compute the support weights within a square window via the use of geodesic distances. This is explained as follows. The geodesic distance \( D(p, c) \) between a pixel \( p \) of the support window and the window’s center \( c \) is defined as the shortest path that connects \( p \) with \( c \) in the color volume:

\[
D(p, c) = \min_{P \in P_{p,c}} d(P). \tag{1}
\]

Here, \( P_{p,c} \) denotes the set of all paths between \( p \) and \( c \). A path \( P \) is defined as a sequence of spatially neighboring points in 8-connectivity \( \{p_1, p_2, \ldots, p_n\} \). The costs \( d() \) of a path are computed by

\[
d(P) = \sum_{i=2}^{n} d_C(p_i, p_{i-1}). \tag{2}
\]
with $d_c()$ being a function that determines the color difference. This function is implemented by

$$d_c(p, q) = \sqrt{(r_p - r_q)^2 + (g_p - g_q)^2 + (b_p - b_q)^2}$$  (3)

where $r_p$, $g_p$, and $b_p$ are the values of $p$’s RGB channels.

Intuitively spoken, the geodesic distance from pixel $p$ to the window center $c$ is low, if there exists a path between these points along which the color varies only slightly. We refer to this property as connectivity.

We assume that pixels of low geodesic distance lie on the same disparity as $c$ itself. Low geodesic distances should therefore convert into high support weights. The function $w()$ implements this by

$$w(p, c) = \exp\left(-\frac{D(p, c)}{\gamma}\right)$$  (4)

with $\gamma$ being a user-defined parameter that defines the strength of the resulting segmentation.

Once the support weights are known, they are exploited to aggregate pixel dissimilarities within the support window. The aggregated matching costs for a pixel $c$ at disparity $d$ are derived by

$$m(c, d) = \sum_{p \in W_c} w(p, c) \cdot f(p, p - d)$$  (5)

with $W_c$ representing all pixels of the square support window centered on pixel $c$. The window size is thereby given by the user. The function $f(p, q)$ computes the color dissimilarity between a pixel $p$ of the reference image and a pixel $q$ of the match view. In principle, one can use equation (3) to implement $f()$. However, we have chosen Mutual Information [4] in order to handle illumination differences that may exist between the two input images. In addition, we put an upper bound $f_{\text{max}}$ on the pixel dissimilarity so that values exceeding $f_{\text{max}}$ are truncated. This serves to reduce the influence of occluded pixels [7].

Finally, we obtain a disparity map by determining the disparity $d_p$ of each a pixel $p$ of the reference view using local optimization:

$$d_p = \arg\min_{d \in D} m(p, d)$$  (6)

where $D$ represents the set of all allowed disparities. In fact, there is nothing that speaks against replacing this Winner-Takes-All strategy with a global optimization technique such as graph-cuts or belief propagation. This would most likely further improve the quality of results.

### 2.2 Geodesic versus Original Adaptive Weights

Let us use the support weight examples of figure 2 to illustrate the advantage of our geodesic approach over the method of [11]. Figure 2a shows an artificial image in which a foreground object is placed in front of a background object of similar color. For computing the support weights, the original adaptive weight method just considers the color difference and the distance to the center pixel. In our example, it therefore erroneously gives high weights to background pixels. In contrast to this, our approach can successfully handle this example by considering the whole structure of the image patch. Recall that our geodesic computation requires that there is a path of approximately constant color to the center pixel. For background pixels, such a constant color path does not exist due to the edge that separates the foreground from the background object. Figures 2b and 2c show this effect on real-world images. Figures 2b thereby uses the same image patches as in the paper of [11].

It is important to notice that the computational efficiency of our geodesic support weight strategy is practically the same as that of [11]. For both methods, the computation of a single weight mask has a complexity of $O(|W|)$ with $W$ being the set of all pixels inside the square window. (We describe an efficient algorithm for approximating the geodesic weight mask in section 2.3). However, both methods share the performance bottleneck that originates from evaluating the window at each disparity. This pixel comparison operation has complexity $O(|W| \cdot |D|)$, with $D$ denoting the set of allowed disparities. Due to using adaptive weights, this operation cannot easily be speeded up using sliding window techniques from which local methods commonly take their high efficiency. Nevertheless, it has been shown [3] that by GPU programming a real-time implementation of [11] can be accomplished. This would most likely also work for our method. In our current implementation, it takes approximately a minute to compute the disparity map for standard test image pairs.

### 2.3 Approximation of Geodesic Distances

We apply the method of [1] for efficiently approximating the geodesic distance of each pixel within the window to the center pixel (equation (1)). This method is reviewed as follows.

Each pixel $p$ inside the window is assigned to cost $C(p)$. Initially, the costs of the center pixels are set to 0, while the costs of all other pixels are set to a large constant value. The algorithm invokes two passes. In the first pass (forward pass) of the algorithm, we traverse the support window in row major order (see figure 3a). The cost of a pixel $p$ is thereby updated by

$$C(p) := \min_{q \in K_p} C(q) + d_c(p, q)$$  (7)

with the kernel $K_p$, being a set of pixels consisting of $p$ itself as well as its left, left upper, upper and right upper neighbors (Grey pixels of figure 3a). The cost update is thereby performed immediately so that the new costs already affect the cost computation of the next pixel. Once the forward pass is completed, we invoke the backward pass. This pass traverses the window in reverse direction (figure 3b). It thereby...
updates the costs using equation (7) in conjunction with the kernel $K_p$ of figure 3b. Forward and backward passes are iterated. (In our experiments, we found 3 iterations to be sufficient for giving reasonable results.) The final costs $C(p)$ represent our estimate of the geodesic distance of $p$ to the center pixel.

2.4 Occlusion Detection and Filling

Up to this point we have ignored the occlusion problem. In order to detect occlusions, we follow common practice in local stereo by applying left right consistency checking. We therefore use the algorithm of section 2.1 to compute a first disparity map with the left image chosen as reference frame. A second disparity map is derived by choosing the right image as reference. For each pixel of the left disparity map, we then check whether it carries the same disparity assignment as its matching point in the right disparity map. If this is not the case, the pixel is invalidated. This left right check is effective in filtering out occluded pixels as well as mismatches.

In the occlusion filling step, we assign invalidated pixels to new disparity values. For each invalidated pixel $p$, we estimate $p$’s first valid neighbour to the left $l$ and to the right $r$. The disparity $d_p$ is then computed by $\min(d_l, d_r)$. This simple filling strategy typically generates horizontal streaks. To eliminate these artifacts, we apply a smoothening filter on the invalidated pixels. We thereby use a weighted median filter with the filter weights obtained from the precomputed geodesic weight masks of section 2.1. In comparison to standard median filtering, our weighted median filter does not suffer from the problem of distorting object boundaries.

3 Results

We have used the Middlebury stereo benchmark [8] to evaluate the performance of our approach. In our test run, the algorithm’s parameters are set to the constant values of $\gamma := 10$ and $f_{\max} := 120$. The window size is chosen to be 31. These parameters have been found empirically. We plot our results on the Middlebury images along with corresponding error maps in figure 4. One can see that our algorithm performs well in the reconstruction of disparity borders, while it also finds correct disparities for regions of low texture. It is traditionally difficult for a local method to fulfill these two requirements at the same time.

Table 1 shows quantitative results that are taken from the Middlebury online table. Our algorithm currently takes the 10th rank out of 74 submissions. This is specifically promising when considering that we do not use time-consuming global optimization. This is in contrast to all better performing algorithms. According to the Middlebury table, our method is the currently best performing local method. It can outperform the original adaptive weight approach [11] relatively clearly.

More results from a new set of Middlebury test images are presented in figure 5. The disparity maps computed by our algorithm are shown in figure 5c and the corresponding measured error in figure 5d. Only disparity errors larger than one pixel are displayed in figure 5d.

Figure 6 shows the performance of our algorithm (figure 6a) and the referenced algorithm [11] (original adaptive weight method) (figure 6b) with varying window sizes. The diagrams give the percentage of error measured in non-occluded regions of the four test images from the Middlebury benchmark. By comparing figures 6a and 6b we see that our algorithm in figure 6a is more robust against changing window sizes than the reference algorithm in figure 6b. This is mainly due to the use of geodesic weights in the support window. The connectivity property that is inherent to our method tends to fix the weights after exceeding a specified window size. Accordingly, the results are nearly the same for windows larger than this size.

Also, we experimented with the effect of changing $\gamma$
Geodesic Adaptive Support Weight Approach For Local Stereo Matching

Figure 4: Results on Middlebury images generated using constant parameter settings. (a) Left image. (b) Ground truth. (c) Results computed by our algorithm. (d) Disparity errors larger than one pixel.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Rank</th>
<th>Avg. Error [%]</th>
<th>Error non-occluded pixels [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Tsukuba</td>
</tr>
<tr>
<td><strong>GeoSup</strong></td>
<td>10</td>
<td>5.80</td>
<td>1.45</td>
</tr>
<tr>
<td>AdaptDispCalib</td>
<td>13</td>
<td>6.10</td>
<td>1.19</td>
</tr>
<tr>
<td>DistinctSM</td>
<td>18</td>
<td>6.14</td>
<td>1.21</td>
</tr>
<tr>
<td>CostAggrOcc [6]</td>
<td>22</td>
<td>6.20</td>
<td>1.38</td>
</tr>
<tr>
<td>SegmentSup [9]</td>
<td>24</td>
<td>6.44</td>
<td>1.25</td>
</tr>
<tr>
<td>AdaptWeight [11]</td>
<td>29</td>
<td>6.67</td>
<td>1.38</td>
</tr>
<tr>
<td>SSD+MF [8]</td>
<td>70</td>
<td>15.7</td>
<td>5.23</td>
</tr>
</tbody>
</table>

Table 1: Rankings of selected local methods in the Middlebury online database. Our algorithm (denoted by GeoSup) is currently the overall best-performing local method.
Figure 5: Results on new Middlebury images generated using the same parameter settings as in figure 4. (a) Left image. (b) Ground truth. (c) Results computed by our algorithm. (d) Disparity errors larger than one pixel.

Figure 6: The effect of changing window sizes on four test images of the Middlebury benchmark. The error displayed is the error in non-occluded regions. (a) Results obtained by our algorithm. (b) Results obtained by [11].
Geodesic Adaptive Support Weight Approach For Local Stereo Matching

values (parameter used in converting geodesic distances to geodesic weights, see equation (4)) and plotted the results in figure 7. The measurements show that our algorithm is also robust against varying values of \( \gamma \) in our test images.

4 Conclusion

This paper has proposed a novel support aggregation approach for stereo matching. To derive support weights, we have computed geodesic distances for all pixels of the support window to the window’s center point. The advantage over previous work is that we implement the concept of connectivity that proves to be effective for obtaining improved segmentation results. We have tested our results using the Middlebury benchmark. According to the results, the proposed geodesic support weight approach is the top performer among stereo methods that rely on local optimization.

Acknowledgements

Asmaa Hosni is supported by Vienna PhD School of Informatics. Financial support has also been obtained from the Austrian Science Fund (FWF) under project P19797 and the Vienna Science and Technology Fund (WWTF) under project ICT08-019.

References

Irregular Laplacian Graph Pyramid

Dan Shao and Walter G. Kropatsch

Vienna University of Technology,
Pattern Recognition and Image Processing Group,
Favoritenstr. 9/1832, A-1040 Vienna, AUSTRIA
shao@prip.tuwien.ac.at

Abstract This paper presents a novel image representation, which incorporates the principles of Laplacian Pyramid into the irregular graph pyramid. The drawback of the regular Laplacian Pyramid is their lack to keep the topological structure of the image, due to the contraction process in building the Gaussian Pyramid. Irregular graph pyramid is able to hierarchically represent the topological structure of an image with multiresolution, where each level is a graph describing the image with various resolutions by contracting the graph from the level below. We build irregular Laplacian graph pyramid by storing the difference of levels in irregular graph pyramid. Experiments and results are presented in the paper to show the characteristic of the irregular Laplacian graph pyramid and some immediate advantages in computer vision applications.

1 Introduction

Image pixels are in generally highly correlated, it is common to have several areas of an image sharing the same or similar pixel values. Therefore, it is redundant to encode the image information by each of its pixel values.

In order to design an efficient compression code, it is necessary to find a representation that decorrelates the image pixels.

Laplacian Pyramid is a versatile data structure with many attractive features for image preprocessing. It represents an image as a series of quasi-bandpassed images, each sampled at successively sparser densities [1].

Laplacian Pyramid has been always related to Gaussian Pyramid, which is a low-pass filtered image sequence which is then subtracted from the original.

1.1 Organization of paper

In Section 2 we recall the regular Laplacian Pyramid. In Section 3 we explain the concept of combinatorial pyramid, which is pre-step of building irregular Laplacian Pyramid. In Section 4 we describe the method of building irregular Laplacian Pyramid, following by experiment and result discussion in section 5. In Section 6 we give the conclusion and open questions.

2 Recall of Laplacian Pyramid

Predictive coding is the base of Laplacian Pyramid, pixels are encoded sequentially in a raster format. However, prior to encoding each pixel, its value is predicted from previously coded pixels in the same way and proceeding raster lines [1].

Laplacian Pyramid has been always related to Gaussian Pyramid, which is a low-pass filtered image sequence which is then subtracted from the original.
2.1 Gaussian Pyramid
The first step to build a Laplacian Pyramid is to low-pass filter the original image \( g_0 \) to obtain the first image level \( g_1 \). This \( g_1 \) image is decreased in resolution and sample density. In the same way \( g_2 \) is a reduced version of \( g_1 \). To build a Gaussian Pyramid this process continue from the reduced image, and so on.

Suppose the original image, which is the base of the pyramid, is represented as an array \( g_0 \) containing \( C \) columns and \( R \) rows of pixels. Each pixel represents the light intensity at the corresponding image coordinate by an integer \( I \) with values between 0 and \( K-1 \) [1].

\( g_1 \) from pyramid level 1 consists of a low-pass filtered version of the previous level, \( g_0 \) from level 0. Where each value of the level 1 is computed as a weighted average of values from level 0 within a 5-by-5 window. The size of the weighting function is not critical [3]. In Fig. 2, is shown in 1D the reduction process, in 1D the density of the nodes are reduced by half while in 2D by fourth from level to level.

\[
g(0), \ldots, g_{i-1}, g_i, g_{i+1}, \ldots, g_{N-1} \quad (1)
\]

The reduction function from level to level is the averaging process such that, for levels \( 0 < l < N \) and nodes \( i, j \), \( 0 \leq i < C_l \), \( 0 \leq j < R_l \).

Where \( N \) refers to the number of levels in the pyramid, \( C_l \) and \( R_l \) are the dimensions of the \( l \)th level.

\[
g_l(i, j) = \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m, n) g_{l-1}(2i + m, 2j + n) \quad (1)
\]

The whole pyramid is only \( 4/3 \) the size of the original image. Each higher level of the pyramid is about half as large for each dimension as its previous level, as shown in Fig. 3.

The pyramid building process is equivalent to convolve the base image \( g_0 \) with a set of equivalent weighting functions \( h_l \) [1].

\[
g_l = h_l \ast g_0
\]

The effect of convolving an image with one of the equivalent weighting functions \( h_l \) is to blur, or low-pass filter, the image. The pyramid algorithm reduces the filter band limit by an octave from level to level, and reduces the sample interval by the same factor [1]. This is a very fast algorithm, requiring fewer computational steps to compute a set of filtered images than are required by the fast Fourier transform to compute a single filtered image [3].

![Figure 2: Gaussian Pyramid.](image)

2.2 Laplacian Pyramid
The expansion function is the inverse of the reduction function.

It expands an \((M + 1)\) by \((N + 1)\) array into a \((2M + 1)\) by \((2N + 1)\) array by interpolating new node values between the given values of pixels.

The expansion applied to the array \( g_l \) of the Gaussian Pyramid would result into an array \( g_{l+1} \) which is the same size as \( g_{l-1} \).

For levels \( 0 < l < N \) and \( 0 < n \) nodes \( i, j \), \( 0 \leq i < C_{l-n} \), \( 0 \leq j < R_{l-n} \)

\[
g_{l+1}(i, j) = \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m, n) \ast g_{l-1}
\]

(2)

If we apply expansion function \( l \) times to the image \( g_l \), the result is \( g_{l+1} \) which is the same size as the original image \( g_0 \).

The purpose of constructing the reduced image \( g_l \) is that it will be used as a prediction for pixel values in the original image \( g_0 \).

To obtain a compressed representation, we encode the error image which remains when an expanded \( g_l \) is subtracted from \( g_0 \). This image becomes the bottom level of the Laplacian Pyramid. The next level is generated by encoding \( g_1 \) in the same way.

The Laplacian Pyramid is a sequence of error images \( L_0 \), \( L_1 \), \( \ldots \), \( L_{N-1} \). Each is the difference between two levels of the Gaussian Pyramid. Thus, for \( 0 \leq l < N \), \( L_l = g_l - \text{EXPAND}(g_{l+1}) \) [1].

In Laplacian Pyramid, the value at each node is the difference between the convolutions of two equivalent weighting functions \( h_{l+1} \) with the original image.

Gaussian Pyramid can be seen as a set of filtered copies of the original image, while the Laplacian Pyramid can be seen as a set of bandpass filtered copies of the image. Image features such as edges appear enhanced in the Laplacian Pyramid as shown in Fig. 9. These enhanced features depend on the size, where fine details are notable in \( L_{0,0} \) and progressively vanish in the higher levels.

The original image can be recovered completely by expanding. First, expand \( L_N \) once and add it to \( L_{N-1} \), then...
3 Combinatorial Pyramid

An irregular graph pyramid combines graph structures with hierarchies. Each level is a graph representation describing the image with various resolutions by contracting the graph from the level below. Graph representations have been investigated widely during last decades for representing structural information in various domains in computer vision such as image segmentation and object recognition.

The graph structure is defined as follows: graph $G(V, E)$ consists of vertices $v \in V$ and edges $e \in E$. An edge $e$ connects two vertices, $(v, w, e) = (v, w)$. The vertices $V$ and edges $E$ of the image graphs carry not only the structural information, but also the additional information.

For this paper we consider using combinatorial maps to present the graph in each level of the irregular pyramid. A combinatorial map is a topological model which allows to represent subdivided objects as planar graphs. A 2D combinatorial map is defined by a triplet $M = (D, \sigma, \alpha)$ where $D$ is a finite set of darts, $\sigma$ is a permutation on $D$ and $\alpha$ is an involution on $D$ without fixed point [7]. For each dart, $\sigma$ gives the next dart by turning around the vertex $v$ in the positive orientation (clockwise); For each dart, $\alpha$ gives the other dart of the same edge $e$. There are always two darts corresponding to a same edge, $\alpha$ allows to retrieve edge $e$, and $\sigma$ allows to retrieve the vertex $v$.

We can see in Fig. 4, the graph is an example of 2D combinatorial map.

![Figure 4: Combinatorial map, contraction/removal operations.](image)

This map can be explicitly defined by giving the set of darts, permutations $\sigma$ and involutions $\alpha$.

<table>
<thead>
<tr>
<th>$D$</th>
<th>0 1 2 3 4 5 6 7 8 9 10 11</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma$</td>
<td>8 10 1 6 0 7 9 11 4 3 2 5</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>1 0 3 2 5 4 7 6 9 8 11 10</td>
</tr>
</tbody>
</table>

During the process of building up the irregular graph pyramid, contraction process removes the edges from the input graph while simultaneously merging together the vertices it used to connect [7]. And reduction process is to take the attributes of all children as input and then compute the parent’s attribute as output.

Taking a simplified image of a cup as example (Fig 5), we build the base graph as the input image, where each vertex represent a pixel in the input image. Then use the contraction methods to build the irregular pyramid. Such approach would lead to a pyramidal structures like Fig 5:

![Figure 5: Irregular Graph Pyramid.](image)

Level 0: The base level of the pyramid consists in a geometric description of the underlying image (here a simplified image of a cup).

Level 1: The second level of the pyramid, simpler boundaries are abstracted from base level (like the handle and the logo of the cup).

Level 2: Adjacent parts of the cup are grouped in order to represent compound abstract objects.

4 Irregular Laplacian Graph Pyramid

Similar as regular Laplacian image pyramid, the irregular Laplacian Pyramid stores the difference of the child’s content with expanded content, but from irregular graph pyramid instead of Gaussian Pyramid. However, the expansion function is slightly different from regular Laplacian Pyramid. In this section, we will formalize the process of building the irregular Laplacian Pyramid.

4.1 Correction Vector

In the process of building up irregular graph pyramid, each level graph is obtained by contracting the graph from the level below. Parent inherits the position of the surviving child. The property of the parent is computed from the properties of all its children by a certain function, such as weighted average. Depending on the concrete task, this function varies. Let $V_p$ refers the parent vertex, and $V_c$ refers the child vertex:

$\text{pos}(V_p) = \text{pos}(V_c)$ if $V_c$ survives

$d(V_c) = \text{pos}(V_c) - \text{pos}(V_p)$ if $V_c$ does not survive

Each non-surviving vertex stores the difference between its coordinate and the its parent’s coordinates. We call this difference as correction vector. Because the parent’s position is equal to the surviving child’s position, those surviving vertices have correction vector of value 0. As show in Fig 6, vertex $V_{c_1}$ is the surviving vertex. So its parent $V_p$ has the same
4.2 Expansion Kernel

Expansion kernel expands (interpolates) the properties of the parents cells into the children’s content at the higher resolution level. With correction vector, the position of the child can be obtained by adding its correction vector to its parent’s position. \( G_k \) refers the graph in level \( k \), and \( G_{k-1} \) refers to the graph in one level below.

\[
\text{pos}(V_c) = \text{pos}(V_p)
\]

where \( V_p \in G_k, V_c \in G_{k-1} \cap G_k \)

\[
\text{pos}(V_c) = \text{pos}(V_p) + L(V_c)
\]

where \( V_p \in G_k, V_c \notin G_{k-1} \cap G_k \)

Let’s call the irregular graph pyramid as \( G \), and the irregular Laplacian Pyramid as \( L \). For the properties of vertices, each level of the Laplacian Pyramid can be obtained by taking the difference of the adjacent levels, same as in regular Laplacian Pyramid.

\[ L_k = G_k - \text{Expand}(G_{k+1}) \]

The reconstruction process is same as in the regular case. The original image can be reconstructed by expanding. First, expand \( G_{k+1} \) once and add it to \( L_k \), then expand this image once and add it to \( L_{k-1} \), and so on until level 0 is reached and \( G_0 \) is recovered.

\[ G_k = \text{Expand}(G_{k+1}) + L_k \]

The main process of building irregular Laplacian Pyramid is divided by three steps: 1). Build Irregular Graph Pyramid \( G \) on the target image. 2). For each level \( k \) in the \( G \), we expand level \( G_{k+1} \) using the correction vectors to have the same size of content as the level \( k \) in \( G_k \). 3). Take the difference of the expanded level \( G_{k+1} \) with the level \( G_k \).

