Automatic segmentation of articulated objects of unknown complexity from videos

Jan Košata

kosatj2@fel.cvut.cz

CTU–CMP–2007–03

January 13, 2007

Available at
ftp://cmp.felk.cvut.cz/pub/cmp/articles/svoboda/Kosata-TR-2007-03.pdf

Thesis Advisor: Tomáš Svoboda

Research Reports of CMP, Czech Technical University in Prague, No. 3, 2007

Published by

Center for Machine Perception, Department of Cybernetics
Faculty of Electrical Engineering, Czech Technical University
Technická 2, 166 27 Prague 6, Czech Republic
fax +420 2 2435 7385, phone +420 2 2435 7637, www: http://cmp.felk.cvut.cz

MASTER THESIS

ISSN 1213-2365

České vysoké učení technické v Praze - Fakulta elektrotechnická

Katedra kybernetiky ***Školní rok:*** 2005/2006

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Jan Košata

Obor: Technická kybernetika

Název tématu: Automatická segmentace artikulovaných objektů o neznámé komplexitě ze sekvence obrazů

Zásady pro vypracování:

Automatická segmentace sekvence obrazů na statické pozadí a pohybující se objekty je v oblasti počítačového vidění dobře prozkoumána a existuje mnoho použitelných algoritmů (motion segmentation). Nás zajímá automatická segmentace objektu, který se skládá z několika částí navzájem pospojovaných (artikulovaný objekt). Typickým zástupcem podobného objektu je člověk, nebo komplikovanější stroj.

Cílem práce je implementace metody inspirované [1], její otestování na reálných datech a navržení případného vylepšení. Vstupem algoritmu je sekvence obrazů nebo video, výstupem pak jednotlivé části artikuloveného objektu. Práce je součástí projektu MultiCam [5], v jehož rámci se vyvíjejí algoritmy počítačového vidění pro analýzu dynamické scény.

Seznam odborné literatury:

[1] M. Pawan Kumar, P.H.S. Torr a A. Zisserman: Learning Layered Motion Segmentation of Video. International Conference on Computer Vision ICCV, Oct 2005, IEEE-CS Press.
[2] P.F. Felzenschwalb, D.P. Huttenlocher a J.M Kleinberg: Fast Algorithms for Large-Statespace HMMs with application to Web Usage Analysis. Advances in Neural Information Processing Systems, 2003.
[3] J. Pearl: Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann, 1988.
[4] Y. Weiss: Belief Propagation and Revision in Networks with Loops. AI Memo No. 1616,MIT AI Lab, November 1997.
[5] Projekt MultiCam, GA CAV 1ET101210407, http://cmp.felk.cvut.cz/projects/multicam/

Vedoucí diplomové práce: Ing. Tomáš Svoboda, Ph.D.

Termín zadání diplomové práce: zimní semestr 2005/2006

Termín odevzdání diplomové práce: leden 2007

[illegible]

prof. Ing. Vladimír Mařík, DrSc.
vedoucí katedry

L.S.

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta elektrotechnická -4-

v zast. [illegible]

prof. Ing. Vladimír Kučera, DrSc.
děkan

V Praze dne 18.01.2006

Abstract

Motion segmentation is an important topic in computer vision. Unlike simple motion segmentation articulated segmentation gives much more information about observed scene and makes easier classification of the observed object, i.e. by kinetic patterns. We are interested in automatic segmentation of an articulated object with an unknown complexity. Typical representative is a human being or a more complicated machine. We implemented the method described in M. Pawan Kumar, P.H.S. Torr a A. Zisserman - Learning Layered Motion Segmentation of Video. This algorithm is applicable to any video containing piecewise rigid motion. Given the sequence, the generative model which best describes an articulated object and a background is learnt in an unsupervised manner. The generative model for a layered representation describes the scene as a composition of layers.

Segmentace videa je duležité téma v oblasti počítačového vidění. Na rozdíl od jednoduché segmentace pohybu segmentace artikulovaných částí přináší mnohem více informací o pozorované scéně a umožuje snazší klasifikaci pozorovaného objektu například podle pohybových vzorcu. Nás zajímá automatická segmentace artikulovaných objektu o neznámé komplexitě. Typickým zástupcem takového objektu je člověk nebo komplikovanější stroj. Implementovali jsme metodu popsanou v M. Pawan Kumar, P.H.S. Torr a A. Zisserman - Learning Layered Motion Segmentation of Video. Tento algoritmus se dá použít na libovolné video obsahující po částech rigidní pohyb. Ze vstupní sekvence je automaticky získán generující model, který popisuje jak objekt, tak pozadí. Generující model vrstvové representace popisuje scénu jako kompozici vrstev.

Statement

This is to certify that I wrote this thesis on my own and that the references include all the sources of information I have utilized.

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

[illegible]

V Praze dne 2.1. 2007 ……………………………………

podpis

Acknowledgement

I would like to thank to my supervisor T. Svoboda PhD for leadership of the diploma thesis, for his advices and comments. I am also grateful to M. Pawan Kumar for his advices and suggestions during personal communication. Finally but not least I would like to thank to my wife, family and everybody who supported my education.

Automatic segmentation of articulated objects of unknown complexity from videos

Jan Košata

January 13, 2007

Contents

List of Figures

Chapter 1

Introduction

A motion segmentation of a video is well known in computer vision. The simplest method is pixel based motion detection without correspondence search – [2], [12]. This algorithm detects a motion between two frames of video by comparing values of corresponding pixels of two frames. The result of this method gives only information of the motion. The essentially more challenging task is to not only segment the pixels but also classify them to rigidly moving parts, see Figure 1.1.

Articulated objects, e.g. pedestrians, are often very difficult to recognize from a scene due to a large variability in their local and global appearance. It is caused by various types and styles of clothing, so that only few local regions are really characteristic for the entire category, more information can be found in [7]. As can be seen on the Figure 1.1 simple motion segmentation inform only about moving pixels. A silhouette can be acquired from this information but sometimes it could be difficult task to recognize object only from this shape. Properties of a silhouette, i.e. area, circumference or number of the curves can represent input to classifier, which should recognize object on scene. However, these simple properties sometimes are not enough. Articulated motion segmentation gives us much more information of the scene. From articulated motion segmentation we acquire not only segmented moving object but also information describing motion of each part. This information may be used to learn kinetic patterns belonging to the individual parts of the articulated structure and better recognition of the scene.

There are two main groups of approaches for motion segmentation without correspondence search; background model based [4] and those based on consecutive frames (optical flow)[13], [9]. Comparison of the current frame of the sequence with the background model is very popular way of the motion segmentation. The idea is to create and maintain a background model.