5 Experiment

We take a gray image as example, to demonstrate the experiment of building the irregular Laplacian Pyramid on it. For initialization, we convert this gray image into a graph which is encoded as combinatorial map (Fig 7). This graph is the base level of the irregular graph pyramid. The red crosses represent the pixels in the input image, which are the faces in the base graph. Faces are bounded by the blue lines presenting the edges in the graph, and the intersections of the blue lines present the vertices in the graph which are linked by the edges. When the edges are contracted or removed during the building process of irregular graph pyramid, the faces would get merged. So in this pyramid, faces in the graphs represents image regions and the edges of graphs represent the boundaries of image regions.

Each level in the irregular graph pyramid is a graph, while the vertices store their geometric coordinates from the surviving children’s position in the original image. With the geometric coordinates of the vertices, we convert the graph in each level into the image of original size for the visualization purpose, see the results in Fig 8. Fig 8 shows the visualization of irregular graph pyramid built on this image. The contraction kernel we select is Maximum Independent Directed Edges Set [14].

Compared to the results from Gaussian Pyramid, the irregular graph pyramid preserves the structure of the object segments, while Gaussian Pyramid simply blurred the object by applying low pass filter. The high frequency information, such as the shadow of the stick of the apple survives until higher levels in the irregular graph pyramid. However, in Gaussian Pyramid, high frequency information is lost due to low pass filtering building process. In the low resolution (high level) of Gaussian Pyramid, it is nearly impossible to define the shape of the target object while the shape is still preserved in irregular graph pyramid.
As shown in Fig 9, the results of the irregular Laplacian Pyramid obtain the same advantage as irregular graph pyramid, the advantage of adapting image data into structure, keeping structural information in the pyramid and preserving the topology information.

We may view the regular Laplacian Pyramid as a set of bandpass filtered copies of the original image. Irregular Laplacian Pyramid shows the bandpass filter effect regarding on the length of correction vectors of vertices. As presented in Fig 10, 11 and 12, the length of correction vectors increases as the level of the pyramid increases.

For the graph $L_i$ of level $i$ in the irregular Laplacian Pyramid, $N_i$ refers to the total number of vertices in $L_i$.

The histograms of level $i$ (Fig 10, Fig 11, Fig 12) show the distribution of vertices with various correction vector length, where horizontal axis presents the length of correction vectors while the vertical axis presents the percentage of vertices with certain length of correction vector respect to the total number of vertices $N_i$. The histograms show the band pass characteristic regarding to the length of correction vector, which is similar to regular Laplacian Pyramid, as regular Laplacian Pyramid is also band pass filter regarding to the frequency of the image.

In the first level of the pyramid, most of the vertices have correction vector with length 1. Those vertices are the adjacent neighborhood of the surviving vertices, with distance of 1 pixel. The vertices of length 0 are surviving vertices. All the correction vectors length fall in the range between 0 to 5.

In the second level of the pyramid, the overall length of correction vectors increases compared to level 1, with the maximum value exceeds 5.

In the third level of the pyramid, the overall length of correction vectors increase considerably, with the maximum value exceeds 20, because of large regions get merged.

6 Conclusion
This paper presents a novel image representation, irregular Laplacian graph pyramid. It integrates the principles of regular Laplacian Pyramid with the main advantages of irregular Laplacian Graph Pyramid.
lar graph pyramid. In future work, irregular Laplacian graph pyramid can be applied in image compression and other computer vision problems. The irregular Laplacian Pyramid preserves topological information of target objects. Therefore, this representation may be applied in motion detection, as in which one needs to find the objects movements in the scene.

Acknowledgement

Supported by the Austrian Science Fund under grant P18716-N13

References


Multi-View Integration for a Rotating 3D Object

Luis A. Mateos and Walter G. Kropatsch

Vienna University of Technology,
Pattern Recognition and Image Processing Group,
Favoritenstr. 9/1832, A-1040 Vienna, AUSTRIA
lam@prip.tuwien.ac.at

Abstract The aim of this paper is to propose a novel object representation for a 3D rigid object in tracking applications. A 3D volumetric object is bounded by a closed surface (2D manifold) on which visible features are perceived (e.g. color, texture). In our representation, we collect the topological structures from the visible surface of the target object. As the object rotates, new visible parts of surface would reveal new topological structures. This collection of small 3D surface patches is reconstructed by processing topological completion. And the object representation is incrementally updated from observing the target object in a video sequence.

1 Introduction

Tracking methods using structural information often can overcome the disadvantage of traditional appearance based representations. Locally, features describe the object details; globally, the relations between features encode the object structure. By using structural information, the representation is able to track objects having certain amount of variations caused by illumination changes, occlusion as well as discriminating between structurally different object types. By exploiting the spatial and temporal structure of the scene [1] Lopez and Artner improve the performance of object tracking in motion sequences. They present some cases where Mean Shift fails and propose an improved Mean Shift with a structural tracking approach which is robust to different types of occlusions. In [2], the video sequences are described by the so-called Spatio Temporal Region Graph (STRG). And [3] describes local and relational object information by use of Attributed Relational Graphs (ARGs) where their goal is to recognize and track objects.

However, all these methods assume the visible topological structures of the target object do not change significantly when the object rotates. In practical, a non uniform object will change its topological structure due to rotation or movement of camera. We propose to extend tracking with structure methods into the non uniform 3D object tracking applications, taking into consideration the changes in structures when the target object rotates. In our previous work we focus on tracking the translation and the scaling movement for an approaching object by using Irregular Graph Pyramid [13].

1.1 Tracking 3D Object.

In 3D object tracking, most model-based methods use 3D models prepared off line. Vacchetti et al. [8] propose a tracker based on matching with key frames. The method obtains impressive results on out-of-plane rotation data. However it still needs off line manual selection of key frames which are critical for its stability.

Instead of using a single camera, several methods build elaborated 3D models from multiple views. These methods rely on carefully constructed and expensive setup and require special scene arrangement because they are based on scene/object segmentation [9, 10, 11, 12]. Other multi view approaches mostly represent objects as blobs. Blob representation assumes that the appearance of an object does not significantly change when the object rotates.

We propose a framework to automatically create and update a 3D object model for tracking by using a single camera. Consider in a video frame, a 3D object may be occluded or partially visible. We call the visible part of the surface the front surface. In our framework, from a single image frame, the front surface is extracted as a graph. This extracted graph embeds the topological structure and discriminative visible features of the object: in the vertex, attributes like size, color and position of the corresponding pixels (region) can be stored and the spatial edges are used to specify the spatial relationships (adjacency, border) between the vertices (regions)[1]. For initialization, the graph is closed on the invisible back side to create a closed 2D manifold. When the target object rotates, some of its hidden structure will appear. We add those new topological structure into the previous 2D manifold to obtain the updated object representation. When the camera has covered all the aspects of the object, which means all the parts of the object have been visible to the camera, the topological structure of the target object is complete, this is the process we defined as topological completion.

Major contributions of this paper are 1) The object representation completes the visible topological structures of the target object. 2) The object representation completes the invisible back surface topologically. It isolates the object’s surface from the observed image frame. 3) The object representation is generated by segmenting the observed object in a video and it is incrementally updated.
1.2 Organization of paper
In section 2 we recall the concept of Combinatorial Maps. In section 3 we describe how to build and update the object representation by observing the target object in a video. Section 4 shows an experiment for two image frames, and in section 5 the conclusion and open questions.

2 Recall of Combinatorial Maps
Graph representations have been investigated widely during last decades for representing structural information in various domains in computer vision such as image segmentation and object recognition. The graph structure of an image provides an essential clue for object tracking in video, as it allows to identify objects in images by their topological structure.

The graph structure is defined as follows: graph $G(V, E)$ consists of vertices $v \in V$ and edges $e \in E$. An edge $e$ connects two vertices, $v, w$, $e_i = (v, w)$. The vertices $V$ and edges $E$ of the image graphs carry not only the structural information, but also the additional information. Specifically in the purpose of video object tracking, the additional information of features which are able to discriminate the object from the background (e.g. color, texture) are also abstracted and embedded in the image graphs.

We use the combinatorial map as a mathematical model of graph representation. The combinatorial map is introduced by Cori [5] the first time. Alternatively, [4] defines the combinatorial map by the triplet $G = (D, \sigma, \alpha)$ where $D$ represents the set of darts, $\sigma$ is a permutation in $D$ connecting all darts encountered when turning clockwise around a vertex $v$ and $\alpha$ is an involution connecting two darts that belong to the same edge $e$.

We can see in Fig. 1, the graph is an example of 2D combinatorial map.

3 Topological Representation of 3D object
If a non uniform object rotates, it may show or hide some of its structural descriptors, for example, a rotating cup will reveal the handle that before was hidden, and hide the logo that before was shown.

For a general valid representation, the use of geometry and topology must incorporate the local object appearance. Topology and shape are complementary, and it is very useful to compute both types of information [6].

3.1 Initialization
Different scenarios of initializing a closed 2D surface can be conceived: for instance, a box with a picture of a cup on front might be represented by two layers in topological means (front and back of the box).

We abstract the topological structures in terms of a 2D combinatorial map from a single image frame. The initial graph only includes the visible features and structures from the visible surface (we call it front surface). We can still assume the back surface (invisible part of the object) has the same structure as front. In such way, we create a back surface by copying the front surface and link these two graphs [14]. By doing this, an observed hole in the 2D image creates a tunnel in the 3D surface. Figure 3 shows an example of a 3D object with two tunnels. Fig. 3(1) shows the front visible surface of this object. One can image the two holes in the 2D image represent two tunnels in 3D object structure. Fig. 3(2) shows the graph abstracted from the front surface. We duplicate this graph, and link them together to achieve a closed 2D manifold shown in Fig. 3(3). In such way, we are able to obtain a closed 2D manifold to represent the closed surface of the target object.

This map can be explicitly defined by giving the set of darts, permutations $\sigma$ and involutions $\alpha$.

<table>
<thead>
<tr>
<th>Darts</th>
<th>0 1 2 3 4 5 6 7 8 9 10 11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permutation</td>
<td>8 10 1 6 0 7 9 11 4 3 2 5</td>
</tr>
<tr>
<td>Involition</td>
<td>1 0 3 2 5 4 7 6 9 8 11 10</td>
</tr>
</tbody>
</table>

Figure 1: Combinatorial map.
In most of cases, we know that the back surface of the object is not the same as front. We label the back surface as unknown structure for future update by processing topological completion.

3.2 Topological Completion

Each of the two layers (front and back surface) is again described by a topological description combined with its geometrical appearance. One can imagine that the use of the surface appearance (here the picture of the cup) increases the discriminative power of such a representation.

In Figure 4 shows the simplified updating process when the cup rotates from the front view (logo on the front side) to the left view (logo on the left side). A combinatorial map is used. A combinatorial map is only represented by darts, permutation \( \sigma \) and involution \( \alpha \), without vertices and edges. However \( \sigma \) allows one to retrieve edges and \( \alpha \) allows one to retrieve vertices. Therefore we still adopt the concept of vertices and edges in order to conveniently describe our graph operation with reference to graph figures.

The process of adding the new structure (vertices, edges) into the current combinatorial map includes two operators: face split and face merge.

The first image frame \( I_1 \) is abstracted as a graph \( G_1 \), and the second image frame \( I_2 \) is abstracted as a graph \( G_2 \). As shown in Fig. 4(1), the structure of the cup handle does not appear in graph \( G_1 \) but appears in \( G_2 \). It belongs to the new structures we intend to add.

The edge \( e(v_2, v'_2) \) (marked with red color) is the right side boundary edge of \( G_1 \). When the cup rotates from right to left, \( e \) would be in the middle of \( G_2 \). New structures may appear on the right side of \( e(v_2, v'_2) \) in \( G_2 \).

Therefore \( e(v_2, v'_2) \) in \( G_2 \) provides a clue as a separation boundary to separate graph \( G_2 \) into two sub graphs: graph \( G_{old} \) embedding existing structures and graph \( G_{new} \) embedding new structures due to rotation, as shown in Fig. 4(2). This separation step is done by face split operation. We split face \( f \) into two faces \( f_{old} \) and \( f_{new} \) by the edge \( e(v_2, v'_2) \). All the vertices and edges which compose the face \( f_{old} \) are the existing vertices and edges in previous graph \( G_1 \), while the ones in \( f_{new} \) store new structure information.

To achieve the goal of obtaining adaptive representation, we must merge the previous structure graph \( G_1 \) with the new structure \( G_{new} \) to obtain an updated graph, which integrates both structures from the previous image frame and the current second image frame. As shown in Fig. 4(3), we merge face \( f_{new} \) (derived from \( G_2 \)) with face \( f_1 \) (derived from \( G_1 \)) to obtain face \( f_2 \). The face merge operation is an inverse operation of face split operation, starting from two adjacent faces sharing the same edge \( e(v_2, v'_2) \), in gathering them into a single face union. Split and merge algorithm is achieved in a local way, by processing each dart of the region in the algorithm proposed by [7]. The merge operation brings an integrated graph which contains the structure of the complete logo (which \( G_2 \) does not contain) and the cup handle (which \( G_1 \) does not contain).

If an object rotates 360°, revealing its complete structure, then it is possible to compute its full topological completion and obtain its 2D manifold.

In Figure 5, taking the example of the cup, by rotating it 90° per frame, for four frames, it is possible to compute the topological completion, the only constraint is to label a reference point in order to know when it is completed. In this case, the reference point is the logo on the cup because of its discriminative feature (the color).

3.3 Critical Poses

If the object rotates to such a characteristic view that the visible part has no overlap area with previous video frames, we call the object is in critical pose, which is challenging for
this tracking method. In this case, the current image frame encodes the new structure information but without enough previous structure information for making correspondence to the existing object representation. The tracking may get unstable. We still abstract structures from this image frame, but save the structures in memory without adding it to the model directly. After the object leaves the critical pose, in later image frames, if other new structures appear which can bridge those isolate structures into the model, we can consider to add it. Otherwise, these structural information has to be deleted.

4 Experiment

The object used for the experiment is a cup with the PRIP(Pattern Recognition and Image Processing Group) logo. The experiment consists in take a front picture of the object as first graph image, Fig. 6, then the cup is rotated and another picture is taken, Fig. 7. The rotation is arbitrary. With these two pictures we implement the topological completion technique described in Figure. 4.

By using the chessboard technique it is possible to detect the shape of the 3D object, and subdivide this 3D object in proportional pieces of information. In this case the cup is a cylinder as show in Fig. 8, which shape is confirmed by the chessboard spatial - arrange.

In Figure 9, by overlapping all the corners detected from the chessboard over the 3D object, it is possible to subdivide the 3D object into proportional areas, in such a way we are able to know the information of each area or cell.

5 Conclusion

This paper presents a multi-view integration representation for tracking 3D rigid object. By processing the topological completion, we are able to collect and reconstruct the complete visible topological structures of the target object. We encode those structures and discriminative appearance fea-
Multiview Integration for a Rotating 3D Object

Luis A. Mateos and Walter G. Kropatsch

Figure 10: 2D manifold.

Figure 11: Topological completion.

tures in a 2D manifold graph, which is mathematically represented as closed 2D combinatorial map. Some open questions we have currently are how to make this model robust to critical poses, to complex objects, to noise and illumination effect such as shadow. Another direction is to apply a hierarchical structure such as Irregular Graph Pyramid to abstract our topological representation in an efficient way.

Acknowledgement

Supported by the Austrian Science Fund under grant P18716-N13

References


[13] Luis A. Mateos, Dan Shao and Walter G. Kropatsch: Expanding Irregular Graph Pyramid for an Approaching Object. CIARP 2009: 885-891
Multivariate Online Kernel Density Estimation

Matej Kristan¹ ² and Aleš Leonardis¹

¹Faculty of Computer and Information Science, University of Ljubljana
²Faculty of Electrical Engineering, University of Ljubljana
mailto:matej.kristan@fri.uni-lj.si

Abstract We propose an approach for online kernel density estimation (KDE) which enables building probability density functions from data by observing only a single datapoint at a time. The method maintains a non-parametric model of the data itself and uses this model to calculate the corresponding KDE. We propose an automatic bandwidth selection rule, which can be computed directly from the non-parametric model of the data. Low complexity of the model is maintained through a novel compression and refinement scheme. We compare the online KDE to some state-of-the-art batch KDEs on examples of estimating distributions and on an example of classification. The results show that the online KDE generally achieves comparable performance to the batch approaches, while producing models with lower complexity and allowing online updating using only a single observation at a time.

1 Introduction

Many tasks in machine learning and pattern recognition require building models from observing sequences of data. In real-world environments all the data may not be available in advance, or we even want to observe the process for an indefinite duration, while continually providing the best estimate of the model from the data observed so far. These tasks require online construction of the models.

A popular approach to generating models from data is to model the probability density function (pdf) associated with the observed data. Traditionally, parametric models based on Gaussian mixture models (GMM) [9] have been applied with some success in estimation of the pdf when all data are observed in advance. The parametric mixture models typically require specifying the number of components in the mixture and may not capture the complete structure of the underlying pdf. Non-parametric methods such as Parzen [21, 11, 28] kernel density estimators (KDE), with Gaussian kernels, alleviate this problem by treating each observation as a component in the mixture model and assuming all components have equal covariances (bandwidths). The problem of KDE is then how to automatically set this bandwidth. While most of the literature on the bandwidth selection have dealt with one-dimensional problems, recently Murillo and Rodriguez [20] have proposed an efficient method for calculating the multivariate bandwidths. One drawback of the KDEs is that their complexity (number of components) increases linearly with the number of the observed data. To remedy this, methods have been proposed to reduce the number of components either to a predefined value [11, 28] or by optimizing a database-based [10] and MDL-based [18] cost functions.

There have been several attempts to merge the non-parametric quality of the kernel density estimators with the Gaussian mixture models in online applications. Arandjelović et.al. [1] proposed a scheme for online adaptation of the Gaussian mixture model which can be updated by one sample at a time. However, a strong restriction is made that data is temporally coherent in feature space, which prevents its use in general applications. Priebe and Marchette proposed an online EM algorithm called active mixtures [23] which is less sensitive to the data order, allows adaptation from single observation at a time, and includes a heuristic for allocating new components. Song et. al. [24] aim to alleviate the temporal restrictions by processing data in large blocks. Bischof and Leonardis [4] use a radial-basis-function (RBF) based network and apply an MDL-based procedure for basis function allocation and deletion – they assume a predefined initial rbf size. Deleclerq and Piater [7] assume each data is a Gaussian with a predefined covariance. All data are stored in the model and a heuristic is used to determine when a subset of the data can be replaced by a single component. Han et. al. [14] proposed an online approach inspired by the kernel density estimation in which each new observation is added to the model as a Gaussian kernel with a predefined bandwidth. The model’s complexity is maintained low by retaining only the modes of the distribution, which they approximate by Gaussians. This approach deteriorates in situations when the assumed predefined bandwidths of kernels are too restrictive, and when the distribution is locally non-Gaussian (e.g., in skewed or uniform distribution). Recently, Kristan et.al [17] have proposed an online kernel density estimator, which uses the least assumptions in comparison to the related methods, but is restricted to one-dimensional problems.

1.1 Our approach

We propose a, new, multivariate online version of the kernel density estimator, which enables adaptation from only a single observation at a time. Our approach is grounded in the following two key ideas. The first key idea is that, unlike the related approaches, we do not attempt to build a model of the target distribution directly, but rather maintain a
non-parametric model of the data itself in a form of a sample distribution – this model can then be used to calculate the kernel density estimate of the target distribution. The second key idea is that we treat each new observation as a distribution in a form of a Dirac-delta function and we model the sample distribution by a mixture of Gaussian and Dirac-delta functions. During online operation the sample distribution is updated by each new observation in essentially the following three steps (Figure 1a): In the first step, we update the sample model with the observed data-point. In the second step, the updated model is used to calculate the optimal bandwidth for the KDE – here, the main issue is how to calculate the bandwidth without having access to the previously observed individual samples. In the third step, the sample distribution is refined and compressed. This step is required because, without compression, the number of components in our model would increase linearly with the observed data. However, it turns out that a valid compression at one point in time might become invalid later, when new data-points arrive. We therefore require a refinement algorithm to detect such events and to recover from them. The main issues here are: (i) how to devise an optimization which would efficiently compress the sample model, (ii) how to determine the extent of the allowed compression and (iii) how to recover from early compressions. To allow the recovery from early compression, we keep for every component another model of the data that generated that component in a form of a mixture model with at most two components (Figure 1b). After the compression and refinement step, the KDE can be calculated as a convolution of the (compressed) sample distribution with the optimal kernel calculated at step 2 (see Figure 2).

Our main contribution is the approach for multivariate online kernel density estimation (oKDE), which enables construction of a multivariate density estimate by observing only a single sample at a time and automatically adjusts its complexity. In contrast to the standard bandwidth estimators, which require access to all observed data, we derive a method which can use a mixture model (sample distribution) instead and can be applied to multivariate problems. To enable a controlled compression of the sample distribution, we propose a compression scheme which estimates the distance between the KDE calculated before and after the compression. To this end, we propose an approximation to the multivariate Hellinger distance. We also propose a scheme to detect and recover from early over-compressions.

The remainder of the paper is structured as follows. In Section 2 we define our model. In Section 3 we derive a rule for automatic bandwidth selection. The compression and model refinement scheme is described in Section 4. The online KDE algorithm is presented in Section 5 and in Section 6 we compare it to some existing online and batch state-of-the-art KDE algorithms on examples of estimating distributions and on a classification example. We conclude the paper in Section 7.

2 The model definition

Each separate data-point can be presented in a distribution as a single Dirac-delta function, with its probability mass concentrated at that data-point. Noting that a Dirac-delta can be generally written as a Gaussian with zero covariance, we define the model of (potentially compressed) d-dimensional data as an N-component Gaussian mixture model

$$p_{\alpha}(x) = \sum_{i=1}^{N} \alpha_i \phi_{\Sigma_i}(x - x_i),$$

where

$$\phi_{\Sigma}(x - \mu) = (2\pi)^{-\frac{d}{2}} |\Sigma|^{-\frac{1}{2}} e^{-\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu)}$$

is a Gaussian kernel centered at $\mu$ with covariance matrix $\Sigma$. We call $p_{\alpha}(x)$ a sample distribution and a kernel density estimate (KDE) is defined as a convolution of $p_{\alpha}(x)$ by a kernel with a covariance matrix (bandwidth) $H$ (see Figure 2):

$$\hat{p}_{\text{KDE}}(x) = \phi_{H} * p_{\alpha}(x) = \sum_{i=1}^{N} \alpha_i \phi_{H + \Sigma_i}(x - x_i).$$

To maintain a low complexity of the KDE during online operation, the sample distribution $p_{\alpha}(x)$ is compressed...
from time to time. As noted in the introduction, compressions at some point in time may later become invalid as new data arrive. To detect and recover from these early over-compressions, we keep an additional model of data for each component in the mixture model. We therefore define our model of the observed samples as

\[ S_{\text{model}} = \{ p_i(x), \{ q_i(x) \}_{i=1:N} \}, \]

where \( p_i(x) \) is the sample distribution and \( q_i(x) \) is a mixture model (with at most two components) for the \( i \)-th component in \( p_i(x) \) (Figure 1b). To obtain a KDE, we need to compute the optimal bandwidth from all the observed samples, which are now summarized in the sample model \( p_s(x) \). In the following we propose a method for calculating this bandwidth.

3 Estimation of the bandwidth

If we retained (did not compress) all the observed samples in the sample model, then the sample distribution \( p_s(x) \) would contain only components with zero covariances (i.e., \( \Sigma_{ij} = 0 \) for all \( i \) and \( \Sigma(M_{\text{DE}}) = \sum_{i=1}^{N} \alpha_i \phi_H(x - x_i) \). The goal of all KDE methods is to determine the kernel bandwidth \( H \) such that the distance between the \( \hat{p}_{\text{KDE}}(x) \) and the unknown pdf \( p(x) \), that generated the data, is minimized. A classical measure used to determine the kernel bandwidth \( H \) such that the distance between the \( \hat{p}_{\text{KDE}}(x) \) and the unknown pdf \( p(x) \), that generated the data, is minimized.

A classical measure used to define the closeness of the estimator \( \hat{p}_{\text{KDE}}(x) \) to the underlying pdf is the asymptotic mean integrated squared error (AMISE), defined as ([27], pp.95-98),

\[ AMISE = (4\pi)^{-\frac{1}{2}}|H|^{-\frac{1}{2}}N^{-1}\frac{1}{4} \int \text{tr}^2 \{ H \phi_H(x) \} dx, \]

where \( \text{tr} \{ \cdot \} \) is the trace operator, \( \phi_H(x) \) is a Hessian of \( p(x) \), and \( N = \sum_{i=1}^{N} \alpha_i^2 \). If we rewrite the bandwidth matrix into \( H = h^2 F \) for \( |F| = 1 \), and assume for now that \( F \) is known, then (5) is minimized at \( h_{\text{opt}} \)

\[ h_{\text{opt}} = [d(4\pi)^{-\frac{1}{2}}N_{\alpha} R(p,F)]^{-\frac{1}{2}}, \]

where the term

\[ R(p,F) = \int \text{tr}^2 \{ F \phi_H(x) \} dx \]

is a functional of the second-order partial derivatives of the unknown distribution \( p(x) \). In principle, this functional could be estimated using the plug-in methods [27], however, these are usually numeric, iterative, and assume we have access to all the observed samples. In our case, we maintain only a (compressed) mixture model of the samples, and we require an approximation to the functional using this mixture model.

We first note (see, eg., [27]) that \( R(p,F) \) can be written in terms of the expectations of the derivatives \( \psi_r = \int p^{(r)}(x) p(x) dx \). We can then use the sample distribution \( p_s(x) \) to obtain the following approximations

\[ p(x) \approx p_s(x) : p^{(r)}(x) \approx p_s^{(r)}(x), \]

where we approximate the derivative of \( p(x) \), \( p_s^{(r)}(x) \), by using the following kernel density estimate

\[ p_s(x) = \phi_s(x) * p_s(x) = \sum_{j=1}^{N} \alpha_j \phi_{\Sigma_{ij}}(x - \mu_j). \]

The estimate \( p_G(x) \) plays a role of the so-called pilot distribution with covariance terms \( \Sigma_{ij} = G + \Sigma_{ij} \) and \( G \) is called the pilot bandwidth. Using the approximations in (8) and the Fact C.2.3. on p.181 in [27], we can approximate \( R(p,F) \) by

\[ \hat{R}(p,F,G) = \int \text{tr} \{ F \phi_{p_G}(x) \} \text{tr} \{ F \phi_{p_s}(x) \}. \]

Since \( p_s(x) \) and \( p_G(x) \) are Gaussian mixture models, (10) can be calculated using only matrix algebra:

\[ \hat{R}(p,F,G) = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j \phi_{\Sigma_{ij}}(\mu_i - \mu_j) \times \left[ 2\text{tr}(A_{ij}B_{ij}) + \text{tr}^2(FC_{ij}) \right]. \]

Derivation of (11.12) is rather laborious, and is based on the M. P. Wand’s [26] study of an integral similar to (10). In the interest of space we removed the derivation here.

Note that we still have to determine the pilot bandwidth \( G \) of \( p_G(x) \) and the structure \( F \) of the bandwidth matrix \( H \). We use the empirical covariance of the observed samples \( \Sigma_{\text{emp}} \) to approximate both. First we resort to a practical assumption [27, 8] that the structure of the bandwidth \( H \) can be reasonably well approximated by the structure of the covariance matrix of the observed samples, and thus

\[ F = \Sigma_{\text{emp}} |\Sigma_{\text{emp}}|^{-1/2}. \]

We estimate the pilot bandwidth \( G \) by a normal-scale rule [27]. The normal-scale provides a bandwidth that is optimal in AMISE sense if the unknown distribution \( p(x) \) is in fact normal. While this assumption is too restrictive to directly estimate \( H \), it is admissable in practice for estimation of the bandwidths for the derivatives (see, eg. [27] page 71). The pilot bandwidth using the multivariate normal-scale rule for the derivative ([27], page 111) is given by

\[ G = \Sigma_{\text{emp}} \left( \frac{4}{(d+2)N_{\alpha}} \right)^{\frac{1}{2}}. \]

4 Compression of the sample model

The objective of the compression is to approximate the original \( N \)-component sample distribution

\[ p_s(x) = \sum_{i=1}^{N} w_i \phi_{\Sigma_{ij}}(x - \mu_i) \]

by a \( M \)-component, \( M < N \), equivalent \( \hat{p}_s(x) \)

\[ \hat{p}_s(x) = \sum_{j=1}^{M} \hat{w}_j \phi_{\Sigma_{ij}}(x - \hat{\mu}_j), \]

Kristan Matej, Leonardis Aleš

79
such that the resulting KDE does not change significantly. Since a direct optimization (e.g., [17]) of the parameters in \( p_s(x) \) can be computationally prohibitive, and prone to slow convergence even for moderate dimensions, we resort to a clustering-based approach. The main idea is to identify clusters of components in \( p_s(x) \), such that each cluster can be sufficiently well approximated by a single component in \( \hat{p}_s(x) \). Let \( \Xi(M) = \{ \pi_j \}_{j=1:M} \) be a collection of disjoint sets of indexes, which cluster \( p_s(x) \) into \( M \) sub-mixtures. The sub-mixture corresponding to indexes \( i \in \pi_j \) is defined as

\[
p_s(x; \pi_j) = \sum_{i \in \pi_j} w_i \phi_{\Sigma_i}(x - \mu_i)
\]

(17)

and is approximated by the \( j \)-th component \( \hat{w}_j \phi_{\Sigma_j}(x - \hat{\mu}_j) \) of \( \hat{p}_s(x) \). The parameters of the \( j \)-th component are defined by matching the first two moments (mean and variance) \[3\] of the sub-mixture:

\[
\hat{w}_j = \sum_{i \in \pi_j} w_i, \quad \hat{\mu}_j = \hat{w}_j^{-1} \sum_{i \in \pi_j} w_i \mu_i
\]

\[
\hat{\Sigma}_j = \hat{w}_j \sum_{i \in \pi_j} w_i (\Sigma_i + \mu_i \mu_i^T) - \hat{\mu}_j \hat{\mu}_j^T.
\]

(18)

We seek to identify the clustering assignment \( \Xi(M) \), along with \( M \), such that the error of any cluster approximation under the current KDE does not exceed a prescribed value \( D_{th} \). In other words, we want to minimize the following composite error function

\[
E(\Xi(M)) = \max \limits_{\pi_j \in \Xi(M)} E(\hat{p}_s(x; \pi_j), \mathcal{H}_{opt}) \quad \text{such that} \quad E(\Xi(M)) \leq D_{th},
\]

(19)

where \( E(\hat{p}_s(x; \pi_j), \mathcal{H}_{opt}) \) denotes the local error induced by the \( j \)-th cluster in the resulting KDE. We define this error next.

### 4.1 The local clustering error

Let \( \mathcal{H}_{opt} \) be the bandwidth estimated from the current sample distribution \( p_s(x) \), let \( p_1(x) = p_s(x; \pi_j) \) be a sub-mixture (17) of \( p_s(x) \) and let \( p_0(x) \) be the single-component approximation (18) of that sub-mixture. We define the local clustering error as the distance

\[
\hat{E}(p_1(x), \mathcal{H}_{opt}) = D(p_{1\text{KDE}}(x), p_{0\text{KDE}}(x)),
\]

(20)

between the corresponding KDEs

\[
p_{1\text{KDE}}(x) = p_1(x) * \phi_{\mathcal{H}_{opt}}(x)
\]

\[
p_{0\text{KDE}}(x) = p_0(x) * \phi_{\mathcal{H}_{opt}}(x).
\]

In particular, we can quantify the distance between distributions using the Hellinger distance [22], which is defined as

\[
D^2(p_{1\text{KDE}}(x), p_{0\text{KDE}}(x)) = \frac{1}{2} \int \left( p_{1\text{KDE}}(x)^{\frac{1}{2}} - p_{0\text{KDE}}(x)^{\frac{1}{2}} \right)^2 dx.
\]

(21)

Note that, while the Hellinger distance is a proper metric between distributions and is bounded to interval \([0, 1]\) (see [22]), it cannot be calculated analytically for the mixture models. We therefore calculate its approximation using the unscented transform [16]. For convenience, the derivation is given in the Appendix A.

### 4.2 Compression by hierarchical error minimization

In principle, the global minimization of \( E(\Xi(M)) \) (19) would require evaluation of all possible cluster assignments for the number of clusters ranging from one to \( N \), which becomes quickly computationally prohibitive. A significant reduction in complexity of the search can be obtained by a hierarchical approach to cluster discovery. Similar approaches have been previously successfully applied for speeding up the EM algorithm [19], online visual category discovery [12] and controlled data compression with Gaussian mixture models [11].

In our implementation, the hierarchical clustering proceeds as follows. We start by splitting the entire sample distribution \( p_s(x) \) into two sub-mixtures using the Goldberger’s [11] K-means algorithm for mixture models\(^1\) with \( K = 2 \). Each sub-mixture is approximated by a single Gaussian and the sub-mixture which yields the largest local error \( \hat{E}(p_s(x; \pi_j), \mathcal{H}_{opt}) \) is further split into two sub-mixtures. This process is recursively continued until the largest local error is sufficiently small and the condition \( E(\Xi(M)) \leq D_{th} \) in (19) fulfilled. This approach generates a binary tree with \( M \) leaves among the components of the sample distribution \( p_s(x) \), in which the leaves of the tree represent the clustering assignments \( \Xi(M) = \{ \pi_j \}_{j=1:M} \). Once the clustering \( \Xi(M) \) is found, the compressed sample distribution \( \hat{p}_s(x) \) (16) is calculated using (17) and (18).

Recall that we keep track of a detailed model for each component in the sample distribution. The detailed model \( \hat{q}_j(x) \) of the \( j \)-th component in the compressed model \( \hat{p}_s(x) \) is calculated as follows. If the set \( \pi_j \) contains only a single index, i.e., \( \pi_j = \{ i \} \), then the \( j \)-th component of the compressed sample distribution is equal to the \( i \)-th component in the original sample distribution and therefore the detailed model remains unchanged, i.e., \( \hat{q}_j(x) = q_i(x) \). On the other hand, if \( \pi_j \) contains multiple indexes, then the detailed models corresponding to these indexes are first concatenated into a single extended mixture model

\[
\hat{q}_{j_ext}(x) = \sum_{i \in \pi_j} q_i(x).
\]

(22)

Then the required two-component detailed model \( \hat{q}_j(x) \) is generated by splitting \( \hat{q}_{j_ext}(x) \) into two sub-mixtures again using the Goldberger’s K-means and each sub-mixture is approximated by a single Gaussian using (18).

### 4.3 Revitalizing the sample distribution

The compression identifies and compresses those clusters of components whose compression does not introduce a significant error into the KDE with the bandwidth \( \mathcal{H}_{opt} \) estimated at the time of compression. However, during online operation, new samples arrive, the sample distribution and \( \mathcal{H}_{opt} \) change, and so does the estimated KDE. Therefore, a compression which may be valid for a KDE at some point in time, may become invalid later on. Such an event can be detected through inspection of the detailed model of each component.

\(^1\)Note that to avoid the singularities associated with the components in the sample distribution with zero covariance, the K-means algorithm is carried out on the corresponding KDE.
in the sample distribution $p_s(x)$. Particularly, the composite error (19) of each component in the sample distribution model can be evaluated against its detailed model to verify whether the compression error is still low enough under the current KDE, i.e.,

$$E_{\text{det}} = \max_{i=1:N} \hat{E}(q_i(x), H_{\text{opt}}) ; \quad E_{\text{det}} \leq D_{\text{th}}.$$  \hspace{1cm} (23)

Those components in $p_s(x)$ for which $\hat{E}(q_i(x), H_{\text{opt}}) > D_{\text{th}}$ are removed from the sample distribution and replaced by the two components of their detailed model. A detailed model is then created for each of the new components. For example, let $w_i \tilde{\phi}_{N_i} (x - \mu_i)$ be one of the new components. If the determinant of $\Sigma_i$ is zero, then this component corresponds to a single datum and therefore its detailed model is just the component itself. However, in case the determinant is nonzero, it means that the component has been generated through clustering of several detailed models in the previous compression steps. Its detailed model is then initialized by splitting $\tilde{\phi}_{N_i} (x - \mu_i)$ along its principal axis into a two-component mixture, whose first two moments match those of the original component. More precisely, let

$$UDU^T = \Sigma_i$$

be a singular value decomposition of $\Sigma_i$ with eigenvectors and eigenvalues ordered by the descending eigenvalues. Then the new detailed mixture model is defined as

$$q_i(x) = \sum_{k=1}^{2} \alpha_k \tilde{\phi}_{N_i} (x - \mu_k), \quad \mu_1 = FM + \mu_i ; \quad \mu_2 = FM - \mu_i,$$

$$\Sigma_k = FCF^T ; \quad \alpha_k = \frac{1}{2} w_i,$$

where $C = \text{diag}([3/4, 0_{1 \times (d-1)}])$, $M = [0.5, 0_{1 \times (d-1)}]^T$, $F = \sqrt{D}$ and $0_{1 \times (d-1)}$ is an all-ones row vector of length $(d-1)$. The entire compression procedure along with the revitalization routine is summarized in the Algorithm 1.

5 Online Kernel Density Estimation

Let us denote the model of the samples observed up to time-step $(t-1)$ as

$$S_{\text{model}}(t-1) = \{p_{s(t-1)}(x), \{q_{i(t-1)}(x)\}_{i=1:M_{t-1}}\}, \quad (25)$$

where $p_{s(t-1)}$ is a $M_{t-1}$-component sample distribution,

$$p_{s(t-1)}(x) = \sum_{i=1}^{M_{t-1}} \alpha_i \tilde{\phi}_{N_i}(x - \mu_i). \quad (26)$$

Let $N_{t-1}$ denote the number of all observed samples up to time-step $(t-1)$. At time-step $t$ we observe a sample $x_t$, and reestimate the sample model $S_{\text{model}}(t) = \{\tilde{p}_{s(t)}(x), \{\tilde{q}_{i(t)}(x)\}_{i=1:M_t}\}$ (and hence the KDE) in the following steps.

**Step 1: Update the sample model.** The number of observed samples is augmented, $N_t = N_{t-1} + 1$. Assuming all observations are equally probable, the sample distribution is updated by the new observation as

$$\tilde{p}_{s(t)}(x) = \frac{N_{t-1}}{N_t} p_{s(t-1)}(x) + \frac{1}{N_t} \phi_0(x - x_t). \quad (27)$$


\[\text{Note that } (\cdot) \text{ denotes the updated model before the compression.}\]

**Algorithm 1:** Compression of the sample model.

**Input:**

$$S_{\text{model}} = \{p_{s(t)}(x), \{\tilde{q}_{i(t)}(x)\}_{i=1:M_t}\} \quad \ldots \quad \text{the } M\text{-component sample model.}$$

$$H_{\text{opt}} \ldots \text{the current optimal bandwidth.} \quad \text{D}_{\text{th}} \ldots \text{the maximal allowed local compression error.}$$

**Output:**

$$S_{\text{model}} = \{\tilde{p}_{s(t)}(x), \{\tilde{q}_{i(t)}(x)\}_{j=1:M} \} \ldots \text{the compressed } M\text{-component sample model.}$$

**Procedure:**

1: Revitalize each $i$-th component in $\tilde{p}_{s(t)}(x)$ for which $\hat{E}(\tilde{q}_{i(t)}(x), H_{\text{opt}}) > D_{\text{th}}$ according to Section 4.3 and replace the sample model with the $N$-component revitalized model:

$$S_{\text{model}} \leftarrow \{p_{s(t)}(x), \{q_{i(t)}(x)\}_{i=1:N}\}.$$  \hspace{1cm} (28)

2: Initialize the cluster set: $\hat{\Xi}(M) = \{\pi_1\}, \pi_1 = \{1, \ldots, N\}, M = 1$

3: while $D_{\text{th}} < \max_{\pi_j \in \hat{\Xi}(M)} \hat{E}(p_{s(t)}(x; \pi_j))$ do

4: Select the cluster with the maximum local error: $\pi_j = \arg \max_{\pi_j \in \hat{\Xi}(M)} \hat{E}(p_{s(t)}(x; \pi_j))$

5: Split the sub-mixture $p_{s(t)}(x; \pi_j)$ into two sets using the Goldberger’s $K$-means:

$$\pi_j \longrightarrow \{\pi_{j1}, \pi_{j2}\}.$$  \hspace{1cm} (29)

6: Update the cluster set: $M \leftarrow M + 1$, $\hat{\Xi}(M) \leftarrow \{\{\hat{\Xi}(M) \setminus \pi_j\}, \pi_{j1}, \pi_{j2}\}.$

7: end while

8: Regroup the components of $p_{s(t)}$ according to clustering $\hat{\Xi}(M)$ and construct the compressed sample model $\tilde{p}_{s(t)}(x)$.

9: For each $j$-th component in $\tilde{p}_{s(t)}$ create its detailed model $\tilde{q}_{\hat{M}_t}(x)$ from the reference detailed models $\{q_{i(t)}(x)\}_{i=1:N}$ according to the clustering $\hat{\Xi}(M)$.

The detailed model $\tilde{q}_{\hat{M}_t}(x) = \phi_0(x - x_t)$ corresponding to $x_t$ is added to the existing set of detailed models

$$\{\tilde{q}_{i(t)}(x)\}_{i=1:M_{t-1}} \leftarrow \{\tilde{q}_{i(t)}(x)\}_{i=1:M_{t-1}} \cup \tilde{q}_{\hat{M}_t}(x). \quad (28)$$

Thus yielding an updated sample model

$$S_{\text{model}}(t) = \{\tilde{p}_{s(t)}(x), \{\tilde{q}_{i(t)}(x)\}_{i=1:M_t}\}. \quad (29)$$

**Step 2: Reestimate the bandwidth.** The empirical co-variance of the observed samples $\hat{\Sigma}_{\text{amp}}$ is calculated by approximating $\hat{p}_{s(t)}(x)$ by a single Gaussian using the moment matching (18). The new optimal bandwidth is then calculated (Section 3) as

$$H_t = |d(4\pi)^{d/2} N_t \hat{R}(p, F, G)|^{\frac{1}{d}}, \quad F = \hat{\Sigma}_{\text{amp}}[\hat{\Sigma}_{\text{amp}}]^{-1/d}, \quad G = \hat{\Sigma}_{\text{amp}} \left( \frac{4}{(2 + d) N_t} \right)^{1/2}, \quad (30)$$

and $\hat{R}(p, F, G)$ is defined in (11).

**Step 3: Refine and compress the model.** After the current optimal bandwidth $H_t$ has been calculated, the sample model $S_{\text{model}}(t)$ is refined and compressed by minimizing the composite error (19), using the Algorithm 1, into

$$S_{\text{model}}(t) = \{p_{s(t)}(x), \{\tilde{q}_{i(t)}(x)\}_{i=1:M_t}\}. \quad (31)$$
In our implementation, the compression is called after some threshold on number of components $M_{thc}$ has been exceeded. Note that this threshold does not determine the number of components in the final model, but rather the frequency at which the compression is called. To avoid too frequent calls to compression, the threshold is also allowed to vary during the online operation using a simple hysteresis rule: If the number of components $M_t$ still exceeds $M_{thc}$ after the compression, then the threshold increases $M_{thc} ← 1.5 M_{thc}$, otherwise, if $M_t < \frac{1}{2} M_{thc}$, then it decreases $M_{thc} ← 0.6 M_{thc}$.