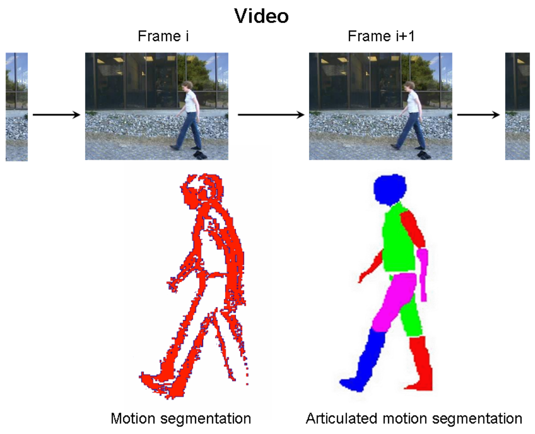


Figure 1.1: Simple motion segmentation gives only information about motion in a picture. On the other hand from articulated motion segmentation we acquire not only segmented moving object but also information describing motion of each part. This information with knowing kinetic patterns leads to a better recognition of the scene. A frame sequence taken from a static camera [11]. The scene consists of a person walking against a static background. Picture of the segmented body taken from [5].

If a new object occurs in foreground, it can be found as inconsistency with the background model. Certain complications arise from a fact, that it can be nearly never assumed that background is static; it's getting darker, sun becomes occluded by clouds, or shadows are moving. Therefore algorithms for background subtraction that use adaptive background [2] model are more successful than those based on the assumption of static background [4].

The methods based on optical flow [13] calculation can be used to detect independently moving objects even in the presence of camera motion and gives very good results. Motion segmentation methods, which are based on the optical flow, are computationally complex and hence not suitable for realtime systems without specialized hardware.

Many different approaches for motion segmentation have been reported. The situation, if the model occlusion is not used (i.e. described in [3]), can lead to problems such as overcounting the data when obtaining the segmentation. Different segments may have different depth. It could cause occlusion of some segments. This problem can be resolved by using a layered representation. However, this method usually relies on a keyframe for the initial estimation. All the information of the layered representation we estimate from the sequence of a video. The next important issue is to use spatial continuity to avoid non-continuous segmentation when foreground and background are similar in appearance. This issue is noted in [15].

In our work, we use a model which does not suffer from the problems mentioned above. The implementation of method was inspired in [5]. An initial estimate of the model is obtained by dividing the scene into rigidly moving components using efficient loopy belief propagation. We use the set of components obtained from all pairs of consecutive frames in a video seqeunce which are later combined to get the initial estimate of the segments. This avoids the problem of finding only those segments which are present in one keyframe of the video. Given this estimate, the shape of the segments, along with the layering, are learnt by minimizing an objective function using $\alpha\beta$-swap and α-expansion algorithms [1].

We tested the algorithm by simple motion and real data. Input of the algorithm is video, output are individual parts of the articulated object. This algorithm can be used for the coarse estimation of the rigidly moving parts in video.

1.1 Evolution of the diploma thesis

The algorithm deals with layered representation so the code for generation the testing data from layered representation was needed. We generate a few types of the testing data with simple motion (see Fig. 4.1).

The implementation starts with the part Initial estimation of the parameters, described in section 2.3. Actual progress of the implementation was tested by simple data and then with real data. We had to get over a few unclearness in [5] and contacted Pawan Kumar. We spent quite a lot of time with understanding the LBP in this implementation and Refining shape.

The final step of the algorithm should perform refining shape of the segments. We tried to adapt codes from [1] and from Pawan Kumar previous work but we did not suceed. Graph construction from picture into graph cut minimization requires more information than we found in [5] and [1]. The result of our algorithm are coarse segments of the rigidly moving parts of a video and their transformation.

1.2 How to read a thesis

The diploma thesis is divided into the following chapters:

- **Chapter 1 – Introduction** deals with a motion segmentation and articulated motion segmentation.

- **Chapter 2 Layered image representation** describes theory of the algorithm and its implementation.
- **Chapter 3 Implemetation issue** contains description of the problems we met at implementation and their solutions.
- **Chapter 4 – Experiments** contains tests and experiments that we used during the implementation.
- **Chapter 5 – Conclusion** sums up work done, discusses fulfillment of the goals.

Chapter 2

Layered image representation and estimation of its parameters

2.1 Layered representation

Layered representation is a generative model that can describe and generate any frame of a video from set of layers and their transformation parametres, see Figure 2.1. The goal of this work is to estimate such a model from a video. A frame is divided to layers according to its appearance. Each rigidly moving part of the frame has its own layer. For a full description of the set of frames by a layered representation we also need transformations. That includes a scale, a rotation and translations to both directions x and y.

When we generate a frame j, we start from the blank image and map every point $\mathbf{x}$ from the respective layer using the transformations a scale, a rotation and translations in both directions. It means that points belonging to the same segment move rigidly together. The generated frame is then obtained by composing the transformed segment in proper order of their layer numbers. For this work, each transformation has four parameters: scale, rotation and translations in direction x and y. The original method [5] models also motion blur and illumination changes.

2.2 Overview of the algorithm

Given a video, its layered representation is learnt in an unsupervised manner by minimizing the energy of the representation in four stages. The first stage is initial estimation of the parameters. Inter–frame motion is estimated

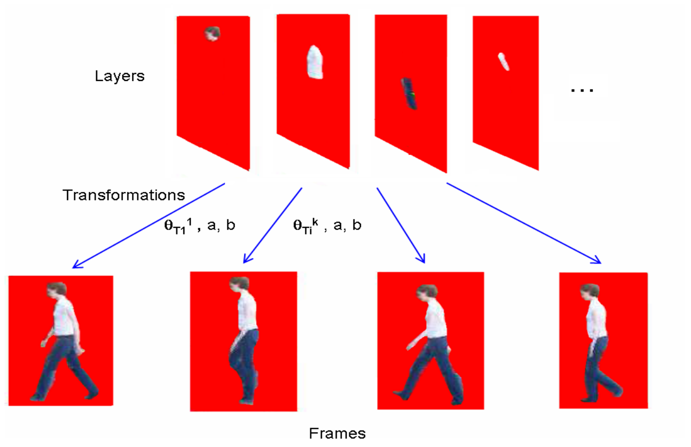


Figure 2.1: The top row of images shows individual layers, bottom row final frames generated from layers and information of transformations. Θ_{Ti}^{k} denotes set of transformations scale, rotation and translations of segment i to generate part of frame k, a and b are lighting parameters.

for each pair of consecutive frames. This provides us with the rigidly moving connected components of the patches between each pair of consecutive frames. These components are then clustered according to their appearance to obtain an initial estimate of the model parameters. The second stage is refining shape of the segments. The initial estimation of the shape of the segments is refined by the energy minimization. Following stage is updating appearance. The appearance of the points added to the matte of a segment in the second stage is obtained. The point is simply assigned the mean of its observed RGB values over all frames in the video where it is visible. The last stage is refining transformation. The transformations are refined by searching around their initial estimate. The transformation resulting in the least sum-of-squared differences is chosen.

2.3 Initial estimation of parameters

The goal of this part of the algorithm is to find initial parameters of each layer. This part is described in paragraph 3.1 in [5] and for the sake of clarity we repeat the main points here. In order to obtain an initial estimate of the model parameters, we obtain rigidly moving components between every pair of consective frames. This involves computing the image motion for every pixel in frame n to frame $n+1$. However, at this stage we are only interested in obtaining a coarse estimate of the model parameters which we refine later.

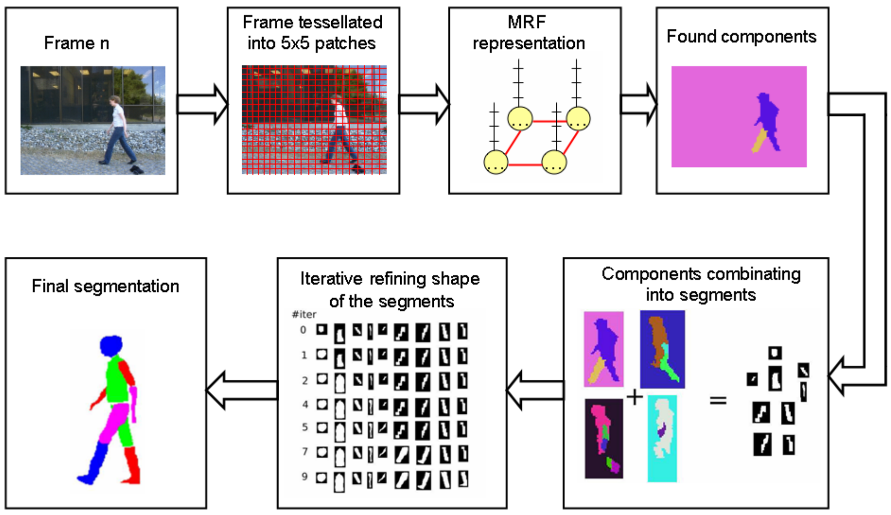

Frame n
Frame tessellated into 5x5 patches
MRF representation
Found components
Components combinating into segments
Iterative refining shape of the segments
#iter
0
1
2
4
5
7
9
Final segmentation

Figure 2.2: The complete algorithm: Input of the algorithm are frames from a video. Given a frame, algorithm divides it into uniform patches and generates MRF representation. Every patch is represented by site of the MRF and every putative transformation of every patch is represented by label of the MRF. In next step rigidly moving components are found by Loopy Belief Propagation. These tentative components with similar transformation are clustered into segments which are refined. Refining of the shape is performed by algorithms minimizing graph cuts, specifically by the $\alpha\beta$-swap algorithm and α-expansion. The output of the algorithm are individual parts of the articulated object separated into layers and their transformations. Some of these pictures were taken from [5]

Firstly we divide each frame into uniform patches of size $m \times m$ pixels, we used $m = 5$. To track the patches from frame n to $n+1$ we define a Markov random field (MRF) over the patches. Each site of the MRF corresponds to one patch in the frame. The labels of a site correspond to the putative transformations of that patch. The likelihood of a label is given by the cross-correlation of the corresponding patch in frame n, after undergoing the transformation specified by that label, with frame $n+1$. Since we are interested in finding rigidly moving components, we specify the prior that neighboring patches tend to move rigidly together. The neighborhood of a patch is its 4-neighbourhood. We are looking for the transformation of each patch from the one frame to the next one that maximizes joint probability

$$Pr(\varphi) = \frac{1}{Z} \prod_k \psi(s_k) \prod_{n_l \in \mathcal{N}_k} \psi(s_k, s_l), \tag{2.1}$$

where φ is a set of transformations, k is the index through all transformations, $\mathcal{N}_k$ is the 4-neighbourhood of the patch k, s_k and s_l are the labels of the transformation, $\psi(s_k)$ is the likelihood of the transformation s_k, $\psi(s_k, s_l)$ is a pairwise potential, which will be discussed later.

We use loopy belief propagation (LBP) for finding this transformation from the set of putative transformations. At the beginning of the algorithm we define range of the putative transformations, number of the frames and the resolution of each patch. We calculate likelihood $\psi(s_k)$ for each patch and each transformation by equation

$$\psi(s_k) = e^{\mathcal{L}(\mathbf{f_k}, \psi_k)}, \tag{2.2}$$

where $\mathcal{L}(\mathbf{f_k}, \psi_k)$ is the cross-correlation of the transformed patch and the relevant patch from the next frame. Cross-correlation is calculated by equation

$$\mathcal{L}(\mathbf{x}, \mathbf{y}) = \frac{(\mathbf{x} - \overline{\mathbf{x}})^{\mathrm{T}}(\mathbf{y} - \overline{\mathbf{y}})}{\sqrt{(\mathbf{x} - \overline{\mathbf{x}})^{\mathrm{T}}(\mathbf{x} - \overline{\mathbf{x}})(\mathbf{y} - \overline{\mathbf{y}})^{\mathrm{T}}(\mathbf{y} - \overline{\mathbf{y}})}}, \tag{2.3}$$

where $\mathbf{x}$ is the vector of the RGB values of the pixels from the patch from the actual frame and $\mathbf{y}$ is the vector of the RGB values of the pixels from the patch from the next frame transformed by transformation ψ_k. Linear correlative coefficient $r(x, y)$ (normalized cross-correlation) denotes how much values of vectors $\mathbf{x}$ and $\mathbf{y}$ are bounded by linear function. If they are accurately bounded by linear function with positive derivation then $r(x, y) = 1$, with negative derivation $r(x, y) = -1$. Generaly can be said $-1 \leq r(x, y) \leq 1$.