After the three steps of the online update have finished, the sample distribution is a $M_t$-component mixture model
\begin{equation}
 p_{th}(t) = \sum_{i=1}^{M_t} \alpha_i \phi_{\Sigma_i}(x - \mu_i),
\end{equation}
and the current KDE is calculated according to (3):
\begin{align}
 p_{\text{KDE}}(x) &= p_{th}(t)(x) * \phi_H(x) \\
 &= \sum_{i=1}^{M_t} \alpha_i \phi_{\Sigma_i + H}(x - \mu_i).
\end{align}

Since oKDE may be initialized using smaller number of samples than the sample dimensionality, the updates may suffer from singular covariances. To avoid these, the oKDE and the new data-point are first projected into a subspace using PCA, the updates are carried out there, and then the oKDE is backprojected into the original space.

6 Experimental study

6.1 Estimation of probability density functions

A significant difference between the online and batch algorithms is that the batch algorithms have access to all data, while the online algorithms discard the data and retain only their models. These models thus have to retain enough information to be able to successfully update when the new observations arrive. We have therefore compared the performance of the oKDE with several batch approaches. The first three were batch state-of-the-art KDE methods: Hall’s et al. [13] plug-in (implementation [15]), Murillo’s et. al. [20] cross validation and Girolami’s et. al. [10] reduced-set-density estimator. The fourth method was the batch EM algorithm with integrated regularization and model selection [9]. We have compared the oKDE also to the adaptive mixtures (AM) [23], which is essentially an online EM algorithm for Gaussian mixture models with automatic component-allocation mechanism. The parameters of the adaptive mixtures were set as in [6].

The experiment involved estimation of two separate distributions. The first was a 2D sinusoidal distribution defined by
\[ x = [a, \sin(3a) + w]^T ; a = 4(t - 1/2) ; w \sim \phi_{\sigma_w}() \]
with $\sigma_w = 0.2^2$ (Figure 3a). The second distribution was a 3D spiral distribution defined by the following model
\[ x = [(13 - \frac{1}{2} t) \cos(t), -(13 - \frac{1}{2} t) \sin(t), t]^T + w \]
\[ w \sim \phi_{\Sigma_w}() ; t \sim U(0,14), \]
\[ \Sigma_w = \text{diag}\{\frac{1}{2}, \frac{1}{2}, 1\} \], and $U(1,14)$ is a uniform distribution constrained to interval $[0,14]$ (Figure 3b). A set of 1000 samples was generated from the model distribution. Batch models were estimated from all 1000 samples. In the online approaches, the first ten samples were used for initialization and the rest were added one at a time. The predictive performance of the models was evaluated by the average log-likelihood of additionally drawn 50,000 observations. The experiment was repeated 20 times. The performance of the batch an online methods after observing 50, 100, 400 and 1000 samples is shown in Table 1 and Table 2. Among the batch approaches, the CV performed on average best for 50 samples, but with increasing number of samples, the batch EM algorithm was superior. Among the online approaches, the oKDE consistently outperformed the AM in terms of the log-likelihood even for the values $D_{th} = 0.05$. Note also, that it also required less components to achieve better performance. The oKDE also consistently outperformed the Hall’s batch KDE in terms of the log-likelihood while requiring significantly smaller number of components – e.g., after observing a thousand samples the oKDE required only roughly 15 percent of number of components compared to the Hall’s batch KDE.

![Figure 3: Samples from the sinusoidal (a) and the spiral distribution (b) along with the models obtained by the oKDE_{0.02}.

6.2 Construction of an online classifier

We have compared the classification performance of the oKDE with three batch KDEs: the CV batch KDE [20], the reduced-set density estimator [10] initialized by the CV and the Hall’s batch KDE [13]. For the baseline classification, we have applied a multiclass SVM with an rbf kernel [5]. We have used $D_{th} = 0.05$ in the oKDE. The methods were compared on a set of public classification problems [2] (Table 3). The classification performance of the KDE-based methods was tested using a simple Bayesian criterion
\begin{equation}
 \tilde{y} = \arg \max_i p_{\text{KDE}}(x|c_i).
\end{equation}

The parameter for the SVM kernel was determined separately in each experiment via cross validation on the training data set. The classification experiment was conducted via four-fold cross validation and the results are shown in Table 3. On average, the SVM and CV method produced best classification. The oKDE outperformed RSDE and slightly Hall’s KDE batch method, and produced a comparable classification to the SVM and the CV. An important observation is that the oKDE outperformed, or produced comparable performance, to the batch methods, even though the oKDE was
constructed by observing only a single sample at a time. In contrast, the SVM and batch KDEs optimized their structure by having access to all the samples. A further thing to note is that, with the exception of the Pima dataset, the oKDE’s classification performance quite closely matched that of an SVM, even though the oKDE is in its nature reconstructive, while the SVM optimizes its structure to maximize discrimination.

7 Conclusion

We have proposed an approach for a kernel density estimation which can be applied in online operation (oKDE). The central point of the proposed scheme is that it maintains a compressed model of the observed samples and uses this model to compute the kernel density estimate of the underlying distribution. The oKDE automatically maintains its complexity thorough the process of compression and refinement. Experiments have shown that, in terms of probability density estimation, the oKDE produces comparable results to the state-of-the-art batch approaches and outperforms the online EM algorithm. In the experiment with classification problems the oKDE outperformed some batch KDEs and produced comparable classification to the state-of-the-art batch KDE and the support vector machine. While the proposed oKDE is a contribution to the literature on kernel density estimation as such, parts of our approach can contribute to solutions of other problems as well. The proposed unscented Hellinger distance may be used, for example, as a general metric in applications where one needs to compare mixtures of Gaussians (e.g., [11]). Recently, an approximate probability density estimator was proposed for visual tracking in [14]. The estimator is based on KDE, however, the kernel bandwidth is user predefined. Our bandwidth selection rule can be directly applied to that estimator to provide means of automatic bandwidth selection. In our future work we will apply the oKDE to estimation of non-stationary distributions and consider extensions to enhance its discrimination performance, which, we expect, will lead to online KDE-based probabilistic discriminative models.

A The unscented Hellinger distance

The unscented transform is a special case of a Gaussian quadrature, which, similarly to Monte Carlo integration, relies on evaluating integrals using carefully placed points, called the sigma points, over the support of the integral. Therefore, as in Monte Carlo integration [25], we define an importance distribution $p_0(x) = \gamma (p_1(x) + p_2(x))$, which contains the support of both, $p_1(x)$ as well as $p_2(x)$, with $\gamma$ set such that $\int p_0(x)/dx = 1$. In our case, $p_0(x)$ is a Gaussian mixture model of a form $p_0(x) = \sum_{i=1}^{N} \omega_i \phi_{\Sigma_i}(x - x_i)$, and we rewrite (21) into

$$D^2(p_1, p_2) = \frac{1}{2} \int g(x)p_0(x)dx =$$

---

Table 1: The average negative log-likelihood ($\mathcal{L}$) and the number of components in the model ($N_{cmp}$) w.r.t. the number of observed samples for the experiment with the sinusoidal distribution.

<table>
<thead>
<tr>
<th>Batch methods</th>
<th>50 samples</th>
<th>100 samples</th>
<th>400 samples</th>
<th>1000 samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>CV</td>
<td>$1.70 \pm 0.10$</td>
<td>$1.52 \pm 0.06$</td>
<td>$1.37 \pm 0.02$</td>
<td>$1.37 \pm 0.01$</td>
</tr>
<tr>
<td>Hall</td>
<td>$2.39 \pm 0.04$</td>
<td>$2.28 \pm 0.04$</td>
<td>$2.09 \pm 0.02$</td>
<td>$1.98 \pm 0.01$</td>
</tr>
<tr>
<td>RSDE</td>
<td>$1.88 \pm 0.13$</td>
<td>$1.63 \pm 0.07$</td>
<td>$1.38 \pm 0.02$</td>
<td>$1.32 \pm 0.02$</td>
</tr>
<tr>
<td>EM</td>
<td>$1.90 \pm 0.13$</td>
<td>$1.67 \pm 0.11$</td>
<td>$8 \pm 1.3$</td>
<td>$1.34 \pm 0.03$</td>
</tr>
</tbody>
</table>

Table 2: The average negative log-likelihood $\mathcal{L}$ and the number of components in the model ($N_{cmp}$) w.r.t. the number of observed samples for the experiment with the 3D spiral.

<table>
<thead>
<tr>
<th>Batch methods</th>
<th>50 samples</th>
<th>100 samples</th>
<th>400 samples</th>
<th>1000 samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>CV</td>
<td>$8.05 \pm 0.45$</td>
<td>$7.39 \pm 0.29$</td>
<td>$6.76 \pm 0.02$</td>
<td>$6.62 \pm 0.01$</td>
</tr>
<tr>
<td>Hall</td>
<td>$8.12 \pm 0.33$</td>
<td>$7.61 \pm 0.12$</td>
<td>$7.15 \pm 0.02$</td>
<td>$6.95 \pm 0.01$</td>
</tr>
<tr>
<td>RSDE</td>
<td>$8.83 \pm 0.77$</td>
<td>$7.87 \pm 0.47$</td>
<td>$7.38 \pm 0.06$</td>
<td>$6.82 \pm 0.02$</td>
</tr>
<tr>
<td>EM</td>
<td>$10.36 \pm 1.37$</td>
<td>$9.55 \pm 1.07$</td>
<td>$7.05 \pm 0.11$</td>
<td>$6.58 \pm 0.02$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Online methods</th>
<th>50 samples</th>
<th>100 samples</th>
<th>400 samples</th>
<th>1000 samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>AM</td>
<td>$2.17 \pm 0.10$</td>
<td>$2.02 \pm 0.09$</td>
<td>$1.83 \pm 0.10$</td>
<td>$1.74 \pm 0.11$</td>
</tr>
<tr>
<td>oKDE0.01</td>
<td>$1.95 \pm 0.07$</td>
<td>$1.80 \pm 0.04$</td>
<td>$20 \pm 1.5$</td>
<td>$27 \pm 2.3$</td>
</tr>
<tr>
<td>oKDE0.02</td>
<td>$2.00 \pm 0.08$</td>
<td>$1.80 \pm 0.04$</td>
<td>$14 \pm 1.2$</td>
<td>$15 \pm 1.7$</td>
</tr>
<tr>
<td>oKDE0.04</td>
<td>$2.00 \pm 0.08$</td>
<td>$1.80 \pm 0.04$</td>
<td>$9 \pm 1.0$</td>
<td>$1.72 \pm 0.04$</td>
</tr>
<tr>
<td>oKDE0.05</td>
<td>$2.01 \pm 0.06$</td>
<td>$1.91 \pm 0.05$</td>
<td>$7 \pm 1.2$</td>
<td>$1.78 \pm 0.04$</td>
</tr>
</tbody>
</table>
where we have defined \( g(x) = \frac{(\sqrt{\mathbf{p}(x|x)} - \sqrt{\mathbf{p}(x)})^2}{\mathbf{p}(x)} \). Note that the integrals in (35) are simply expectations over a non-linearly transformed Gaussian random variable \( \mathbf{X} \), and therefore admit to the unscented transform. According to [16] we then have

\[
D^2(p_1, p_2) \approx \frac{1}{2} \sum_{i=1}^{N} w_i \int g(\mathbf{X}_i) d\mathbf{X}_i,
\]

(36)

where \( \mathbf{X}_i \) are weighted sets of sigma points corresponding to the \( i \)-th Gaussian \( \mathbf{p}(x|x) \), and are defined as

\[
\begin{align*}
(0) \mathbf{X}_i &= \mathbf{x}_i; \\
(1) \mathbf{X}_i &= \mathbf{x}_i + s_j \sqrt{1 + \kappa}(\sqrt{\Sigma_j}); \\
(2) \mathbf{X}_i &= \frac{\kappa}{2(1 + \kappa)}; s_j = \begin{cases} 
1 & j \leq d \\
-1 & \text{otherwise}
\end{cases}
\end{align*}
\]

with \( \kappa = \max([0, m - d]) \), and \( (\sqrt{\Sigma_j}) \) is the \( j \)-th column of the matrix square root of \( \Sigma_j \). Concretely, let \( \mathbf{U} \mathbf{D} \mathbf{U}^T \) be a singular value decomposition of covariance matrix \( \Sigma_j \), such that \( \mathbf{U} = [U_1, \ldots, U_d] \) and \( \mathbf{D} = \text{diag} \{ \lambda_1, \ldots, \lambda_d \} \), then

\[
(\sqrt{\Sigma_j}) = \sqrt{\mathbf{U} \lambda_j \mathbf{U}^T}.
\]

In line with the discussion on the properties of the unscented transform in [16], we set the parameter \( m \) to \( m = 3 \).

Acknowledgement

This research has been supported in part by: Research program P2-0214 (RS), Research program P2-0095 (RS), TP-MIR SOVZO (Ministry of Defense of Republic of Slovenia), and EU FP7-ICT215181-IP project CogX.

References

Representing the Surface of Objects by Combinatorial Pyramids*

Esther Antúnez¹, Helena Molina-Abril¹², and Walter G. Kropatsch¹

¹Pattern Recognition and Image Processing Group
Vienna University of Technology
{eortiz,habril,krw}@prip.tuwien.ac.at

²Computational Topology and Applied Mathematics Group
University of Seville
habril@us.es

Abstract This paper introduces a new method to represent the surface of objects using two dimensional combinatorial maps. The classical definition of two dimensional combinatorial maps is extended here by adding a “back face” that corresponds to the non–visible part of the object. As a first step, every object in the scene is extracted in one image pyramid, where levels are related by means of coordinates. Finally, an algorithm to complete the description of the surface is presented. Such representation is translation, rotation and scale invariant. It will allow to update information about movements and parts of the object that become visible, reducing the complexity and the computational time.

1 Introduction

A large amount of computational resources is required to deal with image and video data. The main advantage of hierarchical structures is the rapid computation of a global information in a recursive manner. Thus, an hierarchical structure (like image pyramids) might be the answer to the time and space complexity in computer vision systems.

From a single image frame, the visible front surface of an object can be represented in terms of combinatorial maps or graphs. This extracted map (or graph) embeds the topological structure of the object, and describes in an efficient way its visible features. In order to efficiently represent the object in a hierarchical manner, a combinatorial pyramid [2] (or a dual graph pyramid [13]) can be constructed above the initial map (or graph).

The benefit of using combinatorial maps, is that they combine the advantages of dual graphs with an explicit orientation of the boundary segments of the embedded object. Moreover, the combinatorial map formalism is defined for any dimensions.

Nevertheless the use of such a complex structure implies some drawbacks. The extension of combinatorial maps to higher dimensions (3D, 4D, …, nD) is feasible, but at the price of much higher memory costs [7].

In addition, in many applications in computer vision due to the fact that inner parts are usually not visible, what we really need to deal with is the surface of the object.

The goal of this paper is to extend the classical two dimensional combinatorial maps, in such a way that they represent two dimensional manifolds. This new representation is translation, rotation and scale invariant.

The inclusion of this new structure in the environment of image pyramids will be very useful when for example a three dimensional object is moving in a video sequence, and at each moment, some parts are occluded or partially visible.

The paper is structured as follows: In Section 2 we introduce the concepts of combinatorial maps and combinatorial pyramids. In Section 3 the description of an object of the initial image using a single pyramid is outlined. In Section 4 we present the procedure in which the invisible part is added to the previous representation in order to complete the object surface. The paper concludes in Section 5.

---

*Partially supported by the Austrian Science Fund under grants P20134-N13 and P18716-N13, the “Computational Topology and Applied Mathematics” PAICYT research project FQM–296, the “Andalusian research projects” PO6-TIC-02268 and P07-TIC-03106 and the “Science and Innovation Ministry” project MTM2009-12716
2 Recall on combinatorial pyramids

Image pyramids are a stack of images with decreasing resolutions [4] (see Figure 1). Such pyramids present many interesting properties within the Image Processing and Analysis framework such as [1]: Reducing the influence of noise by eliminating less important details in lower-resolution versions of the image, making the processing independent of the resolution of the regions of interest in the image, converting global features to local ones, reducing computational costs for many computations, etc.

The construction of the pyramid hierarchy follows the philosophy to reduce the amount of data at each higher level of the hierarchy by a reduction factor \( \lambda > 1 \) while preserving important topological properties like connectivity and inclusion. Every cell in level \( k \) is linked with cells on level directly below \( k - 1 \). Those cells are called its children and cells on the level directly above \( k + 1 \) its parent(s). The highest level of the pyramid is called its apex.

There exist different topological representations for structured objects that can be used in the hierarchical framework of image pyramids. These representations are dual graphs, combinatorial maps and generalized maps [11]. Image pyramids that contain these structures are called topological pyramids [12].

Combinatorial maps define a general framework, which allows to encode any subdivision of \( nD \) topological spaces orientable or non-orientable with or without boundaries. They were introduced in [6], at first as a planar graph model, and extended in [14] in dimension \( n \) to represent orientable or not-orientable quasi-manifolds.

Informally speaking, a combinatorial map is a mathematical model describing the subdivision of a space, and encoding all the cells of the subdivision and all the incidence and adjacency relations between the different cells. In this way, the topology of the space is fully described.

A more formal definition, describes a \( n \) dimensional combinatorial map as a \((n+1)\)-tuple \( M = (D, \beta_1, \beta_2, ..., \beta_n)\) such that \( D \) is the set of abstract elements called darts, \( \beta_1 \) is a permutation on \( D \) and the other \( \beta_i \) are involutions on \( D \). An involution is a permutation whose cycle has the length of two or less. In the case of 2D, combinatorial maps may be defined with the triplet \( G = (D, \alpha, \sigma) \), where \( D \) is the set of darts, \( \sigma \) is a permutation in \( D \) encoding the set of darts encountered when turning (counter) clockwise around a vertex, and \( \alpha \) is an involution in \( D \) connecting two darts belonging to the same edge:

\[
\forall d \in D, \alpha^2(d) = d \tag{1}
\]

Figure 2 shows an example of a combinatorial map. In Table 1 the values of \( \alpha \) and \( \sigma \) for such a combinatorial map can be found. In this example counter clockwise (CCW) orientation has been chosen for \( \sigma \). Note that every border cell is adjacent to the background/infinite region (\( Bg \)).

3 Extracting objects from a combinatorial pyramid

Given an image pyramid, we would like to describe a single object contained on it. Let us recall that our aim is to obtain a complete representation of the object’s surface. This representation must be translation, rotation and scale invariant.

The first step will consist in adding coordinates to the structure. The second step is to extract the pyramid that describes the object from the pyramid of the complete picture.

3.1 Relation between pyramid levels

In order to add coordinates to the structure, each cell (faces in case we are considering the dual map, or vertices in case we consider the primal) in the base level of the pyramid is attributed with the coordinates of the corresponding image pixels. For higher levels the coordinates of each cell are computed by inheritance from the surviving child to the parent. But storing all this information along the structure, will cause a big amount of redundant data. To avoid this fact, we compute and store at each level only the difference between the coordinates in one level with the coordinates in the level above, following the idea of the Laplacian pyramid [5]. This difference is called the child’s correction vector:

\[
d(c) = p(parent(c)) - p(c), \tag{2}
\]

where \( d(c) \) is the child’s correction vector, \( p(c) \) are the child’s coordinates and \( p(parent(c)) \) are the parent’s
coordinates.

The original position of each cell can be reconstructed accurately by adding all the correction vectors from the apex. When the object is translated, such translation is applied to the coordinates of the apex. After that, the coordinates of the rest of the cells are reconstructed in the new position using the correction vectors in a top-down process, making the object representation invariant to translation.

In order to make our representation rotation and scale invariant, the orientation of the object is added as an attribute in the apex cells [9] as well as a scale factor. Such information replaces the previous one stored in the correction vectors. The orientation correction vector is called \( \theta(c) \).

\[
\theta(c) = o(\text{parent}(c)) - o(c),
\]

where \( \theta(c) \) is the child’s orientation correction vector, \( o(c) \) is the child’s orientation and \( o(\text{parent}(c)) \) is the parent’s orientation.

Once we have performed this process, all the information about position and orientation of the object is concentrated in the apex. Given the position and orientation in the top level the coordinates of each cell can be obtained by means of the following formulas:

\[
\begin{align*}
x_c &= x_p + \lambda \cdot r(c) \cdot \cos(o_p + \theta(c)) \\
y_c &= y_p + \lambda \cdot r(c) \cdot \sin(o_p + \theta(c)),
\end{align*}
\]

being \( x_c, y_c, x_p, y_p \) the coordinates of the child and the parent respectively, \( \lambda \) the scale factor, \( r(c) \) the child-parent euclidean distance, \( o_p \) the parent orientation and \( \theta(c) \) the child’s orientation correction vector.

One of the advantages of this representation is that for example in the analysis of video sequences it is not necessary to compute one pyramid for each frame; it is enough to apply all the transformations in the apex and then the computation of the correction vectors allows an accurate reconstruction of the whole pyramid in a top-down process.

3.2 Cutting out objects

In order to separate the data structure for each (moving) object, we would like to have one pyramid per object in the image and one for the background, keeping the connection between them by the coordinates previously added. This representation will allow to update movements and properties in an independent way.

In a sequence of video frames it may happen that some objects in the scene move and others remain in the same position. Having one pyramid per object allows to update only the information corresponding to the moving objects. This fact decreases the number of updates in the structure from frame to frame and the processing time with the advantage that the connectivity between the objects and the background is always preserved by means of the stored coordinates in the apex of each pyramid.

In order to extract the object from the image pyramid, we first need to identify it. Several segmentation methods can be used for that purpose [10, 15], allowing to distinguish cells that belong to the object. Once those cells have been determined, the corresponding combinatorial map that represents the desired object at each pyramid level is extracted. Every cell that is not part of the object, will be considered as part of the background. Consequently, the border cells of the object will be adjacent to the background. Removal operations need to be performed on the initial map in order to delete parts that do not belong to the object.

At each level a new combinatorial map that only corresponds to the desired object is obtained by adding to a removal kernel [3] the darts that do not belong to the object. After that we also have to update the information that relates two levels in the pyramid.

Our method may also be useful for articulated objects since it is possible to concentrate the information of each rigid part in the apex. The transformations can be applied to the apex of the moving part, and at the end only the part of the structure that has moved will be updated.

3.3 Preservation of connectivity

A significant advantage of using topological pyramids is that the connectivity is always preserved. This is not always true for other image processing tools (e.g. Photoshop). For instance, when some rotations are applied, these tools have to use some kind of interpolation or re-sampling when the coordinates are not integers. Figure 5 shows an example where a thin line is rotated. Figure 5b contains the result obtained using Photoshop. The red stars mark the new rounded coordinates of each point and the black squares are the connected line (or in a thicker line if bilinear interpolation or anti-aliasing is applied) (see Figure 5b). Using our approach, although the position of the points change with the rotation, the relation between the cells in the graph remains the same since the edges of the graph always connect the same vertices. The black stars in Figure 5c point out that there is a connection between those points in the graph. In our case it is possible to apply again the same rotation in the other direction.

| dart | 1  | -1 | 2  | -2 | 3  | -3 | 4  | -4 | 5  | -5 | 6  | -6 | 7  | -7 | 8  | -8 | 9  | -9 | 10 | -10 | 11 | -11 |
|------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----|----|-----|
| \( \alpha \) | -1 | 1  | -2 | 2  | -3 | 3  | -4 | 4  | -5 | 5  | -6 | 6  | -7 | 7  | -8 | 8  | -9 | 9  | -10 | 10 | -11 | 11 |
| \( \sigma \)  | -11 | 10 | 5  | -1 | 7  | -8 | -6 | 3  | -3 | 10 | -7 | 1  | -4 | -5 | 11 | 9  | 8  | -9 | -2  | 6  | 2   | 4  |

Table 1: Values of \( \alpha \) and \( \sigma \) for the combinatorial map in Figure 2
Representing the Surface of Objects by Combinatorial Pyramids

Figure 3: Object surface completion process

Figure 4: Object surface completion example

Table 2: Values of $\sigma$ and $\alpha$ of the complete combinatorial map in Figure 4
and obtain the original line while in other cases it is impossible since the line has been destroyed.

![a) a thin line rotated b) by Photoshop c) as attributed graph](image)

**Figure 5:** 50° rotation of a thin line

4 Object surface completion

Once we have the visible part (front face) of the object described using a combinatorial map, we would like to complete it with the non visible part of the object surface. To do this, we use one or more invisible surface patches in order to completely cover the surface of the volumetric object.

The main idea is to create a combinatorial map that is identical to the initial one, and that will conform the back face. These two maps (front and back) will be glued together, sharing the border cells. Therefore, only darts that are not adjacent to the background, are duplicated and conform the back face. Darts that are adjacent to the background (border of the object) will be common to the front and back faces (see Figure 3). In order to correctly cover every face of the object surface, we must use counter orientation for front and back maps. Thus, in case we use for example the front maps, the back ones will be defined in terms of \( \sigma \).

The pseudo-code of the algorithm applied to every level of the pyramid is presented below. In this pseudo-code \( Bg \) represents the background face. The function \( B \) relates every dart with its corresponding one at the back face and vice versa.

Given a combinatorial map \( G = (D, \alpha, \sigma_{ccw}) \), where \( G \) contains \( n \) edges

**Step 1:**
for every edge of the map, formed by the darts \( d \) and \( -d \) (where \( d = \alpha(-d) \)) do
if \( d \) is not in \( Bg \) and \( -d \) is not in \( Bg \) then
    Add two new back darts \( b \) and \( -b \) to \( D \)
    \( \alpha(b) = -b \), \( \alpha(-b) = b \)
    \( \sigma(b) = \sigma_{ccw}(d) \), \( \sigma(-b) = \sigma_{ccw}(-d) \)
    \( B(d) = b \), \( B(-d) = -b \)
else
    \( B(d) = d \), \( B(-d) = -d \)
end if
end for
**Step 2:**
for every dart \( t \in D \) do
if \( B(t) = t \) and \( B(\sigma(t)) = \sigma(t) \) then
    \( \sigma(t) = B(\sigma_{ccw}(t)) \)
end if
end for

In the first step of the algorithm, we include the darts that conform the back face, and initialize their \( \sigma \) and \( \alpha \) values (see Figure 4). These darts are related with their correspondent front darts by the function \( B \). For instance, in Figure 4 \( B(3) = 9 \) and \( B(1) = 1 \). In the second step, \( \sigma \) must be updated, in order to be consistent with the newly added darts. In Table 2, the values of \( \sigma \) that have to be updated are marked in red.

After applying this algorithm each level of the pyramid is a map, presenting the closed surface of the object in multiple resolutions. The pyramid representation can cope with this structure and the same operations can be applied as in the case of an image. In order not to store so much information, the back face can be contracted to a single cell.

As new visible parts of the surface would reveal previously invisible parts, the object representation will be incrementally updated automatically from observing the target object in a video sequence. This requires the registration of the visible parts and the replacement of some invisible patches. For this purpose, some of the existing methods dealing with multi-view integration might be useful [8, 17].

When some hidden parts appear, the new topological structure will be added into the previous 2D manifold to obtain the updated object representation [9]. The new view could imply more complexity on the topology of the object (like when the handle of a cup becomes visible) or could also imply the simplification of the former topological structure (for example a twisted torus, can be perceived from some view points as a double torus, see Figure 6).

![Figure 6: Two different views of a twisted torus](image)

When all the observable parts of the object have been integrated in the object model, the topological structure of the target object is complete. This is the process defined as topological completion [16]. Due to the fact that only visible parts of the object are considered, parts that never become visible will not be included on its representation. Therefore the word “complete” is applied here in a non rigorous sense.
5 Conclusion

A new representation of object surfaces using two dimensional combinatorial maps has been presented in this paper. This representation is translation, rotation and scale invariant and preserves the topological properties of the visible parts of the object. In this way we reduce the complexity and the amount of stored data, that would be higher in case of using pure three dimensional structures.

Another major advantage is that the computation time in the analysis of video sequences is considerably reduced due to the reduction of updates from frame to frame since the movement of the objects modifies only one cell in the structure (the apex). This last advantage allows to deal with real time processing requirements.

As future work we plan to develop efficient methods to update this structure as new visible parts of the object become visible, and make experimentations and comparisons to other methods.

References

Taking in Shape: Detection and Tracking of Basic 3D Shapes in a Robotics Context

Richtsfeld Andreas, Mörwald Thomas, Zillich Michael, and Vincze Markus

Automation and Control Institute (ACIN), Vienna University of Technology

{arichtsfeld, moerwald, zillich, vincze}@acin.tuwien.ac.at

Abstract Segmenting and tracking generic objects in an unknown dynamic scene remains an elusive goal for computer vision. In this paper we tackle a simplified problem, namely detecting and tracking objects from a class of basic shapes (cuboids, cylinders, cones, spheres) in scenes containing a ground plane. We use perceptual grouping of edges to identify generic views of basic shapes, instantiate 3D wire-frame models assuming that objects initially rest on the ground and subsequently track these using a particle-filter based tracker. During tracking we augment the wire-frame models with surface texture which substantially increases the robustness of tracking with respect to background clutter as well as occlusions and lighting effects.

1 Introduction

Perceiving 3D shape plays a major role in computer vision applications for robotics tasks. Most obviously this is true for tasks involving manipulation of objects, be it grasping, simple pushing or just observing a human handling an object. While tracking based on 3D models of objects has a long history and many increasingly sophisticated methods exist, robustness and speed remain a challenge to this date. Also the question of how sufficiently accurate object models are acquired in the first place is often left open.

In this work we present a system that aims to close this gap between model acquisition from an unknown scene and robust tracking of models. We make several simplifying assumptions. Having in mind robotics applications we assume a robot mounted camera which allows us to at least assume the ground plane (or table plane) while the rest of the scene remains of course unknown. Furthermore we restrict our search to objects belonging to simple shape classes (cuboids, cylinders, cones and spheres) which are detected in generic views from edge images. Note that there are in principle no limits to shape complexity for the tracker once we have a model (in fact more “non-regular” shapes simplify the task for tracking) but that the difficulty of segmenting objects from edge images limits the scope of shapes we can handle.

Fig. 1 shows how we proceed from 2D images to tracked 3D objects.

The remainder of the paper is organized as follows. Section 2 reviews related work. Section 3 explains detection of shapes while their tracking is detailed in Section 4. Section 5 shows experimental results and Section 6 concludes the paper.

2 Related Work

Detection of geometric shapes based on perceptual grouping of edges is a well known topic in computer vision with an abundance of literature since the eighties. [1] introduced geons as a means of describing objects of arbitrary shape by components and [3] discuss the potential of geons in computer vision systems. [20] present a method based on joint statistical constraints defined on complex wavelet transforms to represent and detect geons. These representations are however not size invariant and a sliding window is needed to cover the...
whole image at different positions and scales. Moreover the test images are rather artificial with grey objects on a black background. Note that our proposed approach does not aim to detect all geons but is tailored to a subset.

[9] uses perceptual grouping of edge segments to reduce the complexity of detecting 3D models in edge images and shows impressive results on highly cluttered images. This however requires precise CAD-like 3D models given a-priori which we want to avoid. Approaches such as [13], [6] and [14] use groups of edge fragments to detect learned classes of shapes and show impressive results on databases. Our models differ in that they are only topological models of generic views of basic shapes such as an arrangement of lines and ellipses forming a cylinder, where the limited number of these shapes allows us to ignore learning.

[4] combine qualitative and quantitative object models to detect and track ten object primitives (box, cylinder, tapered cylinder etc). That system is still brittle in the presence of texture and clutter. To this end [15] describe objects beyond simple shapes but still of limited complexity (cups, hats, desks) with qualitative, parts-based shape abstraction base on a vocabulary of 2D part models corresponding essentially to closed contours of various shapes. Their system can extract such representations from images containing textured objects as well as complex backgrounds.

A major challenge in perceptual grouping is the combinatorial explosion when identifying possible groups of image features as the number of possible groups grows exponentially with group size. [9] addresses the problem of exponential run-time complexity using a grid overlaid on the image indexed by line endpoints. A typical problem of indexing is the appropriate choice of bin size. [18] use further curve parameters to construct index spaces of higher-parametric models and also addressed the problem of bin size and indices close to bin boundaries. [22] proposes to use indexing in the image space where search lines emanating from the ends of image edges are used to find collinearities and junctions and finally closed contours. Search lines are grown incrementally over processing time, thus avoiding problematic distance thresholds.

An overview of monocular model-based 3D tracking is given in [8]. With respect to tracking based on Monte Carlo particle filtering the authors state the following:

We attribute this relative lack of popularity to two different causes. First a large number of particles, perhaps as many as several thousands when the motion is poorly defined, can be required, which slows down the tracking process.

To cope with this problem our particle filter based approach uses texture information, which is more distinctive than only using geometry edges and therefore requires a smaller number of particles. Secondly we implemented the major part of the particle filter on the graphics processing unit (GPU) which allows us to evaluate many particles efficiently in parallel.

Tracking based on surface texture is of course not new. [10] also use edges and texture for tracking. They extract point features from surface texture and use them together with edges to calculate object pose. This turns out to perform very fast and robust with respect to occlusion. Our approach not only uses patches around a few interest points but the whole texture. Also [17] fuse interest points (FAST features) from surface texture and geometry edges for improved tracking. Whereas in both of the above approaches texture and edges are treated explicitly with different underlying operators our approach treats them the same using one underlying operator, namely edge extraction.

Recent approaches typically rely on (possibly combinations of) edge contours, interest point descriptors and optical flow [2, 19, 21]. While interest point descriptors (such as SIFT, FAST etc.) and optical flow are becoming faster to compute thanks in part to GPU implementations edges are still considerably faster to compute but are of course inherently less discriminative than interest point descriptors.

The work presented in this paper is based on [7] where the authors also take advantage of the GPU by projecting a wire-frame model into the camera image. A particle filter with a Gaussian noise model is used to evaluate the confidence level with respect to the pose.

Our approach not only uses geometry edges but also edge features from textures which extends the class of trackable models to those that have curved surfaces as illustrated on the right of Fig. 2. This is because in a standard 3D model curvature is approximated by triangles and quadrangles which would produce virtual edges which do not correspond to the actual edges as shown on the left of Fig. 2.

We found [12] to be the closest related work to our tracking part, but instead of computing the cross-correlation of a pixel patch we propose to evaluate the match between the edge gradients of the rendered model and the camera image. This means that we have fewer comparisons for each pixel which makes it faster. Furthermore edges are more robust against changing lighting conditions. Our approach was partially pre-
sent in [11], with major modifications regarding the particle filtering.

3 Detection of Basic Shapes

In the following sections we show how we detect generic views of basic shapes in edge images by employing an incremental perceptual grouping approach. Having detected 2D shapes we then use a ground plane assumption to generate 3D shape models which are subsequently handed to the tracker.

3.1 Incremental Indexing and Anytimeness

Perceptual grouping in our system is based on the work of [22] which provides an anytime solution to finding junctions between edge segments and subsequently closed contours avoiding the need for arbitrary distance thresholds and [16] which adds higher level features. Indexing is used to efficiently identify candidates for junctions, where the indexing space is the image itself. Each edge endpoint defines a set of search lines consisting of tangential and normal search lines. These search lines are drawn into the index image using Bresenham line drawing. Whenever two lines index into the same bin, i.e. their search lines intersect, we create a new junction. Depending on the types of search lines intersecting we form an L-junction, a collinearity or a T-junction between the respective originating lines. If more than two lines intersect, the according number of pairwise junctions are created. Shortest path search in the resulting graph consisting of edges and junctions then finds closed contours.

In order to avoid the definition of certain length thresholds for search lines they are drawn incrementally, continuously checking for junctions. So the longer we search, the more junctions and eventually closed contours will be found, where “easy” cases typically pop out fast and “difficult” ones (broken edges, partial occlusions, more clutter) follow later. This allows us to stop processing anytime, e.g. after a certain frame time has elapsed or, if we happen to know that we expect pre-processing time. So we rank all hypotheses according to their significance and mark all visited lower level features. With longer processing time the number of estimated hypotheses will grow exponentially, because the more junctions are found between edges the more combinatorial possibilities for higher level groups will appear. Basically this means we will start accumulating “crappy” hypotheses. However many of these higher level hypotheses will share lower level features, thus essentially providing different interpretations for the same underlying data. We are obviously only interested in the best interpretation.

So we rank all hypotheses according to their significance, which is derived from properties such as parallelism, closeness or completeness of Gestalt. We then traverse the list of hypotheses in order of decreasing significance and mark all visited lower level features. Whenever a hypothesis finds one of its constituent features already marked, it will be masked by the higher ranked owner of that feature.

This additional pruning step gets rid of most false hypotheses. Remaining accidental groupings that happen to constitute valid hypotheses could be identified by changing the viewpoint or more generally by observation over time as typically only correct hypotheses, i.e. actual shapes will be stable over viewpoints. In the context of this paper the tracker described in Section 4 takes care of these as only correct hypotheses will produce stable tracks.

3.2 Perceptual Grouping

We then define a hierarchy of grouping principles to enable efficient abstraction of image edges into basic geometric Gestalts as shown in Fig. 3. Triggered by the incrementally growing search lines referred to in the above section, lower level Gestalts such as closures or ellipses are formed and in turn trigger formation of higher level Gestalts such as cuboids and cylinders. Concretely cuboids are defined as three overlapping “flaps”, where a flap is defined as two rectangles connected along one edge. Ellipses are derived from convex groups of intersecting arcs. Cylinders are defined using ellipse junctions: search lines emanating from the major semi-axes of an ellipse meeting straight lines. Two ellipses and two parallel straight lines (possibly of course comprised itself of several collinear lines) thus make up a cylinder. Cones are defined likewise.

Figure 3: Abstraction hierarchy: from edges to basic shapes

Note that as we move up the abstraction hierarchy the corresponding Gestalts get more and more distinctive. So while we will generally find lots of closures, rectangles and ellipses are already somewhat less frequent. Finally cuboids comprised of three flaps or cylinders and cones being composed of a specific topological arrangement of lines and ellipses already rarely appear accidentally. The next section explains how we further reduce the number of false hypotheses.

3.3 Ranking and Masking

With longer processing time the number of estimated hypotheses will grow exponentially, because the more junctions are found between edges the more combinatorial possibilities for higher level groups will appear. Basically this means we will start accumulating “crappy” hypotheses. However many of these higher level hypotheses will share lower level features, thus essentially providing different interpretations for the same underlying data. We are obviously only interested in the best interpretation.

So we rank all hypotheses according to their significance, which is derived from properties such as parallelism, closeness or completeness of Gestalt. We then traverse the list of hypotheses in order of decreasing significance and mark all visited lower level features. Whenever a hypothesis finds one of its constituent features already marked, it will be masked by the higher ranked owner of that feature.

This additional pruning step gets rid of most false hypotheses. Remaining accidental groupings that happen to constitute valid hypotheses could be identified by changing the viewpoint or more generally by observation over time as typically only correct hypotheses, i.e. actual shapes will be stable over viewpoints. In the context of this paper the tracker described in Section 4 takes care of these as only correct hypotheses will produce stable tracks.

3.4 From 2D to 3D

The following tracking procedure in Section 4 requires a 3D wire-frame model of the detected object shape
as well as an initial pose estimate relative to the camera. Note that everything so far is purely 2D, i.e. we detect projections of shapes in generic views onto the image plane. Assuming a camera with known elevation and tilt angle and further assuming that detected objects (cubes, cones, cylinders and spheres) rest on the ground, allows us to convert them to 3D shapes. We intersect view rays with the ground plane and thus obtain 3D position on the plane as well as unambiguous size. Note that we restricted our search to objects belonging to a limited number of simple shape classes. This allows us to “fill in” the unseen backside from simple symmetry considerations.

4 Tracking

Once we have detected objects and generated wire-frame models along with initial pose estimates we initialize a model based 3D pose tracker. Remember that we view our work in the context of robotic applications where we expect to observe manipulated objects or manipulate them ourselves. So it is essential that we have robust and fast estimates of object trajectories.

To improve robustness we enhance the wire-frame models with surface texture, captured directly from the camera image. Tracking based solely on wire-frame edges already provides reasonable performance in many cases but is insufficient for rotationally symmetric objects and runs into problems with degenerate views and heavy background clutter. Adding texture edges provides much more dense information for the tracker to “snap” on to, allowing degenerate views as well as large scale partial occlusions.

To meet real-time requirements we harness the parallel computing power of modern graphics processing units (GPU) for image processing. Graphics boards are designed to render virtual scenes as realistically as possible. The basic idea is to compare those virtual scenes with an image captured from reality. Texturing is a common method of simulating realistic surfaces. In this paper, the edges of those textures are used for comparison. Fast progress in computer graphics will soon allow the inclusion of more and more optical effects such as shadows, reflections, shading, occlusions or even smoke, fire, water or fog. This requires of course available in-

3D position on the plane as well as unambiguous size. 

4.1 Image Processing

In the following an object is described by the geometry of its surface \( S \) (approximated by polygons and vertices \( v \)) and its 6 DOF pose \( x \). Furthermore with the

Figure 4: Block scheme of particle filtering

the algorithm for tracking is illustrated by Fig. 4. It is a modified version of the well known bootstrap filter in [5] applied to vision based object tracking. With respect to computational costs it can be separated into image processing and particle filtering.

\[ \hat{x}_k = \sum_{i=1}^{N} w_{k-1}^{i} x_{k-1} \quad (1) \]

and use this “representative” pose for rendering and subsequent edge extraction from the rendered image. This edge image is then re-projected to the object in world space, temporarily replacing the original surface color texture \( S \) with a corresponding edge texture \( S_e \). This edge texture is then projected several hundred times instead of the color texture. Again this projection is fast as it uses GPU functions.

Note that the obvious solution of generally and permanently replacing color texture with edge texture in the object model does not work. 1-pixel wide edges when projected at different scales and viewing angles will suffer badly from aliasing effects severely affecting edge matching. Generating, for each new input image,
the edge texture once as the edges appear for a “representative” view of the object and using that for rendering of all other (similar) views substantially reduces such effects. Of course this relies on the assumption that the various object pose hypotheses represented by the particles are in fact similar enough, which for normal tracking situations they are.

4.2 Particle Filtering

For each tracking step the particle filter executes the methods shown in Fig. 4. First the particles $\mathbf{x}^i_0$, $i = 1, \ldots, N$, representing the pose of the object, are generated using Gaussian noise. Then the confidence level $c^i_k$ and importance weight $w^i_k$ of each particle $\mathbf{x}^i_k$ are evaluated by matching its corresponding edge image against the edge image of the camera $I^e_{C,x}$. According to the importance weights the set of particles is resampled and then perturbed using again Gaussian noise.

The loop formed by the blocks “Importance Evaluation” and “Resampling with Replacement” is executed several times (2-5 times depending on the power of the processor) before proceeding to the step “Weighted Mean”. We refer to this as Iterative Particle Filtering. This increases accuracy, and therefore also increases robustness especially when the object is moving.

First particles $\hat{x}^i_k$ are resampled from the prior particle distribution $x^i_{k-1}$ according to the importance weights. Then $\hat{x}^i_k$ is perturbed by adding Gaussian noise $n(\sigma)$ with a standard deviation scaled by the prior confidence level $c^i_{k-1}$:

$$\mathbf{x}^i_k = \hat{x}^i_k + n(\sigma)$$

$$i = 1, \ldots, N$$

The standard deviation is evaluated by

$$\sigma = \sigma(c^i_{k-1}, \mathbf{m}_w)$$

$m_w$ is the transformation from the normalized Gaussian noise in the range of $[0, \ldots, 1]$, to the metric world coordinates with respect to the object size.