The beliefs are calculated by equation

$$b(s_k) = \psi(s_k) \prod_{n_l \in \mathcal{N}_k} m_{lk}^t(s_k). \tag{2.4}$$

Variable $m_{lk}^{t}(s_k)$ is the message, that is sent by patch l to patch k in iteration t and is calculated by equation

$$m_{kl}^{t}(s_l) = \sum_{s_k} \left(\psi(s_k, s_l)\psi(s_k) \prod_{n_d \in \mathcal{N}_k \setminus n_l} m_{dk}^{t-1}(s_k) \right), \tag{2.5}$$

where $\psi(s_k)$ is likelihood calculated in previous part of code (2.2) by the equation:

$$\psi(s_k, s_l) = \left\{ \begin{array}{cl} 1 & \text{if rigid motion} \\ \exp(\zeta \nabla(\mathbf{f_k}, \mathbf{f_l})) & \text{otherwise} \end{array} \right. , \tag{2.6}$$

where $\nabla(\mathbf{f_k}, \mathbf{f_l})$ is the sum of the gradients of the neighboring pixels $\mathbf{x} \in \mathbf{f_k}$ and $\mathbf{y} \in \mathbf{f_l}$, where value of the pixel brightness is between 0 and 1. $\mathbf{f_k}$ is the fragment that we are interested in and $\mathbf{f_l}$ is the neighbouring fragment. ζ is normalization constant, we used $\zeta = 1$. There is a typo in sign in this equation in the original paper [5]. The proper sign in exponent should be plus.

All messages (2.5)are initialized to 1 in iteration 0 for all k and l. In every step of iteration all messages must be normalized by $\sum_{s_l} m_{kl}^{t}(s_l) = 1$. The termination criterion is that the maximum of change of all beliefs falls below certain threshold. We used threshold 10^{-4}. In one iteration we must calculate all messages and look for the maximal change of beliefs of all patches. When iterations stop, we find the transformations with the maximal belief for each patch.

In next step we color the patches with same transformations by same color and find connected components with same color. Components smaller then 100 pixels are merged with their surrounding. Each patch of the component appointed for merging searches its neighborhood and acquires the color of the majority neighbor with different color. Merging has two stages: before finding components is merged isolated patches with different color than the surrounding (by the same way as described above). If this first stage was not used there would form imaginary component on background surrounded by isolated patches (see figure 4.8. The second stage is merging founded components smaller than 100 pixels.

2.3.1 Loopy Belief Propagation

For sake of clarity we describe here main points of Loopy Belief Propagation (LBP). Details can be found in [14]. LBP is a message passing algorithm used to find the MAP maximizing (2.1).The rules of the local belief propagation without loops were proposed by [10]. This method is guaranteed to converge to the optimal (maximal) beliefs. It was empirically demonstrated, that the same algorithm has a good performance on networks with loops, but a theoretical understanding of this performance has not yet been achieved [14].

Finding the optimal beliefs of all sites is reached iteratively. For a singly connected unit, the beliefs vector is guaranteed to converge to the posterior probability of the node n_l given all the evidence. In every iteration the messages are sent among all the neighboring sites. Now we explain calculation of the messages according to (2.5). In iteration 0 all messages are initialized to 1. Sending messages in next iterations are subjected to following rules. The message that site n_k sends to site n_l is calculated as follows:

- Combine all messages coming into n_k except coming from n_l into vector v. The combination is done by multiplying all the messages vectors element by element.
- Multiply vector v by the likelihood of the site n_k and pairwise likelihood corresponding to the link between n_k and n_l (see equation 2.6).
- Normalize all the messages so they sum to 1. The normalized vector is send to n_l.

These Pearl's propagation rules [10] were designed for Bayesian networks, while here we are interested in Markov networks. Bayesian networks have arrows on the links, and of any two neighboring nodes, one is considered to be the cause and the other is the effect. Markov networks replace the notion of causal link, with a weaker notion of compatibility which is symmetric in the two neighboring nodes. The probability distributions represented by Markov networks are a subset of those that can be represented by Bayesian networks.

Another difference between the algorithm presented here and Pearl's original algorithm is in the normalization. In Pearl's algorithm messages going in one direction are normalized while those in the other direction are not. As pointed out by Pearl [10], the normalization step does not influence the final beliefs but ensures the stability of the message passing scheme.

The belief revision rules described here are equivalent to these described by Pearl except for the normalization. In Hidden Markov Models, belief revision is equivalent to the Viterbi update rules, and is a special case of concurrent dynamic programming. As can be seen from the previous discussion, update procedure of the type described by Pearl have been analyzed in many areas of optimization and applied mathematics. However, according to [14], in all these contexts the network is assumed to be singly connected. These procedures are well defined for any Markov network – all it requires is a decomposition of the posterior probability into pairwise compatibilities.

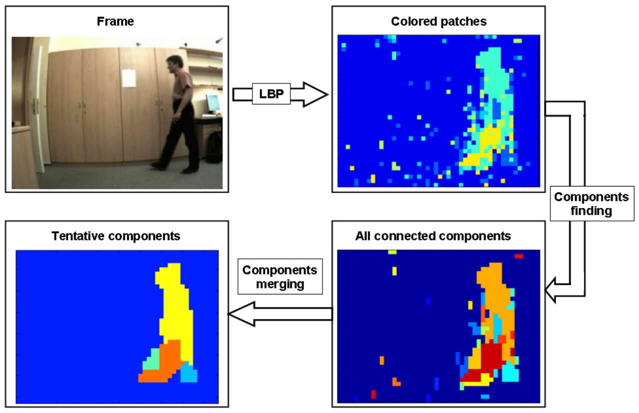


Figure 2.3: Progress of the part Initial estimation of parameters. From input frames the patches with the same transformation are colored by LBP. The patches with same transformation have same color. Connected components are found from this data and components smaller then 100 pixels are merged into surroundings.

2.4 Combining components

Now we have rigidly moving components of each inter-frame and we combine these components through all frames. We define similarity matrix S of size $c \times c$, where c is total number all rigid components of all frames. We compute $S(i, j)$ as normalized cross-correlation of segments i and j. The number of pixel used to compute $S(i, j)$ is the number of pixels in the smaller component and from the bigger component we use the part of component that maximize $S(i, j)$. $S(i, j)$ is computed only for a small number of components pairs (i, j) due to a several reasons. First, the similarity matrix is symmetrical, second, we compute $S(i, j)$ only for components, which overlay of these components is bigger than half of the smaller one. For components that lies far from each other we define $S(i, j) = -1$.