Each particle is tested against the camera image and a confidence level is calculated. To this end the correlation between the gradients of the edges $g_{S,x}^i(u, v)$ and $g_C(u, v)$ is evaluated by comparing the direction of the edges at each image point $(u, v)$.

$$g_{S,x}^i(u, v) = \left( I^e_{S,x}^i(u, v), I^e_{S,y}^i(u, v) \right)$$

$$g_C(u, v) = \left( I^e_{C,x}(u, v), I^e_{C,y}(u, v) \right)$$

The angles between these vectors are calculated, producing the edge correlation image $\Phi^i$:

$$\phi = \arccos(g_{S,x}^i, g_C)$$

$$\Phi^i(u, v) = \left\{ \begin{array}{ll} 1 - \frac{\phi}{\pi} & \text{if } \phi < \pi/2 \\ 1 - \frac{2\phi - \pi}{\pi} & \text{if } \phi > \pi/2 \\ 0 & \text{if } (u, v) \not\in \mathbf{v}_S \end{array} \right.$$
The second term allocates more weight to the total number of matching pixels \( m^i \) which is intrinsically higher for the rotated particle. \( n_{\text{max}} \) is the maximum number of visible edge pixels in the actual area and scales the pixels to the proper range. To scale the range of the outcome to \([0, \ldots, 1]\) the terms are divided by the sum of their maximum values.

The weights of the particles are calculated by raising \( c_k \) to the power of \( p \), which controls the speed of convergence of the particles. With a higher power \( p \), \( w_k^i \) increases, which leads to more particles assigned to \( x_k \) when resampling the whole particle distribution and therefore to a faster convergence.

As explained in Section 4.1 for projection and re-projection of the model, a single pose \( x_k \) is required. We use the weighted mean of the particle distribution as in equation (1), because it shows good results, both in terms of accuracy and smoothness of the resulting pose.

5 Experimental Results and Evaluation

We made experiments and evaluations with the proposed basic shape detector and with the model tracker. The detector and the tracker are running as separated threads in a distributed framework, using shared memory for the exchange of geometry and pose information of objects. While the detector is triggered only every second to detect major changes in the scene, the tracker runs with high priority at frame rate to achieve high pose accuracy. For each new object appearing in the scene, the detector drops a new geometric model to the shared memory. On the other hand the tracker updates the pose of the models in the shared memory, which allows the system to identify a re-detection.

5.1 Detection of Basic Shapes

The incremental grouping method has been evaluated experimentally with a mobile robot moving among simple geometric 3D objects. Fig. 6 shows an example image, with several objects. The picture indicates a typical problem of grouping, namely that shadows or image noise create spurious features such as lines or arcs. A grouping into higher level Gestalts sometimes accidentally includes a wrong feature. With the incremental approach object detection depends on processing time. The first two images present the cluttered edge image and the extended search lines after 468 ms from the voting image. The following images show the detected object shapes after 328 ms and 468 ms processing time.

The proposed method is able to detect non-textured as well as textured objects in real world scenes. As can be expected, the amount of texture, background clutter and occlusion limits detection rates. Gaps in edges due to occlusion are filled in by search lines if they are not too large (otherwise filling in would take rather long). Adjoining faces with the same color tend to lead to weak edges if lighting is uniform, leading to edge detection failure. Also too much texture over shape edges (e.g. for colourful packaging) will cause problems for detection of the geometry edges. Fig. 7 shows results from two indoor table-top scenes.

To evaluate the capabilities of the perceptual grouping approach, we explored a playground scene containing several cubes with a mobile robot using different processing times. Tab. 1 shows the results of the different runs using an Intel Core2Duo with 2.5 GHz. The whole scene consists of 148 images within 417 cubes to detect. Our approach allows only detection of basic shapes in generic views. With degenerated views, when only one or two surfaces of a cube are visible, only a detection of rectangles or flaps is possible, which may indicate a cube at this position but is not sufficient to build a cube hypothesis. Therefore it is impossible to reach a perfect detection rate of 100 percent with this method of grouping.

Fig. 8 shows the detection rate graphically from the playground scene with the different processing times. As expected the detection rate increases with increased processing time but also the rate of falsely detected cubes. Under 400 ms the false positives (FP) are few but will increase steadily with increasing processing time. I.e. the longer we search the more “hallucinated” shapes will appear.

Please note that the first 150 ms of processing are
used up by Canny edge detection (we use Deriche edge extraction and self-adjusting hysteresis thresholds) and line and arc fitting. Only after that fixed amount of time the incremental processing elements take place and start filling the hierarchy of Gestalts. Hence the detection rate curve starts at 150 ms.

Table 1: True and false positive and negative (TP, FP, TN, FN) detection from a playground scene with four cubes in 148 images at different processing times.

<table>
<thead>
<tr>
<th>Processing time</th>
<th>TP</th>
<th>FP</th>
<th>TN</th>
<th>FN</th>
</tr>
</thead>
<tbody>
<tr>
<td>150 ms</td>
<td>211</td>
<td>0</td>
<td>-</td>
<td>206</td>
</tr>
<tr>
<td>200 ms</td>
<td>294</td>
<td>0</td>
<td>-</td>
<td>123</td>
</tr>
<tr>
<td>300 ms</td>
<td>340</td>
<td>1</td>
<td>-</td>
<td>77</td>
</tr>
<tr>
<td>400 ms</td>
<td>359</td>
<td>12</td>
<td>-</td>
<td>58</td>
</tr>
<tr>
<td>600 ms</td>
<td>378</td>
<td>28</td>
<td>-</td>
<td>39</td>
</tr>
</tbody>
</table>

Figure 8: Detection rate from a playground sequence with 148 images and 417 possible cube detections.

5.2 Tracking of Textured Objects

The advantage of tracking using the edges of the object texture is its high robustness against changing lighting conditions, occlusion, reflections and background clutter. In other words, they are very distinctive, robust and only a relatively small portion of the surface needs to be visible to determine the correct pose. Of course the latter is only true if the visible surface is rich in texture.

Robustness comes with at the cost of speed, which corresponds to the number of particles used. Fig. 10 illustrates the dependency between frame rate and number of particles, where the red shaded area indicates either too low frame rate (bottom) or too few particles (left) which causes jittering and loss of object. We experienced that running the particle filtering loop several times within one image but with less particles further increases robustness without wasting calculation time.

Figure 9: Robustness of tracking. The detected box (upper left image) is occluded by another box in the next frames (lower left image). The algorithm is still capable to track the object although accuracy and the maximum trackable speed drops. The right images show the output of the edge detection algorithm used and illustrate the high level of clutter in the scene (different colors represent edge orientations).

Figure 10: Frame rate with respect to the number of particles

6 Conclusion and Further Work

We presented a system for detection and tracking of basic geometric objects. Shape detection is based on a hierarchical perceptual grouping system where the use of an incremental processing approach eliminates the need for many thresholds and parameters in various Gestalt principles. Assuming known camera elevation and tilt we use the ground plane to generate 3D wire-frame models which are subsequently covered with surface texture and tracked by a particle filter based tracker.

Tracking converges quickly to the correct pose and is able to handle large deviations, for example when initializing. Also partial occlusion, reflections, light changes, shadows and cluttered background are handled thanks to the increase in robustness with the use of tracking.
Table 2: Frame rate with respect to number of polygons of the geometrical model with different number of recursions and particles, computed on a GeForce 285 GTX.

<table>
<thead>
<tr>
<th>Example Objects</th>
<th>Faces</th>
<th>Frames per Second</th>
</tr>
</thead>
<tbody>
<tr>
<td>Box</td>
<td>6</td>
<td>240</td>
</tr>
<tr>
<td>Cylinder (low)</td>
<td>24</td>
<td>220</td>
</tr>
<tr>
<td>Cylinder (mid)</td>
<td>96</td>
<td>210</td>
</tr>
<tr>
<td>Cylinder (high)</td>
<td>384</td>
<td>190</td>
</tr>
<tr>
<td>Complex Scene</td>
<td>1556</td>
<td>160</td>
</tr>
</tbody>
</table>

of texture edges besides simple wire-frame edges. Exploiting the power of a graphics processing unit allows high tracking speed at frame rate.

We are currently working to remove the ground plane assumption required to get from 2D to 3D shape by using stereo, where we perform stereo matching not on pixel but on higher feature level. These are generally much more distinctive than small image patches used in dense stereo and matching of edges allows high accuracy in the stereo reconstruction.

Tracking rate is limited by the number of particles required to obtain sufficiently accurate pose. The number of particles can be reduced by employing a better motion model (right now we simply assume a 0-th order motion model, i.e., a static object). A first order motion model could already improve tracking during smooth trajectories. But actually it is the non-smooth trajectories that pose the real problems and here first order motion models would not help. So we plan to incorporate predictive models from a physics simulation to predict events such as sudden stops when a falling object hits the ground.

Another important issue is the ability of the tracker to report lost tracks, which would result in a re-detection request to the detector. Deciding when a tracked object is truly lost and not just partially occluded or entering strong shadow is non-trivial. To this end we are working on robust tracking confidence measures.

Acknowledgement

The research leading to these results has received funding from the European Community’s Seventh Framework Programme [FP7/2007-2013] under grant agreement No. 215181,CogX.

References


Abstract We present a novel method to extend traditional structure and motion (S+M) analysis of stationary scenes towards dynamic scenes that include independent foreground motion. Outliers - normally discarded by conventional S+M approaches - are used to describe object motion. Our method starts by outlier verification and clustering in scene coordinates. Next, we introduce an object centered representation to independently treat the individual motion per foreground object, and present a simple yet powerful method for the stable and online determination of each object center. Finally, we introduce ideas on solving the loop closing problem for the underlying S+M approach. First experimental results on our own video data demonstrate the viability of this purely geometry-based method that can be used to complement existing appearance- and shape-based object categorization, as well as to improve the camera motion tracking for augmented reality in complex scenes.

1 Introduction

Structure and Motion (S+M) simultaneously reconstructs 3D scene information and observer motion. S+M is purely geometry-based and allows the localization of the moving observer as well as the reconstruction of a stationary scene, in a scene coordinate system. However, traditional approaches to S+M can only handle stationary scenes, and therefore consider independent foreground motion as outliers. Thus, these approaches will fail to reconstruct independent foreground motion or even break down in cases of dominant foreground motion.

Besides online mapping of 3D scene structure, observer pose estimation is the main application area of S+M. For example, Augmented Reality (AR) requires the reconstruction of scene structure and observer pose to augment the real scene with artificial objects at certain positions (e.g. [4, 8]). We are in particular interested in applications of S+M to online object categorization, where many algorithms benefit from camera-to-object pose and object scale.

The work presented in this paper describes an algorithm that treats S+M outliers as feature points on moving rigid objects. This extends the conventional S+M approach to structure, motion, and foreground motion. Thus, this work presents an approach towards the online mapping of Structure and Motion (S+M+O), which has been an open issue up to now. The algorithm is purely geometry-based and allows rigid object detection and motion analysis. It can handle a stationary background structure, a stationary as well as a moving observer, and independent foreground motion. The algorithm introduces an object centered representation for each moving object to provide reliable information on object pose and motion.

We build on the online S+M algorithm by Schweighofer et al. [8]. This algorithm computes online the stable stationary scene structure $S$ in a scene coordinate system:

$$S = \{\vec{x}_a| stationary\}.$$  

In S+M, all information is represented globally in 3D scene coordinates, because the scene is limited to static 3D structure. In addition, observer motion is reconstructed in the same scene coordinate system, leading to a space-time representation of the observer trajectory. The work in this paper distinguishes between stationary background structure and foreground motion. Thus, a preprocessing step either assigns each feature point $\vec{x}_a$ to the set of stable stationary background features $S$ or to independent foreground motion, i.e. objects $O_j$:

$$\vec{x}_a \rightarrow O_j, S.$$  

Fig. 1 illustrates the suggested scene representation. We see the global scene coordinate system $(x_s, y_s, z_s)$, a moving observer, and one moving foreground object $O_k$. The blue features that are associated with $O_k$ are represented in a local, object centered coordinate system $(x_k, y_k, z_k)$, also shown in blue. Observer and object trajectories are represented in the scene coordinate system. This means, that the rigid motion of $O_k$ is represented by the motion (translation and rotation) of its attached coordinate system $(x_k, y_k, z_k)$ in the scene coordinate system $(x_s, y_s, z_s)$.

This paper presents in detail our novel S+M+O algorithm, as well as first experimental results that are limited in terms of: (i) the feature point set $S$ of stationary background features must contain more than 50% of the total number of feature points, and (ii) the current videos show exactly one moving foreground object in the scene. However, despite these

1. The most prominent contribution is certainly by Klein et al. [5, 9, 4], but their concept relies on keyframes and is inherently unable to handle significant changes in the scene (as posed by significant foreground motion).
Using Outliers in Structure and Motion Analysis to Reconstruct Foreground Motion

Figure 1: Scene representation: stationary background and trajectories in scene coordinates \((x_s, y_s, z_s)\), rigid object features (blue) in object centered coordinates \((x_k, y_k, z_k)\).

experimental constraints, the presented method can handle an arbitrary number of independently moving foreground objects.

2 Related Work

Structure and Motion (S+M) has been researched for almost a decade, and still is an active area of ongoing research. S+M introduces a moving observer, which views a certain subset of the scene at a time. By that, S+M allows the reconstruction of a stationary scene. S+M is a very general approach, as it is purely geometry-based. No prior model information is required. In general, one can distinguish between the keyframe-based approach by Klein et al. [4, 9, 5] and various continuous tracking approaches [7, 1, 8]. All are purely geometry-based and rely on feature points detected in the scene.

In its most elaborate version, the work of Klein et al. [5] combines point correspondences with edglets to improve pose estimation of the moving observer. The main idea is to create a map of the observed scene by tracking stable feature points and storing keyframes captured at discrete time steps. Once the moving observer re-enters the scene, the system can quickly resume tracking via the preassigned map. Even if the observer moves very rapidly, position estimation and re-initialization works very well. A very efficient recovery on tracking errors is possible due to the generated map. However, the keyframes just represent discrete (stationary) information. Thus, there is no straightforward extension of this keyframe-based approach towards foreground motion detection. Furthermore, Klein et al. work with a monocular observer. Thus, initial motion is required to initialize the 3D scene reconstruction.

Another map based S+M approach is by Davison et al. [1]. But instead of a prior generated map, here the map is updated continuously by an Extended Kalman Filter (EKF). On appearance of new features, they can be added to the map. If necessary, features can also be removed from the map.

Schweighofer et al. [8] perform continuous tracking without prior map generation. The approach is designed for the general camera model (GCM) which allows to use any kind of camera geometry. This is achieved by the substitution of a pixel by a raxel as core element. A raxel is a union of a pixel and the direction of the light ray. This is based on the assumption, that line travels along a line into the camera [3]. In the experiments of [8], the GCM is reduced to a calibrated stereo rig. Continuous tracking is performed with this pair of cameras. Thus, 2-1/2 D initialization can be done without any initial motion sequence, based on the first captured stereo pair. In contrast to the keyframe approach, no map is generated. The position of the moving observer is computed from the visible feature points of the scene. Once a corresponding feature point is out of the field of view, it is lost. When reappearing, no knowledge of the previous positions is preserved. Thus, the position of the reappeared feature points is computed based on the remaining set of the visible scene. This results in an accumulative error, which increases over time. This error accumulation and propagation has also been identified as a serious problem by Nister et al. [7]. They suggest to use a “firewall” concept, where a further error accumulation is prevented by re-initialization.

Further related works are Leibe et al. [6] and Ess et al. [2]. In contrast to S+M, these works are not purely geometry-based. The first one provides car detection and tracking, the second one pedestrian detection and tracking. Both approaches handle object as well observer motion. However, they can be applied to particular object categories only, either cars or humans, by using quite elaborated object detection algorithms.

3 S+M+O Algorithm

The proposed algorithm is based on feature points, thus relies on point correspondences. Common issues for feature point detectors (such as occluding edges and the visual rim) and errors due to imprecise stereo reconstruction directly flow into the output of the described work. We presume rigid objects in the scene only. Any motion of a rigid object is subject to a continuity constraint, i.e. no discrete changes in position are allowed. Even in cases of very dominant foreground motion, a stable background structure is required. Without stable structure no observer pose estimation and scene reconstruction are possible and the described algorithm will not work properly.

The algorithm consists of three parts. The first part relies on the feature points gathered by the used S+M algorithm [8] (refer to fig. 2). First, stable background features are marked as inliers, whereas unstable as well as moving feature points are declared as outliers. If a rigid object moves in the foreground, a certain subset of the outliers should behave consistently. Based on this idea, outliers derived from moving objects are separated from “real” outliers, i.e. reconstruction errors (refer to 3.1). Second, a distance matrix is generated with all valid outliers, i.e. features on moving objects. This
distance matrix is then used as an input to a clustering algorithm, which is described in detail in section 3.2. Third, subsection 3.3 presents the core of the current algorithm: based on the preprocessed information, an object centered coordinate system is established for each moving object. The basic three-step algorithm is illustrated in fig. 3.

In summary, in two preprocessing steps, we distinguish between outliers and stable structure, and cluster all 3D points into distinct sets of 3D points that correspond to objects. The remainder of the algorithm runs online, where we compute a stable object centered representation, enable object motion prediction based on position, velocity, acceleration (Kalman Filter), and maintain the tracking of temporarily lost (occluded) features.

3.1 Outliers

Feature points are gathered by the S+M algorithm of Schweighofer et al. [8]. Thus, point correspondences are established and tracked by S+M. The output of the S+M algorithm provides a unique ID and the corresponding 3D coordinate for any detected feature point per time step. Additionally, each feature point provides a tag on whether the feature point is considered as stable (i.e. stable structure information) or as outlier (i.e. moving feature point or reconstruction error). The visual output of the S+M scene analysis for inlier and outlier detection is shown in fig. 2.

The preprocessing step analyzes each outlier over the time steps it appears. Thereby, the following investigations are started:

\[ \text{visibility}(\vec{x}_s) > F_{\text{min}}. \]  

A certain feature point \( \vec{x}_s \) has to be visible for at least \( F_{\text{min}} \) frames (i.e. the amount of time steps).

\[ \text{variance} \left( \begin{bmatrix} \vec{x}_{s,1} \\ \vdots \\ \vec{x}_{s,n-1} \\ \vec{x}_{s,n} \end{bmatrix} \right) > TH. \]  

The visibility of a feature point \( \vec{x}_s \) over successive time steps defines the term trajectory. The variance of the feature point trajectory \( k \) over all \( n \) frames (i.e. time steps) has to be above a certain threshold \( TH \) for all directions \( x, y, \) and \( z \). In our experiments, \( TH = 1 \, \text{mm} \) has been used.

\[ \Theta(\vec{x}_{k,j-2}, \vec{x}_{k,j}) > \phi_{\text{min}}. \]  

The spanned angle of 3 successive time steps \( j-2, j-1, \) and \( j \) on the trajectory \( k \) has to exceed a certain minimum angle \( \phi_{\text{min}} \) for all \( n \) time steps. In our experiments, \( \phi_{\text{min}} = 75^\circ \) has been used (see fig. 4).

\(^2\)In this work, we measure temporal differences in “time steps”, that correspond to the constant frame rate of 20Hz of our stereo rig. The time difference between a captured frame \( j \) and the successive frame \( j+1 \) is described as one time step.
If all these requirements are fulfilled, the feature point is considered to be located on a moving rigid object. Thus, it is passed as input to the object clustering routine.

### 3.2 Object Clustering

A distance matrix with valid feature points is created, after the real outliers have been separated from those on moving rigid objects in the previous step (see section 3.1). The distance matrix has a dimension of $N \times N$, where $N$ is the total amount of feature points (and therefore also trajectories) on moving rigid objects in the scene.

A feature trajectory is derived by tracking one feature point over the time it appears. Every feature $j$ along the $n$ point long feature trajectory $k$ is compared to every feature $i$ on every other $m$ point long valid feature trajectory $l$. This results in $n \times m$ difference vectors of length $m$. For simplicity, the norm is saved only. These $n \times m$ difference vectors are then represented as a matrix (see fig. 4). Each column contains one of the $n$ norm vectors $\vec{v}_n$ with length $m$. Once this matrix is established, the minimum is searched in each of the $m$ rows. The norm vector $\vec{v}_n$ (i.e. the column $\vec{v}_n$) with the smallest minimum value is also the closest distance between a point on the trajectory $k$ and a point on trajectory $l$. The variance of this column $\vec{v}_n$ is then taken and passed to the distance matrix on position $(k, l)$.

![Figure 4: Comparison of trajectory $k$ with another trajectory $l$. In this figure, step $j$ out of $n$ is illustrated, i.e. the result is the $j^{th}$ difference vector with size $m$. All points on trajectory $k$ are compared with all $m$ points on trajectory $l$. By repetition, the result is $n$ vectors with the appropriate norm values for $m$ points on trajectory $k$.](image)

This distance matrix is used as input for the K-means clustering algorithm. Additionally, the desired amount of clusters has to be passed as parameter. This, of course, is scene dependent. In the simplest case - one moving object and stable background - it is one. However, our experimental results show that correct clustering is also achieved, if a higher number of clusters than actually moving objects is selected. If an object is split up into more than one cluster, we refer to the discussion in the results section (see section 4) on further processing to re-merge these clusters. The number of clusters is scene dependent. Currently, this number of clusters for scenes is specified manually. However, it turned out if specifying more clusters than objects still works for certain experiments. The major drawback here is that the probability of object subclustering increases.

Currently, outlier retrieval and clustering are implemented in a preprocessing step. However, one could easily include at least the clustering step into online processing by using a short frame (i.e. time step) period as history for proper cluster information. Up to now, our investigations have been focused on the object centered representation and tracking of moving objects (see section 3.3). Thus, online clustering has been neglected so far.

![Figure 5: 90° Rotation of a rigid object. Left: $OC_{comp} \ominus OC_{real}$. The rotation at $origin = OC_{real}$ results in the same rotation for $OC_{comp}$. Right: $OC_{real} \neq OC_{comp}$. The rotation at $origin = OC_{real}$ results in a rotation and translation for $OC_{comp}$.](image)

### 3.3 Rigid Object Representation

After successful clustering, ideally one cluster corresponds to one object. Each cluster is represented by a row in an object matrix. Each row in the matrix contains the unique IDs of feature points belonging to the actual cluster. At this point the online processing starts.

In general, for each frame (i.e. time step) $j$ the available feature points are loaded and mapped to the clusters as assigned in the object matrix. An object centered coordinate system is established after initialization. This is done by computing the weighted object center (OC) for the visible feature points at a certain time. $n$ nearest neighbors are searched and the position vectors to the weighted OC are stored. These position vectors are equal to the difference vectors between OC and feature point in the scene coordinate system.

For each new frame (i.e. time step) $j + f$, $f > 0$, the object center can be re-estimated by finding the same neighbors as in the previous frame $j + f - 1$. Then, for each neighbor the relative distances to the object center can be evaluated. On disappearance of one or more neighbors, new neighbors have to be established.

The goal of the algorithm is to find a stable point $OC_{comp}$ per object which represents the origin of each object’s centered coordinate system. This origin need not necessarily be equal to the actual center of gravity of the observed object - we term this $OC_{real}$ - but needs to be reliable enough
for analyzing the object motion behavior. However, in case $OC_{\text{comp}} \neq OC_{\text{real}}$, self-rotation at origin = $OC_{\text{real}}$ results in a rotation and translation at position $OC_{\text{comp}}$ (refer to fig. 5). Our experiments (see section 4) have shown, that neglecting the translation vector in case of $OC_{\text{comp}} \neq OC_{\text{real}}$ still provides good motion behavior description. This of course depends on the actual object size. The larger the object, the more influence the translation vector gains.

The next section describes the initialization process. Section 3.5 explains the regular update of the established object center, while section 3.6 explains the update of the OC when used neighboring feature points are lost. Section 3.7 describes the idea of re-mapping lost feature points onto newly appearing feature points. Besides the OC computation, a Kalman Filter (KF) is used to provide smoothing of object motion and a prediction of the OC motion over time. This is useful in case no OC computation is possible, for instance when too few feature points are found on the object. The KF design is explained in section 3.8.

3.4 Initialization of the object centered representation

First, for each cluster (i.e. assumed object) a weighted OC is computed. Each cluster then is represented by a tree data structure with hierarchy depth 1. The weighted OC is stored with its scene coordinates (i.e. the position of the weighted OC in the initialization phase) as root in the tree. Once the weighted OC is established, n nearest neighbors are searched in the current frame. n is arbitrary but its minimum value is 4. This is due to the rotation (and translation) estimation, which requires at least 4 corresponding feature points in two successive time steps. For each neighbor, the position vector to the OC is used as coordinate instead of its scene coordinate. This introduces the object centered coordinate system. The neighbors are then stored as children of the weighted OC in the tree. Each neighbor stores its position vector to the weighted OC $\vec{d}$ in the tree data structure. The established OC and its neighbors of one object as well as the data structure are illustrated in fig. 7.

Besides the tree representation of each cluster, a Kalman Filter (KF) is introduced for each cluster, i.e. object. The aim of the KF is to predict the weighted OC in cases of temporal loss of neighbors, such that no object center determination is possible. At the initialization step, the weighted OC is the first input to the Kalman Filter (KF).

3.5 Update

The update routine presumes the successful initialization of each visible object. For each object the weighted OC and all its children are stored in the tree data structure (see fig. 7). Each child contains the initial position vector $\Delta \vec{d}$ to the OC.

In the next time step $j$ (e.g. next frame) every child (i.e. neighbor) provides a hypothesis for the new position of the OC. However, $\Delta \vec{d}$ does not contain any information on the occurred object rotation $\Delta R$ performed between two successive time steps. Therefore, the rotation $\Delta R$ (and translation $\Delta \vec{t}$) between the previous time step $j - 1$ and current $j$ are required. Unfortunately, the object coordinate system and with it the relative coordinates of the neighbors can be established only if the origin (i.e. the OC) is known. However, the OC can only be computed by knowing $\Delta R$ and $\Delta \vec{t}$. To overcome this chicken and egg problem, a simple solution has been implemented:

- In time step $j - 1$, the $OC_{\text{comp}}$ position in the scene is established. $e > 4$ neighbors are found and stored in the tree data structure with their position vectors $\Delta \vec{d}_{e}$ to $OC_{\text{comp}}$.
- In time step $j$, we assume that the position of the $OC_{\text{comp}}$ is still the same. We find the same $e > 4$ neighbors now in time step $j$. In case of motion or rotation, the neighbors will provide slightly different relative positions compared to time step $j - 1$. Now, we compute the rotation matrix $\Delta R$. The resulting rotation matrix $\Delta R$ is an estimation due to the assumption that the $OC_{\text{comp}}$ did not change from time step $j - 1$ to $j$.
- We take the previously stored position vector $\Delta \vec{d}_{e}$ of all $e > 4$ neighbors of the OC and apply the rotation matrix $\Delta R$ to all position vectors $\Delta \vec{d}_{e}$. For each neighbor, a hypothesis for the new OC $OC_{\text{new}}$ is created:

$$OC_{H} = \vec{x}_{\text{neighbor}} + \Delta R \ast \Delta \vec{d}$$

(6)

where $\vec{x}_{\text{neighbor}}$ is the position in scene coordinates of the actual neighbor in time step $j$.

- Now, every neighbor of the origin of the object coordinate system (i.e. OC) provides a hypothesis for the new position of the OC in scene coordinates. Currently, the mean of all hypotheses per object is selected as new origin of the object coordinate system (i.e. $OC_{\text{comp}}$). This new origin $OC_{\text{comp}}$ is also passed to the KF for prediction.

- The $OC_{\text{comp}}$ coordinates are updated in the tree data structure.

- This procedure is repeated for each object.

3.6 Update after feature loss

Features appear and disappear over time due to occlusion, rotation, or even leaving the field of view. Thus, children of the specified OC may disappear at any time. One has to distinguish between two scenarios, where either all children disappear, or a subset of children disappears.

- All children disappear: In this case, no OC computation can occur. In the current time step, there exists no neighbor which is stored as child of the OC in the tree representation. The KF has to be used to predict the OC. After prediction, new neighbors of the predicted OC can be established. This happens in the same manner as in the initialization process (see section 3.4).

- A subset of children disappears: There are still $r > 1$ neighbors visible. The remaining neighbors provide hypotheses for the OC position. After assigning the OC to the preferred hypothesis, $e - r$ new neighbors have to be established. This follows the procedure described in section 3.4.
Using Outliers in Structure and Motion Analysis to Reconstruct Foreground Motion

Figure 6: 360° rotation of a rigid object. A feature point and its object centered coordinates are established (a). The feature point disappears, but can be estimated due to the saved object coordinates (b-c). The feature point reappears (d). Now the estimated position of the disappeared feature point has to be compared to any newly entered feature points. The closer $OC_{comp}$ is to $OC_{real}$, the more accurate is the estimation. In (e) the feature point is still visible.

Figure 7: Left: established weighted OC of a rigid object. As shown, $n = 4$ neighbors have been established. The line from each neighbor to the OC represents the position vector. Right: illustration of the stored data type of an object.

Figure 8: Output of the described algorithm. The bounding box of each cluster is colored yellow. The features of each cluster are represented as colored circles. The yellow cross illustrates the 2D projection of the established object center $OC_{comp}$. The 2D projection of the Kalman prediction is shown as green cross. Left: the object has been correctly clustered. Right: sub-clustering of the actual object.
3.7 Re-mapping of re-appeared feature points

Re-mapping addresses the re-detection of previously disappeared feature points and thus the loop closing problem. The used S+M approach discards all information about disappeared feature points, no matter if they are self-occluded by rotation of an object or disappeared (not detected / occluded / out of the field of view) for the time being. No information on re-appearing features is provided. Every new feature point gets a new unique ID.

To enhance the stable object centered representation, the need for re-mapping arises. We propose a re-mapping routine which shall be included into our algorithm. At this time, the re-mapping routine has not been implemented yet. It constitutes the next step in our ongoing work.

Once a neighbor (i.e. a child of the OC in the tree representation) is lost, it is not discarded from the tree. It is kept together with the stored distance information to the OC. However, a visibility flag is used to mark this child as invisible.

Now, any time new feature points appear in the scene, the distance to the actual OC can be compared to every invisibly tagged child in the tree representation of a certain cluster. This is possible even on self-occlusion by rotation of the object, as the rotation is estimated over time. If a previously lost feature reappears, distance information is similar. However, this is only valid for rigid body objects and a stable OC $OC_{comp}$. The closer $OC_{comp}$ is to the real OC $OC_{real}$, the more precise the matching. Fig. 6 illustrates a schematic re-mapping procedure. While this re-mapping concept is also purely geometry-based, it might also be assisted by feature point descriptor, using any state of the art local descriptor.

3.8 Kalman Filter

In addition to the implementation of the object centered coordinate system estimation, a Kalman Filter (KF) is used to predict and smooth the origin computation (i.e. $OC_{comp}$) over time. This can be useful in case the origin of the object coordinate system cannot be established, e.g. when no feature points (and thus no neighbors) are visible. This may occur due to temporary (self-) occlusion or lack of textured information on the object.

Our KF uses a 9-dimensional state vector, which consists of the 3D position $(x, y, z)$, velocity $(\dot{x}, \dot{y}, \dot{z})$, and acceleration $(\ddot{x}, \ddot{y}, \ddot{z})$. The computed origin of the object coordinate system $OC_{comp}$ serves as position input to the KF. The position difference $\Delta d_1$ from time step $j-1$ to $j$ represents the velocity input, and the difference of the position differences from time step $j-2$ to $j-1$ and $\Delta \Delta d$ are considered as the acceleration input. On the generation of the Kalman gain, each of the entities - position, velocity, and acceleration - are weighted differently. At current stage of implementation, the KF is smoothing the object motion description very well (refer to fig. 9). However, as can be seen in fig. 8, where the prediction is a bit behind the true position, there is a tradeoff between smoothing and lag, as should be expected from this kind of modeling with a KF. Thus, we might consider further extensions or tuning of the Kalman gain computation w.r.t. the acceleration input in our future work.

4 Experimental Results

As this is current, ongoing work, we can report here about first experiments on S+M+O with dominant foreground motion of precisely one single foreground object (a toy horse that slides on a table by tearing it with a string). Fig. 8 shows two results for this experiment using different clustering parameters. The yellow rectangles represent the bounding boxes of each cluster (i.e. object). The circles within the bounding boxes represent the feature points for each cluster. The yellow cross illustrates the computed OC $OC_{comp}$ back projected to 2D, whereas the green cross is the 2D projection of the KF prediction (note the lag discussed above in section 3.8).

The sequence was captured by two cameras mounted on a stereo rig using a constant frame rate of 20 Hz. It consists of 313 frames per camera, i.e. the sequence has a total duration time of approximately 15 seconds. The algorithm has been tested on an Intel Core 2 Quad PC with 2.8 GHz and 1 GB RAM using Matlab 7.6 on a 32 bit version of Ubuntu 8.04. However, Matlab was run on one core only. The preprocessing step - which includes feature point trajectory creation and distance measuring - took about 30 seconds. The online processing step - the geometric model updating process and rotation estimation - depends on the amount of objects (i.e. clusters) in the scene. In case K-means correctly clustered one object, it took approximately 30 seconds (i.e. 10 frames / second). However, if K-means falsely clustered the object into 3 clusters, the online processing took about 60 seconds (i.e. 5 frames / second). We want to mention, that the preprocessing step has not been optimized so far. Thus, the overall execution time could be decreased.

In the first result (fig. 8, left), K-means clustered the scene into two clusters only (shown at frame 100), whereas in the second result (fig. 8, right), 6 clusters have been enforced by the clustering procedure (frame 137). One can see clearly that sub-clustering of the real object occurs. To prevent sub-clustering, some investigations on the behavior of sub-clusters have been done. In fact, when sub-clustering of an object occurs, the computed OCs of each sub-cluster as well as the KF predictions behave in a similar manner. This information could be used to merge these clusters to a single one. Together with the tuning of the Kalman gain and the re-mapping of lost feature points, this is one of three major steps in our ongoing work.

The output of the OC measurement is shown in fig. 9. For better legibility, only the width $x$ and depth $z$ are shown. The height $y$ actually is stable and its values are reproducible. Focusing on the depth $z$, one clearly can see the differences in depth estimation depending on the number of sub-clusters (fig. 9, left vs. right). This difference of approximately 1 cm is due to the differently clustered neighbors of the OCs (one cluster vs. three clusters), which leads to slightly different motion behavior of the computed OCs $OC_{comp}$.

As stated in the introduction, our current S+M algorithm requires more than 50% stationary background points to reliably estimate observer motion w.r.t. a stable scene coordinate system. This requirement is fulfilled in the above experiment. In our ongoing work, we search for ways to relax
Using Outliers in Structure and Motion Analysis to Reconstruct Foreground Motion

Figure 9: Left: The scene has been clustered into two clusters only. estimation (red). Right: The scene has been clustered into 6 clusters. cluster 2 (green), cluster 3 (cyan), and their Kalman predictions (red).

this constraint. This could be achieved either by higher level semantic reasoning (finding a few, reliable, well detectable, stationary points in the background), or by splitting the dominant foreground motion into independently moving objects. As long as there are several objects moving in arbitrary, different directions, this should be feasible. Problems will certainly occur in cases of dominant, but homogeneous foreground motion, when many objects move in a synchronous manner.

5 Conclusions

The novel algorithm described in this paper uses outlier information which is discarded by conventional S+M approaches. We model these outliers as rigid body motion information in the foreground of a static scene. The purely geometry-based algorithm extends the S+M approach with foreground motion, which we call “S+M+O”. K-means clustering is used in a preprocessing step to distinguish between various objects. The rigid body constraint allows to analyze motion behavior without knowledge of any further object model. Due to the use of an existing S+M framework, foreground motion analysis can deal with either a moving or a stationary observer. The benefit of the described work is the deployment of an object centered representation, where a computed stable origin \( OC_{comp} \) has been introduced. This origin \( OC_{comp} \) is not necessarily the real object center of gravity. A Kalman filter is used to smooth the estimation process. First experiments show that reliable motion information can be gathered. An important issue is the setup of a stable and reliable background structure. A procedure to filter strong stationary background features could further increase the stability of the algorithm and is ongoing work. Currently, we require more than 50% of the found features to belong to the stationary background. Otherwise, the results of the proposed algorithm deteriorate, due to the used S+M approach.

In the next stage, object event handling will be introduced. Thus, events like occlusion, merging and splitting will be analyzed. Loop closing is another open issue, but the remapping approach described in 3.7 starts to address this issue.

Acknowledgement

This work has been funded by FWF project S9103 and FFG project EViS (813399) under the FIT-IT program.

References

Abstract In this work we propose a principal approach combining graph cuts, color weighting and texture information to segment skin regions from images. Successfully detected faces serve as foreground seeds, no assumptions about the background is made. Further, we introduce the concept of skin weights and skin weighted images. Performance is improved by efficiently adjusting pixel weights. Prior approaches do not provide a general skin detection when the information of foreground or background seeds is not available, e.g. when there are no faces present in the image. A concept for processing arbitrary images is introduced: We learn a universal seed to overcome the probable lack of successful seed detections. Experiments on a database of 300 images with annotated pixel-level ground truth show that the scheme outperforms widely used approaches. We also evaluate the effect of different edge detectors on graph cuts based skin segmentation. It is shown that a texture and color based weighting scheme improves skin segmentation compared to a color only weighting scheme.

1 Introduction

Skin color detection is a popular and useful technique because of its wide range of applications both in human computer interaction and content based analysis. Applications such as: detecting and tracking of human body parts [1], face detection [5], naked people detection and people retrieval in multimedia databases [6] benefit from skin detection. Also skin detection gains attention in contributing to block objectionable image or video content on the Internet automatically [31].

The most attractive properties of color based skin detection are potentially high processing speed, invariance against rotations, invariance against partial occlusions and invariance against pose changes. However, standard skin color detection techniques are affected by changing lighting conditions, complex backgrounds and surfaces having skin-like colors.

The primary objective of skin detection or classification is building a decision rule that will differentiate between skin and non-skin pixels. The most widely used approach to identifying skin colored pixels involves creating a static skin filter, a volume into which most skin pixels would fall in a given color space [32]. There is a set of techniques which estimate the distribution of skin color by a training phase. These methods are often referred to as non-parametric skin models [13]. Finally, other methods include parametric skin distribution models, such as the Gaussian skin color model [36].

This paper introduces a skin detection process using graph-cuts. The skin segmentation problem is modeled as a min-cut problem on a graph defined by the image color characteristics. The vertices of the graph represent the image colored pixels and edges represent weights or costs for labeling the vertices as skin or non-skin.

The main contribution of the paper is to use additional skin weights to augment the graph edge weights. We introduce the idea of the skin weighted image, where the pixels are replaced by their corresponding weights in the non-parametric model. The weighted non-parametric model is built from a number of different images containing skin.

The segmentation process starts by building a decision rule based on a seed. For the seed, we use faces present in the image. We therefore assume that there is a face present in the images in which we detect skin. For images without a face we use the proposed concept of a universal seed. For images that are related to each other, such as frames from the same scene in a video, it is possible to use faces found in other images as seeds in related images.

For the seed generation process we use the well known face detection approach from Viola Jones [33]. The skin segmentation technique is based on the interactive graph cuts method, first used for segmentation in [3]. There is a very efficient algorithm for finding min-cut/max-flow in a graph [2]. For general skin detection i.e. without the knowledge of the seed from image locally, we present the concept of universal seed.

The skin segmentation process is summarized in a block diagram, see Figure 1. Two types of weights have to be adjusted in the first step. One is the neighborhood weights and the other is pixel (background/foreground) weights. A graph is constructed given the proper adjustment of the neighborhood weights and background/foreground weights. A graph cut technique is used to segment the skin. Finally we evaluate the approach in the presence of different parameters.

Experiments have been performed to compare skin segmentation using (1) augmented weighted graph cuts technique (2) simple graph cuts technique (3) static models
Weighted Skin Color Segmentation and Detection Using Graph Cuts

2 Related Work

In computer vision, skin detection is used as a first step in face detection, e.g. [27], and for localization in the first stages of gesture tracking systems, e.g. [1]. It has also been used in the detection of naked people [8, 16] and for blocking objectionable content [31]. The latter application has been developed for videos.

The approaches to classify skin in images can be grouped into three types of skin modeling: parametric, non-parametric and explicit skin cluster definition methods. The parametric models use a Gaussian color distribution since they assume that skin can be modeled by a Gaussian probability density function [36]. Non-parametric methods estimate the skin-color from the histogram that is generated by the training data used [13].

An efficient and widely used method is the definition of classifiers that build upon the approach of skin clustering. This thresholding of different color space coordinates is used in many approaches, e.g. [25] and explicitly defines the boundaries of the skin clusters in a given color space, generally termed as static skin filters. The underlying hypothesis is that skin pixels have similar color coordinates in the chosen color space, which means that skin pixels are found within a given set of boundaries in a color space. The static filter used in YCbCr color space for skin detection as reported by [7] is:

\[
Cb_{\text{max}} = 127, \quad Cb_{\text{min}} = 77 \\
Cr_{\text{max}} = 173, \quad Cr_{\text{min}} = 133
\] (1)

A similar static filter for RGB color space as reported by [24] is,

\[
R > 95, \quad G > 40, \quad B > 20 \quad \text{or} \quad |R - G| > 15, \quad R > G, \quad R > B \quad (\text{Max}\{R, G, B\} - \text{min}\{R, G, B\}) > 15
\] (2)

Although this approach is extremely rapid, its main drawback is a comparably high number of false detections [14]. Khan et al. [15] addressed this problem by adapting the skin-color model according to reliably detected faces. When more than one detected face exists in a frame and face information is properly extracted, multiple adapted models are used. The multiple model approach makes it possible to filter out skin for multiple people with different skin tones and reduce its false positives. The dynamic multiple model approach outperforms static approaches.

Color is a low level feature that is computationally inexpensive. For many applications in computer vision, it is suitable for real-time object characterization, detection and localization [16]. The main goal of skin color detection or classification is to build a decision rule that will discriminate between skin and non-skin pixels. Following Kucumanu et al. [14] the major difficulties in skin color detection are caused by various effects such as illumination circumstances, camera characteristics, ethnicity, individual characteristics and other factors like makeup, hairstyle, glasses, sweat, and background colors. An approach for reliably detecting skin has therefore to be stable against noise, artifacts and very flexible against varying lighting conditions.

Color spaces like the HS* family model the RGB cube onto a transformed color space by following perceptual features. The angular Hue component gives the perceptual idea of a color tone. The Saturation gives a measure of the colorfulness. The HS* color spaces are broadly used in the scenarios of skin detection. Examples are found in [4, 10, 11].

To simulate the primates visual attention, perceptually uniform color spaces like the CIELAB, CIELUV are used for skin detection e.g. by [5]. Orthogonal color spaces like YCbCr, YCgCr, YIQ, YUV, YES try to form as independent components as possible. YCbCr is one of the most successful color spaces for skin detection and used by e.g. [34, 27].

Skin detection under varying illumination in image sequences is addressed in [29, 35, 30]. These approaches try to map the illuminance of the image into a common range. They compensate for the variance of changing lighting to equalize the appearance of skin color throughout different scenes. These methods are dependent heavily on the lighting correction techniques and their ability to estimate the illuminant.

Neural networks [17], Bayesian Networks e.g. [26], Gaussian classifiers e.g. [13], and self organizing maps
[4] have been used to try to increase the classification accuracy. These methods require a considerable amount of training and have high classification accuracy for a particular narrow domain.

In the literature of segmentation, Graph-cuts provide a globally optimal solution for $N$-dimensional segmentation when the cost function has specific properties as defined in [3]. A semi-automatic method for general image segmentation was created by Boykov et al. [3]. A user puts marks on the image, acting as a cue for being counted as segments and updating the marks without graph reconstruction. The method of Li et al. [18] consists of two steps: an object marking task as in [3] and the pre-segmentation, followed by a simple boundary editing process. The work of Shi & Malik [28] segments the image into many non-overlapping regions. They introduced normalized graph cuts and the method has often been used in combination with computing pixel neighborhood relations using brightness, color and texture cues [19, 9, 20].