Now we find $(i*, j*) = \arg\max S(i, j)$ and we say $i*$ and $j*$ belong to same cluster. We repeat this until $S(i*, j*) < threshold$, we use threshold between 0.8 and 0.95. If we set low threshold (about 0.8) all components are in same cluster and segment is the least one (only one or two segments). If higher threshold (about 0.95) is used, no greater component (body) is clustered and there are a lot of segments. Number of clusters gives us number of segments and the initial shape of the segment is the least component of every cluster.

Method described above was recommended by from Pawan Kumar [6].

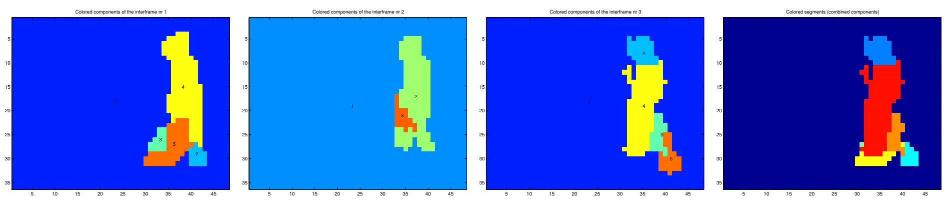


Figure 2.4: The result of the combining components: the first three pictures show tentative components, the last image is result of the components combining. This last image is input to the part Refining shape.

This procedure works properly for some examples, but not for many. Normalized cross-correlation of smaller components gives bigger values then with greater components. We need to preference clustering of the bigger components, so we used weighting function in computing similarity matrix S. Every element different to minus one in matrix is computed by (2.7):

$$S^*_{ij} = S_{ij}(1 - \alpha\sqrt{\max_m C_m / C_i}), \tag{2.7}$$

where S^*_{ij} is new value of the matrix, S_{ij} is original value, α is constant, we used $\alpha = 0,1$, $\max_m C_m$ is size of the greatest component and C_i is the size of the component i. Using this modified procedure we achieved better results (see figure 2.4 and 4.10)

2.5 Refining shape

2.5.1 Overview

We now describe a method to refine the shape of the segments and determine their layer numbers. The refinement is carried out such that it minimizes the energy of the representation. It takes advantage of efficient iterative algorithms for multi-way graph cuts [1]. It iterates over segments and refine the shape of one segment at a time. At each iteration, an $\alpha\beta$ -swap is performed for the segment and its surrounding segments (i.e. segments lying close to it). The $\alpha\beta$-swap algorithm swaps the points belonging to two segments thereby reassigning the points which were wrongly labelled as belonging to a particular segment. Next we perform an α-expansion algorithm which assigns the points surrounding the initial estimate of the segment to it. At each iteration, we also consider two options for overlapping segments, i.e. either segment A occludes segment B or vice versa, and chose the option that results in lesser energy. The iteration stops when the energy of the model cannot be minimized further. After termination, the refined estimate of the shape of the segments is obtained as well as their layering.

2.5.2 Energy minimization via graph cuts

Given an initial coarse estimation of the segments, their shape can be iteratively improved using consistency of motion and texture over the entire video sequence. The refinement is performed such that it minimizes the energy of model using efficient algorithms for multi-way graph cuts which minimizes an energy function over point labellings h of the form

$$\Psi = \sum_{\mathbf{x}\in\mathbf{X}} D_x(h_x) + \sum_{\mathbf{x},\mathbf{y}\in\mathcal{N}} V_{x,y}(h_x, h_y), \tag{2.8}$$

under fairly broad constraints on D and V. Here $D_x(h_x)$ is the cost for assigning the label h_x to point x and $V_{x,y}(h_x, h_y)$ is the cost for assigning h_x and h_y to the neighboring points $\mathbf{x}$ and $\mathbf{y}$ respectively. Especialy, we make use of two algorithms: $\alpha\beta$-swap and α-expansion.

The algrithm iterates over segments and refine the shape of one segment p_i at a time. At each iteration, we perform an $\alpha\beta$-swap for p_i and each of its surrounding segments p_k. This relabels all the points which were wrongly labelled as belonging to p_i. Then an α-expansion algorithm is performed to expand p_i to include those points $\mathbf{x}$ in its limit, which move rigidly with p_i. We choose the option which results in the minimum value of the energy. This determines the occlusion ordering among surrounding segments. We stop iterating when further reduction of the energy is not possible. This provides us with a refined estimate of the shape along with the layer number l_i of the segments.

The $\alpha\beta$-swap algorithm swaps the assignments of certain points $\mathbf{x}$ which have label α to β and vice versa. In this case, it attempts to relabel points which were incorrectly assigned to segments p_α or p_β. Each of the two points $\mathbf{x}$ and $\mathbf{y}$ are represented by one vertex in the graph. In addition, there are two special vertices called the source and the sink which represent the labels α and β. In this representation the source is represented by the pixels of the refined segment and the sink by the pixels of the surroundings segments. The graph cut algorithm perform finding the minimal graph cut, in this case the refined shape of the segment.

2.6 Implementation of the algorithm

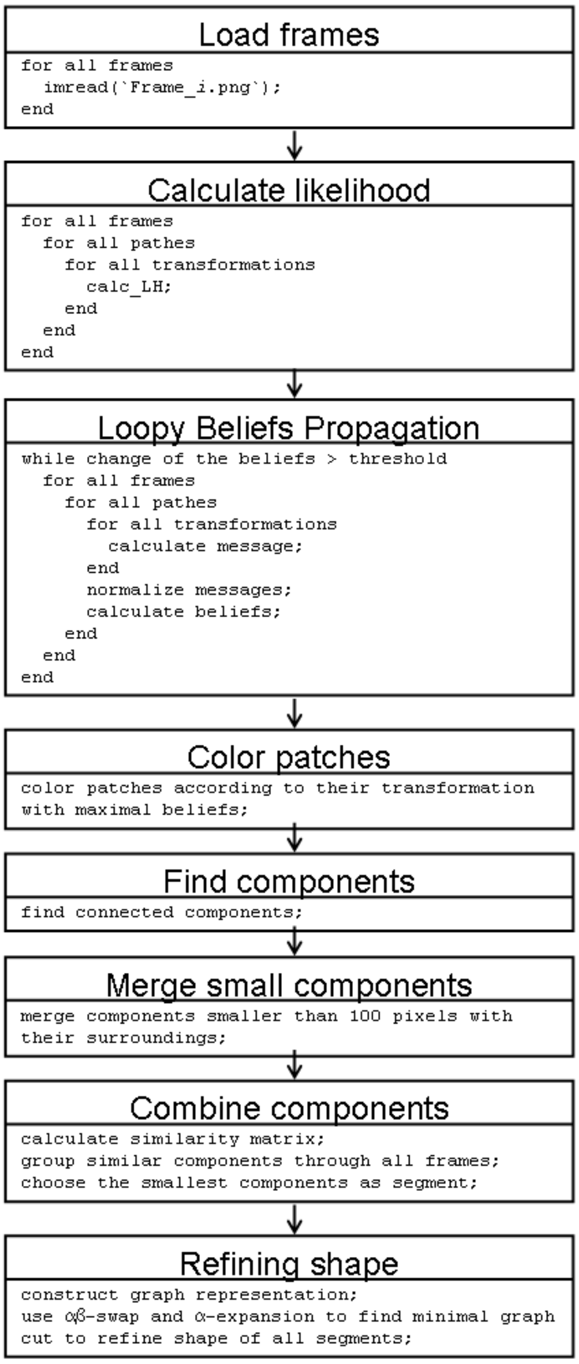
Almost the entire algorithm is implemented in Matlab. The final part Refining shape using graph cut algorithm is written in C++ as a MEX-file. Figure 2.5 shows pipeline diagram with symbolic code fo the implementation.

After loading all the frames into array the likelihood of all transformations for each patch is calculated by the function `Calc_LH.m` according to the equation (2.2). Result of this calculation is saved and later used for calculation of the messages according to equation (2.5) and can be found in m-script `Calc_MSG.m`. Result of the LBP is only beliefs of each transformation of all patches. From this data the maximal beliefs are found and colored according to color that represent the transformation with maximal belief. The information about the transformation of each patch is saved to array `Transformation(k,j,i,transf)` where `k` is number of the inter-frame, `j` and `i` are positions of the patch in the frame and `transf` determine type of the found transformations. `transf`=1 means scale, `transf`=2 rotation, `transf`=3 translation in axis x and `transf`=4 y translation. For our information we display this in `meshgrid` graph by m-script `Disp_Transf.m`.

The patches with same transformation of each inter-frame are grouped into connected components by m-script `Components_finding.m`. The core of the connected component finding algorithm is performed by the Matlab function `bwlabel` adjusted for multi-color labels.

In next step the components smaller then 100 pixels are merged with their surrounding by the m-script `Components_merging.m`. The components greater then 100 pixels are combined into segments through all frames by the m-script `Components_combining.m`. Combination is performed by the algorithm using similarity matrix of the all components of all inter-frames.

The final step of the algorithm should perform refining shape of the segments. This would have been implemented partially in Matlab code and partially in C++ as MEX file due to time complexity. We tried to adapt codes from [1] and from Pawan Kumar previous work but we did not succeed. Graph construction from picture into graph cut minimization is too difficult and partially implemented code from Pawan Kumar has a lot of unclearness and doesn't work in our Matlab implementation. The result of our algorithm are coarse segments of the rigidly moving parts of a video and their transformation.



```
for all frames
  imread(`Frame_i.png`);
end
```

```
for all frames
  for all pathes
    for all transformations
      calc_LH;
    end
  end
end
```

```
while change of the beliefs > threshold
  for all frames
    for all pathes
      for all transformations
        calculate message;
      end
      normalize messages;
      calculate beliefs;
    end
  end
end
```

```
color patches according to their transformation
with maximal beliefs;
```

```
find connected components;
```

```
merge components smaller than 100 pixels with
their surroundings;
```

```
calculate similarity matrix;
group similar components through all frames;
choose the smallest components as segment;
```

```
construct graph representation;
use αβ-swap and α-expansion to find minimal graph
cut to refine shape of all segments;
```

Figure 2.5: The pipeline diagram of the algorithm implementation. Blocks in the diagram denotes the individual parts of the algorithm and symbolic code of the implementation.

Chapter 3

Implementation issues

This chapter contains description of the problems we met at implementation and their solutions. The main issue of our implementation is computation complexity. This issue was partially solved for computation of the likelihood transformation and LBP by coarse to fine strategy. Other issues are mostly caused by arduousness of the algorithm and wrong understanding of the [5].

3.1 Computation complexity of the likelihood estimation

Efficient implementation of the likelihood computation (2.2) is an important issue. The likelihood is computed for each patch and each transformation. There is a large number of transformations to calculate, i.e. if we use the same set of putative transformations like in [5] (scale from 0.8 to 1.2 in steps of 0.2, rotation from -0.3 to 0.3 in steps of 0.15, translations in x and y directions from -5 to 5 pixels and -10 to 10 pixels respectively in step of 1) it would be 3465 transformations for each patch. We compare two possible ways to implement image transformations. The first is to use Matlab commands `imrotate`, `imscale` and `imcrop` for each patch. Commands `imrotate`, `imscale` and `imcrop` perform image transformations by inner Matlab implementation. The second manner is to calculate matrix of transformations, through it calculate new position of each pixel and use Matlab command `interp2` to calculate transformed image. Command `interp2` performs general 2D interpolation of the pixel RGB value for point laying off the grid. It would be more universal to use the matrix of transformations, but there was a time problem with using command `interp2`. Execution of this command takes quite to much time so finally algorithm ran for long time. Exactly for set of putative transformations described above it takes about 4.5 second for one patch at processor AMD Sempron 3000+, it is about 40 hours of calculating time for general image with 213x160 patches.

The best way what we found is to use commands `imrotate` and `imscale` for necessary operations, it means for *rotation* $\neq 0$ and *scale* $\neq 1$ for patches from the first frame and simply copying of shifted patches from next frame. This solution was 10 times faster, because it computes the transformations not for all causes. General shift the patch window is much more effective then using transformation command.

3.2 Understanding equation of messages calculation

We spent quite a bit of time to understand equation for message computation the LBP algorithm (2.5), it is equation (11) in [5]. Question was through what variable we should add at first sum, how they recognize rigid motion and what the maximal value of pixel intensity is in calculation of gradient. At last we asked Pawan Kumar and he explained us main issues of this equation. There was a typo in eqation (11) in [5], proper equation is rewritten in this paper: equation (2.5). We defined rigid motion by simple thresholds. Two motions are considered as rigid, if their translations are the same or different by 2 points, scale is the same and no rotation.

3.3 Normalization of the likelihood

There is a question of normalization the likelihood. By default, sum all values of likelihood $\psi(s_k)$ through all transformations s_k should be equal 1 ($\sum_{s_k} \psi(s_k) = 1$), so we normalize likelihood equation, but after consultations it with Mr. Pawan Kumar we leave it un-normalized.