Branislav et al. [23] make an assumption that each texturized or colored region can be represented by a small template, called the seed and positioning of the seed across the input image gives many possible sub-segmentations of the image. A probability map assigns each pixel to just one most probable region and produces the final pyramid representing various detailed segmentations. Each sub-segmentation is obtained as the min-cut/max-flow in the graph built from the image and the seed. In our work we use detected faces as seeds for skin segmentation. Graph cuts is used for skin segmentation in [12] using both foreground and background seeds. We only use foreground seeds. Additionally we augment the weights and finally present the idea of a universal seed for general skin segmentation.

3 Graph Representing the Skin Image

The skin segmentation technique is summarized in Figure 2. The skin segmentation is based on a seed from the image on which skin segmentation is to be applied. A graph is constructed whose nodes represent pixels and whose edges represent the weights. The min-cut/max-flow algorithm presented in [2] is used for the graph cut. Later on we will show that we can perform skin segmentation without the need for a seed from the image, i.e. general skin detection.

A graph $G$ for skin image $I$ consists of a set of nodes $V$ and a set of edges $E$ that connect them. There are terminal nodes and non terminal nodes. The non-terminal nodes represent pixels in image $I$. The edges which connect pixels or non-terminal nodes to each other constitute a neighborhood. We call these edges $n = \text{links}$. With reference to the skin image, the edges among the non-terminal nodes represent the connectivity or neighborhood relationship between the pixels in the skin image $I$. The non-terminal edge weights are calculated using color and texture features and we call them neighborhood weights, explained in Section 4.

The terminal set consists of two nodes, the source $S$ and the sink $T$. For the skin segmentation problem we represent these nodes as the foreground $F$ node for “skin” and the background $B$ node for “non-skin”. These terminal nodes are connected to the non-terminal nodes through edges. The edges connecting pixels in image $I$ and terminal nodes $B$ and $F$ are called $t = \text{links}$ edges. The cost of a $t = \text{link}$ connecting a pixel $p$ and a terminal node $B$ and $F$ corresponds to a penalty for assigning the corresponding label to the pixel $p$. This is the probability value of a pixel being “skin” or “non-skin”. For the skin detection problem we call it pixel weight, represent it by $\omega$ and the method for its calculation is explained in Section 5. For robust skin detection, we augment these weights using skin distribution or skin confidence where we increase or decrease the weights based on the pixel probability of being a “skin” or “non-skin” using a YCbCr skin distribution.

The general framework for building the graph is depicted in Figure 3. The graph is shown for a 9 pixel image and 8-point neighborhood $N$.

![Figure 2: A graph cut for skin segmentation. A mask represents the object terminal and the whole image itself represents the background terminal. Note that hair is detected as skin because the seed covers the hair portion.](image)

![Figure 3: Left: Graph representation for 9 pixel image. Right: Table defining the costs of graph edges. $\lambda$ is a constant and its value is fixed to be 1000. (source: [22]).](image)

4 Neighborhood Weights

The non-terminal nodes represent the pixels of image $I$ in graph $G$. The edges represent a neighborhood relationship between the pixels. The edge weights of neighborhood $N$ are given by matrix $W_{q,r}$. The neighborhood size and density has great impact on the computation times. For the problem of skin segmentation we control the size and density has great impact on the computation times.
density of the neighborhood through two parameters. These parameters are window size and sampling rate. We are using a neighborhood window of size $21 \times 21$. For skin segmentation we are using a sampling rate of 0.3. This means that we are only selecting at random 30% sample of all the pixels in the window. There are two reasons: Firstly by using only a fraction of pixels we are reducing the computational demands and secondly only a fraction of pixels allows the use of larger windows and at the same time preserve the spatial relationship between the neighboring pixels. We show the calculation of weight matrix $W_{q,r}$ as follows.

For a black and white image the boundary penalties, as reported by [3] are

$$W_{q,r} \propto e^{-\frac{|I_q - I_r|^2}{2\sigma^2}} \cdot \frac{1}{|q - r|}$$

(3)

where $I_q$ and $I_r$ are the intensities at point $q$ and point $r$, $|q - r|$ is the distance between these points and $\sigma$ is a parameter. For skin detection in color images we modify the above function to take color into account as in [21]. The straightforward modification is

$$W_{q,r} = e^{-\frac{|c_q - c_r|^2}{\sigma_1^2}} \cdot \frac{1}{|q - r|}$$

(4)

where $c_q$ and $c_r$ are the RGB vectors of points at the position $q$ and point $r$.$|q - r|$ is the distance between these points and $\sigma_1$ is a parameter. For skin detection purpose a value $\sigma_1 = 0.02$ is used, which is the optimized value for segmentation in [21] and is obtained experimentally for giving the best performance on a large database of images. The boundary penalty of Equation 4 only holds for images having no texture.

For taking texture into account, the neighborhood penalty of two pixels is defined as follows

$$W_{q,r} = \left( e^{-\frac{g(q,r)^2}{\sigma_2^2}} \right)^2$$

(5)

where $\sigma_2$ is a parameter. For skin segmentation we used $\sigma_2=0.08$, and is obtained experimentally on a large database of images and

$$g(q,r) = p_b(q) + \max_{s \in \mathcal{L}_{q,r}} p_b(s)$$

(6)

where $p_b(q)$ is the combined boundary probability and

$$\mathcal{L}_{q,r} = \{ x \in \mathbb{R}^2 : x = q + k(r - q), k \epsilon (0,1) \}$$

(7)

is a set of points on a line from the point $q$ (exclusive) to the point $r$ (inclusive).

The boundary probability $p_b(q)$ involves calculation of color and texture gradients. For the final boundary calculation the color and texture gradients are combined to get a single value. To calculate the boundary probability a sigmoid function with learned parameters is used. Further details regarding boundary estimation can be found in [20].

The calculation of color gradients and texture gradients for boundary calculation is a computationally expensive operation. The alternatives for boundary calculation is using gradient based edge detectors like Sobel, Prewitt, Roberts, Gaussian, Zero-cross and Canny. The benefit of using these simple gradients lies in their low computational times compared to boundary estimation. However, though computationally beneficial, they cannot achieve the same segmentation performance as the boundary (as shown in the Section 6).

Moreover for considering computational costs we can only use color based weight function using Equation 4, where boundary calculation is not needed. However in the Experiments section we show that taking texture into account for the weight function improves skin segmentation.

5 Pixel Weights

The pixels are connected to the two terminal nodes $F$ and $B$ which stands for foreground and background nodes and thus we can incorporate the information provided by the automatic or manual seed/template from the skin image $I$. From the seed/template, we incorporate the penalty for each pixel being “skin” or “non-skin”. We use intensities of pixels which are marked as seeds to get the histogram for foreground. The histogram for background is calculated from all image pixels. These histograms are then used to set the regional penalties as negative log-likelihoods. We use the face detector introduced by Viola et al. [33] for seed selection from image $I$. Seed selection and its settings affect the overall skin segmentation using graph-cuts. The confidence of pixel being skin can be increased by weight augmentation. Using weight augmentation we increase the weights of a pixel by adding its weights from a skin weighted image. The constraint of having a seed patch from the local image can be softened by opting for a universal seed which is obtained from the skin samples from different faces. First we explain the regional penalty calculation based on histograms. We then explain seed selection, weight augmentation and universal seed.

The regional penalty of a point as being “skin” (foreground) $F$ or “non-skin” (background) $B$, as defined in [21] is

$$R_{F|q} = -\ln p(B|c_q)$$

(8)

$$R_{B|q} = -\ln p(F|c_q)$$

where $c_q = (c_L, c_0, c_b)^T$ stands for a vector in $\mathbb{R}^3$ of L*a*b* values at the pixel $q$. Here we show it for the L*a*b* color space. To compute the posterior probabilities in Equation 8 we used Bayes theorem as follows [21]

$$p(B|c_q) = \frac{p(c_q|B)p(B)}{p(c_q)} = \frac{p(c_q|B)p(B)}{p(B)p(c_q|B) + p(F)p(c_q|F)}$$

(9)

For skin segmentation problem we first demonstrate it on $p(B|c_q)$, for $p(F|c_q)$ the steps are analogous. Initially we do not know a priori the probabilities $p(F)$ and $p(B)$ of the “skin” and “non-skin” regions, i.e. we do not know how large the “skin” region is compared to the “non-skin” one. Thus, we fix them to $p(F) = p(B) = 1/2$ as is also reported.
in [21]. After this assumption the Equation 9 reduces to
\[
p(\mathcal{B}|c_a) = \frac{p(c_a|\mathcal{B})}{p(c_a|\mathcal{B}) + p(c_a|\mathcal{F})} \tag{10}
\]
where the “skin” and “non-skin” prior probabilities are
\[
p(c_a|\mathcal{F}) = f_{L,a,b}^L f_{a}^a f_{b}^b \tag{11}
\]
and
\[
p(c_a|\mathcal{B}) = b_{L,a,b}^L b_{a}^a b_{b}^b \tag{12}
\]
where \( f_{i}^{(L,a,b)} \), resp. \( b_{i}^{(L,a,b)} \), represents the foreground, resp. the background histogram of each color channel separately at the \( i \)th bin.

We smooth all histogram channels by using one-dimensional Gaussians, i.e.
\[
\bar{f}_i = \frac{1}{G} \sum_{j=1}^{N} j f_j e^{-\frac{(i-j)^2}{2\sigma^2}} \tag{13}
\]
where \( G \) is the normalization factor enforcing \( \sum_{i=1}^{N} \bar{f}_i = 1 \). In the skin segmentation problem, the number of histogram bins \( N = 64 \) and \( \sigma = 1 \). These values are obtained experimentally for giving the best performance on a large database of images. Finally \( \omega = \lambda r_{i|q} \), where \( \lambda \) is controlling the importance of penalties for foreground and background and is set to 1000 as suggested in [21].

The prior probabilities for “skin” and “non-skin” are calculated from the histograms for “skin” and “non-skin”. The “skin” histogram is straight forward to compute and is computed from all the pixels in the template patch provided by the face detector. To compute the “non-skin” histogram is complex because there is no information about the background patch. It is assumed that the histogram computed from all pixels includes information on all colors (the “skin” and “non-skin”) in the image \( I \). We therefore compute the background histogram from all image pixels. Since \( \sum_{i=1}^{N} b_i = 1 \), the probability \( p(c_a|\mathcal{B}) \) gives smaller values than \( p(c_a|\mathcal{F}) \) for the “skin” colors present in the template. Thus, pixels more similar to the template are assigned in the graph more strongly to the “skin” than to the “non-skin” node.

5.1 Seed/Template As Foreground
In skin image \( I \) the pixel \( p \) is connected to terminal nodes. With this setup we can incorporate the information provided by the automatic or manual seed/template from the skin image \( I \). Manually a user can click on the image and select some small patch for skin from the image. We want to automate this process. For skin segmentation this is possible as we are using images having humans and thus probably faces visible in these images. We use the Viola Jones [33] face detector for automatic seed generation from images. The face detector finds faces and returns a rectangle around the face. The histogram for foreground is calculated from the face area. The histogram for the background is calculated from the whole image including the face area. The weights are calculated and assigned to the edges in a graph. Finally a graph cut segments the image into skin and non skin areas, as shown in Figure 4.

5.2 Pixel Weights and Skin Weights
From example images and the ground truth available for these images, we want to create a weighted skin model. This model is used to augment the pixel weights of skin graph. These weights are set on the terminal edges of the graph. These edges are connected to the source and sink nodes. The weighted model is generated in the YCbCr color space. We choose the YCbCr color space because the favorable property of this color space for skin color detection is the stable separation of luminance, chrominance, and its fast conversion from RGB. The images used to create a weighted skin model cover skin of different people having different skin tones, age, race and ethnicity group. Also images contain skin of the same person under different lighting conditions with the hope to make the model robust to skin tones and lighting conditions.

For model creation we create a 2-dimensional histogram with the Cr component of YCbCr on the x-axis, and the Cb component on the Y-axis from the manually labeled skin pixels. Given a pixel \( p \) in YCbCr color space, its Cb and Cr components are used to increment the weight at the corresponding position in the histogram. The procedure for the whole dataset results in an elliptical structure having weights for a particular Cb and Cr combination as shown in Figure 5. We want the skin detection to be independent of illumination and therefore omit the Y component.

The model in Figure 5 shows that skin covers a smaller well defined portion of the corresponding color space. The model obtained is somewhat similar in idea to the YCbCr and RGB static filters represented by Equation 1 and 2, reported in [7] and [24] respectively. The difference lies in the true representation of skin boundary. The static model constitutes a rectangular window as a filter, thus increasing the true positives, and at the same time as a result the false positives are increased. Our weighted approach reduces the false positives by creating an elliptical yet more natural and adaptive filter. This filter not only covers the true range of skin pixels but decreases false positives.

5.3 Weight Image and Pixel Weight Augmentation
We present the idea of a weight image. A weight image is created by replacing the pixel values with the weights of skin pixels. These weights are obtained from the skin weighted model in YCbCr color space. The weighted image can be used as a probability of skin. The images augment the skin detection process, where skin has to be segmented from the background. The higher the skin pixel weight the more probable that it is classified as a skin. The weighted image alone can be used to segment the skin.
The pixel weights are the probability weights added to the edges of the graph connected to source and sink nodes based on the seed histogram. Let’s say the weight assignment procedure assigned some weight $\omega$ to the edge $E$ of graph $G$, then the same pixel is found in the weighted image in the YCbCr color space. Its weight is retrieved from the weighted image and represented by $w_1$. Then the final new weight to be assigned to the edge $E$ will be

$$\omega_{\text{new}} = (\omega + w_1) \quad (14)$$

where $\omega_{\text{new}}$ is the new weight to be assigned to the edge of a graph $G$. We call it pixel weight augmentation. This augmentation of skin weights is only applicable to the source and sink edges and not to the neighborhood edges. The neighborhood weights or neighborhood edge weights are calculated as explained in the Section 4 and are not augmented.

### 5.4 From Local Seed to Universal Seed

We investigate the following: Can we, without having large amount of manually labeled ground truth, produce a seed that is as general as possible and can be used as a successful filter? The seed thus required should be able to represent all kinds of skin. We define such a seed as the universal seed.

We collect different skin tones from different faces, see Figure 6. For the universal seed calculation, different skin patches (faces) are aligned together to create one image. A face detector is used to get rectangle around the faces. The static filter is used to block the unwanted information which mainly is hair and in some cases the background information. So, a mask image is created which is pure skin patches extracted through the steps described. The foreground histogram in Equation 11 is calculated based on this mask. The histogram is saved for future use and we call such histogram as universal histogram. When a new image is to be segmented, the foreground histogram is loaded and no new histogram is calculated based on local information.

The difference of a universal seed and the skin weighted model is that universal seed is used directly to calculate weights while weighted model is used to augment these weights. The universal seed based skin detection technique is applied to images of people having different skin tones and taken under different lighting conditions and having body parts exposed. The segmentation result using the described universal seed is shown in Figure 7(b) and Figure 7(d). This concept is close to a skin filter bounded by the patches used for building the seed. The quality of skin segmentation using universal seed concept depends upon the proper selection of patches. A proper selection of skin patches improves the usability and its filtering ability. We show in the Experiments Section that the universal seed concept outperforms static filters and dynamic approaches.

### 6 Experiments

We performed number of experiments related to skin segmentation using graph cuts. First we explain the dataset used and then we present the comparison of weighted graph cuts technique with other skin segmentation techniques using F-score. The F-score calculation involves evenly weighting recall and precision. We also show the effect of different parameters on final skin segmentation in case of graph cuts.

#### 6.1 Data Sets

We have used images extracted from 25 videos provided by an Internet service provider that requires a skin detection application for their on-line platform. The sequences contain scenes with multiple people and/or multiple visible body parts and scene shots both indoors and outdoors, with steady or moving camera, see Figure 8. The lighting varies from natural light to directional stage lighting. Ground truth has been generated for all of the 25 videos on a per pixel basis. The data set is available one-line\(^1\).

We also collected images from the web using Google image search. For the special case of graph cuts we have used 300 images. There are 4.69 million skin pixels and 19 million non-skin pixels to be classified.

\(^1\)http://www.prip.tuwien.ac.at/people/julian/skin-detection
6.2 Static, Dynamic, Graph Cut and Weighted Graph Cut Approaches

The weighted graph cuts is compared with simple graph cuts where we do not augment the weights. The addition of weights using equation 14 increases overall segmentation performance, see Figure 9. The F-Score for weighted graph cuts and simple graph cuts is calculated using the CIELAB color space. Also color plus texture is taken into account and thus the natural boundary estimation. Figure 9 shows an improved F-Score when compared with the simple graph cuts, dynamic model and static model. The static model used is from YCbCr color space which is based on [7] and the boundary values used are given by Equation 1. The dynamic model used is from [15] and is based on model adaptation using faces. We conclude that addition of weights from the weighted image increases the overall efficiency of skin segmentation system.

6.3 Effect of Different Edge Detectors

An edge detector has a profound effect on the skin segmentation using graph cuts when used instead of boundary estimation for taking color and texture into account. The effect of using different edge detectors is shown using the overall F-Score, see Figure 10. The F-Score is calculated while using the CIELAB color space. The weight augmentation technique is also used. With these parameters, the highest F-Score is related to boundary probability technique and the lowest to the Prewitt edge detector method. A drawback of using boundary probability is the time it takes during calculation. While simple edge detector is 10 times faster than color plus texture, and thus can be adopted for real time detection of skin in videos.

6.4 Effect of Color Only and Color Plus Texture

We compared the two approaches which are color only and color plus texture for weight function. Figure 10 shows the F-score for texture, color only approach and different edge detectors. The F-Score is calculated using the CIELAB color space. The weight augmentation technique is also used. The boundary estimation is used for texture calculation rather than simple edge detectors. It can be noted that color plus texture has a high F-score compared to color only. The other advantage of using color to texture plus color is the execution times. The color only approach executes 10 times faster than color plus texture, and thus can be adopted for real time detection of skin in videos.

6.5 Universal Seed

The universal seed based skin segmentation is compared with static filters in YCbCr and RGB color spaces. The YCbCr and RGB static filters are represented by Equation 1 and 2, reported in [7] and [24] respectively. On the dataset the universal seed has a higher F-Score than YCbCr and RGB static skin filters as shown in Figure 11. The results in Figure 11 for the universal seed is not as good as Figure 9, because no local seed is obtained from images and the segmentation is based on universal seed which is bounded by the samples used. As a fall back method for images without faces, this universal seed is nevertheless useful.

7 Conclusion

We propose a principal approach combining graph cuts, color weighting and texture information to segment skin
regions from images. The concept of skin weights and the skin weighted image are introduced. It is shown that we improve the overall skin segmentation performance compared to other color based approaches. For more general skin detection i.e. without the information of foreground or background seeds, we advocate the idea of a universal seed. The effect of different edge detectors on graph based skin segmentation is evaluated. We showed that the combination of texture and color based weighting scheme improves skin segmentation compared to color only approaches.

In future we plan to work on universal seed adaptation and improving the over-all execution time.

Acknowledgment

This work was partly supported by the Austrian Research Promotion Agency (FFG), project OMOR 815994, the Higher Education Commission of Pakistan under Overseas scholarships (Phase-II-Batch-1) and the CogVis® Ltd. However, this paper reflects only the authors views; the FFG HEC or CogVis Ltd. are not liable for any use that may be made of the information contained herein.

References


2www.cogvis.at
<table>
<thead>
<tr>
<th>Author Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antúnez Esther 85</td>
</tr>
<tr>
<td>Bach Francis 5</td>
</tr>
<tr>
<td>Bhatti Naeem A. 21</td>
</tr>
<tr>
<td>Bischof Horst 7, 52</td>
</tr>
<tr>
<td>Bleyer Michael 60</td>
</tr>
<tr>
<td>Cohn Anthony 6</td>
</tr>
<tr>
<td>Gelautz Margrit 60</td>
</tr>
<tr>
<td>Hanbury Allan 21, 107</td>
</tr>
<tr>
<td>Holzer Peter 99</td>
</tr>
<tr>
<td>Hosni Asmaa 60</td>
</tr>
<tr>
<td>Khan Rehanullah 107</td>
</tr>
<tr>
<td>Kovačič Stanislav 29</td>
</tr>
<tr>
<td>Krüger Lars 44</td>
</tr>
<tr>
<td>Kristan Matej 29, 77</td>
</tr>
<tr>
<td>Kropatsch Walter G. 66, 72, 85</td>
</tr>
<tr>
<td>Leonardiš Aleš 77</td>
</tr>
<tr>
<td>Mörwald Thomas 91</td>
</tr>
<tr>
<td>Matas Jiří 37</td>
</tr>
<tr>
<td>Mateos Luis A. 72</td>
</tr>
<tr>
<td>Molina-Abril Helena 85</td>
</tr>
<tr>
<td>Pajdla Tomáš 15, 44</td>
</tr>
<tr>
<td>Perš Janez 29</td>
</tr>
<tr>
<td>Pinz Axel 99</td>
</tr>
<tr>
<td>Prantl Manfred 6</td>
</tr>
<tr>
<td>Richtsfeld Andreas 91</td>
</tr>
<tr>
<td>Roth Peter M. 7</td>
</tr>
<tr>
<td>Ruland Thomas 44</td>
</tr>
<tr>
<td>Schilling Tanja 15</td>
</tr>
<tr>
<td>Shao Dan 66</td>
</tr>
<tr>
<td>Stöttinger Julian 107</td>
</tr>
<tr>
<td>Sulić Vildana 29</td>
</tr>
<tr>
<td>Trefný Jiří 37</td>
</tr>
<tr>
<td>Vincze Markus 91</td>
</tr>
<tr>
<td>Wendel Andreas 52</td>
</tr>
<tr>
<td>Wohlhart Paul 7</td>
</tr>
<tr>
<td>Zillich Michael 91</td>
</tr>
</tbody>
</table>