3.4 Overflowing and underflowing of beliefs

As it can be seen in equation of the messages (2.5) in every next step of iteration we multiply previous messages. If the values of all messages were less then 1 after a few iterations the messages and beliefs fall bellow the floating point accuracy. Over against if the values of the beliefs were greater then 1 there would be overflowing of beliefs. We adopt normalization suggested by Yair Weiss [14]. The messages are normalized in every step of the iteration to 1 through all transformations of one patch. This normalization leads to better results without overflowing and underflowing.

3.5 Reducing complexity of LBP - Coarse to fine strategy

The time complexity of LBP is $O(nH^2)$, where n is number of sites, i.e. the number of patches, and H is number of labels per site, i.e. the number of putative transformations. This quadratic dependence implies that we can not use large set of putative transformation. For example if we use entire set of putative transformations described above (3465 transformations), the computing time of two relative small frames (240×180 pixels) is about 80 hours.

The way how to resolve this issue is to use coarse to fine strategy. The idea of this strategy is group together similar transformations that differs only in translation. The set of translations of each direction is divided into two groups. Then the representative $\psi(s_k)$ and $\psi(s_k, s_l)$ of each group is found that is $\psi(s_k)$ with maximal value and by LBP the group with maximal belief is chosen. This group is divided into smaller groups by same way and repeat the algorithm LBP until we have group with basic dimension -single label. This strategy is effective only for large set of the translations, not for large set of rotations and scales, because dilated is only set of translation. In general use it is enough, set of putative translation is much larger than set of rotations or scales.

This strategy leads to better time complexity. Using it we reach computing times about 10 minutes.

Chapter 4

Experiments

For testing we used firstly testing data (see Figure 4.1), what include static background with texture and translating rectangle with different texture. Results of this first experiments you can see on figure 4.3. For next experiments we use simple real data: two frames of video of moving people (see Figure 4.2). We found this frames in [5]. In case of time demand factor of the algorithm we used pictures with low resolution 120×80 pixels. For the other experiments are used photos from laboratory G9 of Czech Technical University, the part of the MultiCAM project.

4.1 Testing likelihood calculation

After calculating likelihoods we wanted to test results of initial estimation based solely on likelihood $\psi(s_k)$ of every patch. In this step we estimate transformation by maximum of $\psi(s_k)$ of every patch (see equation 2.2). The result you can see on Figure 4 testing data (moving textured rectangle). This estimation brings good results for simple data (i.e. our testing data), but it cannot be used for more complicated cases due to motion blur etc.

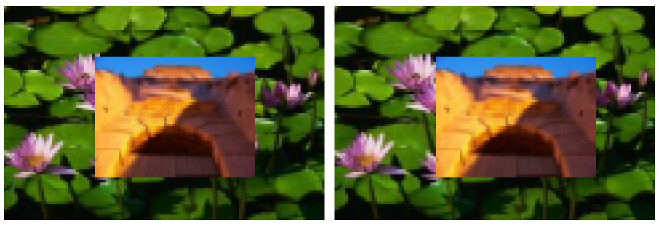


Figure 4.1: Testing data: static background with a rectangle moving in x direction.

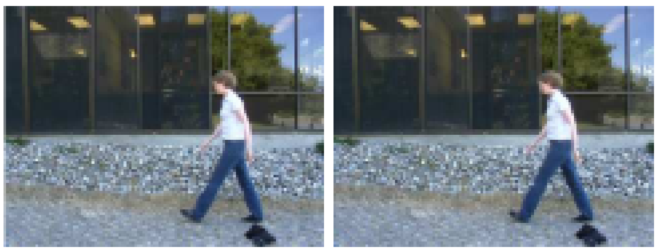


Figure 4.2: Real data for testing: A frame sequence taken from a still camera (courtesy Hedvig Sidenbladh [11]). The scene consists of a person walking against a static background.

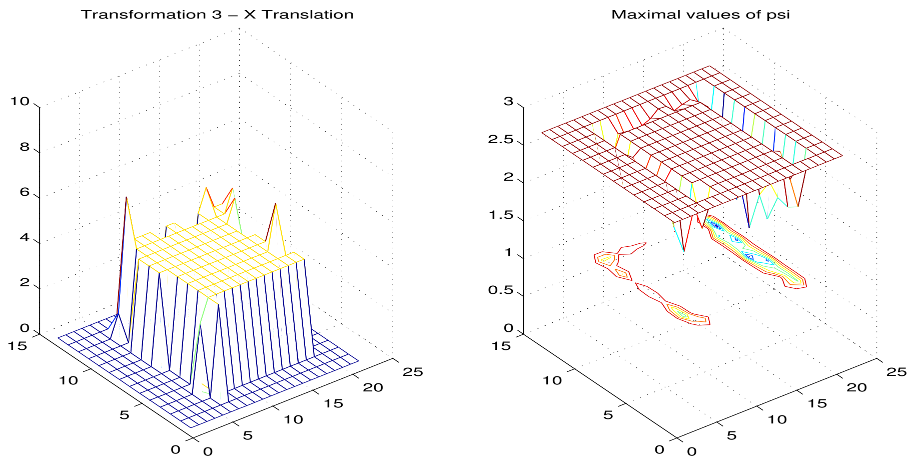


Figure 4.3: Initial estimation based solely on likelihood. The left side of this figure shows values of the recognized transformation (motion in x direction) in the image showed on figure 4.1 (moving rectangle). You can see good estimation of the transformation except of several patches at the edges of the rectangle. The right side of the figure shows value of the maximal likelihood.

4.2 Testing progress of belief calculation in every step of iteration without normalization of beliefs

As we wrote in section 4.5, we had problem with loosing information in every step of iteration. After consulting this problem with Mr. Pawan Kumar we got recommendation to use normalization of messages in every step. This information resolved our problem. On Figure 4.4 you can se previous results without normalization of the messages in example of frames with no motion. Every pair of pictures shows values of estimated transformation and beliefs. In the first step of iteration estimation is quite good, but it got worse and worse by every step.

4.3 Results with normalized messages in LBP

At last we used all information what we got from Mr. Pawan Kumar, it means proper sign and values of pixel brightness in calculation $\psi(s_k, s_l)$, likelihood without normalization and normalization messages in every iteration step. Final form of algorithm we tested on testing data and real data. Iteration stops after second step and result you can se on Figure 4.5, 4.6 and 4.7.

4.4 Components processing

Figure (4.9) shows processing the components. From colored patches we find connected components and merge the small ones to the surrounding.

Figure (4.10) show the found components of four frames and their combination according to the description above with similarity matrix $S(i, j)$.

In this experiments we used frames with resolution 320×200 pixels and resolution of every patch was 5×5 pixels. Results bring coarse estimation of the rigidly moving clouds of pixels located in the observed scene.

4.5 Using frames with large resolution

Using frames with large resolution new issues become. On the figure 4.11 is shown two frames with resolution 640×480 and result of the finding rigidly moving components. Background with texture is classified correctly, but there are a lot of separated components instead of exactly circumscribed body, hands and legs. Further there are a few components on the top of the

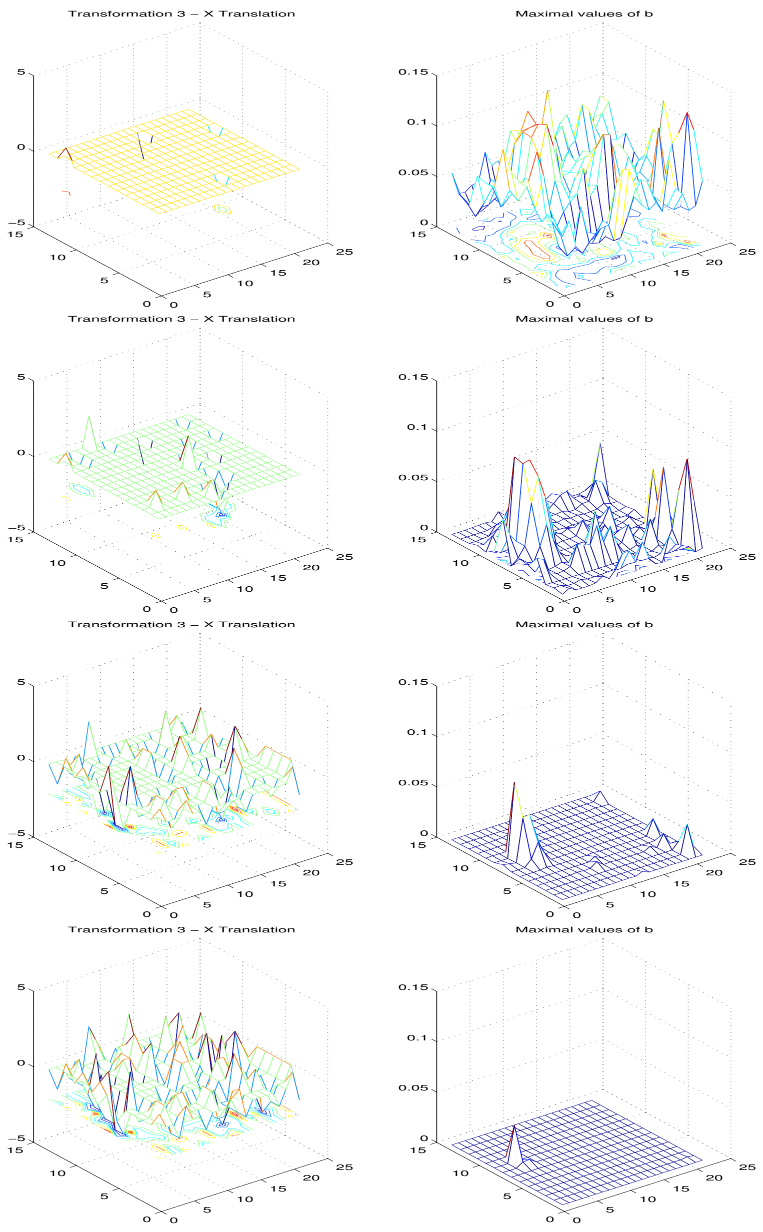


Figure 4.4: Four steps of iteration in no-motion example calculated by messages without normalization. On the first row of the pictures you can see right estimation based on likelihood (iteration 0) and values of beliefs. The next rows of the pictures show gradually loosing the proper estimation and falling of the values of the beliefs.

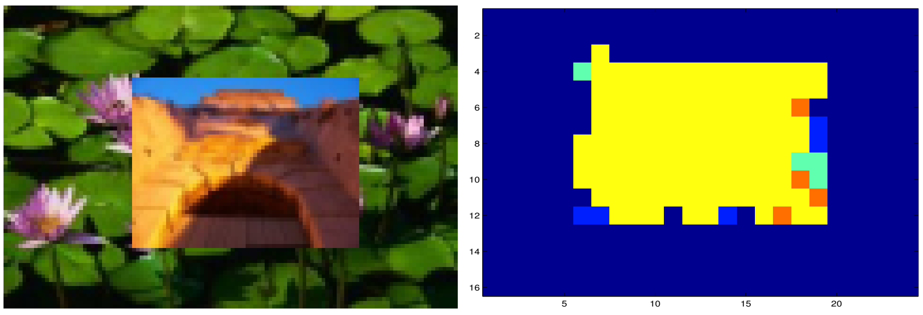


Figure 4.5: Initial estimation of testing data based on proper algorithm with normalized messages. You can see good estimation of the transformation except of several patches at the edges of the rectangle. These inaccuracies are resolved by subsequent part of the algorithm.

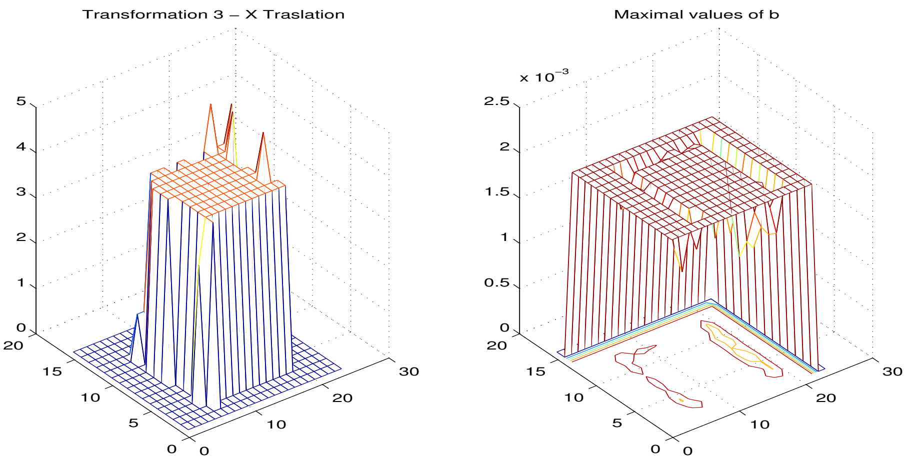


Figure 4.6: Values of estimated transformation of testing data and their beliefs. The peaks are caused by an inaccurate estimation at the edges, where patches are located on limits of the rigid by moving components.

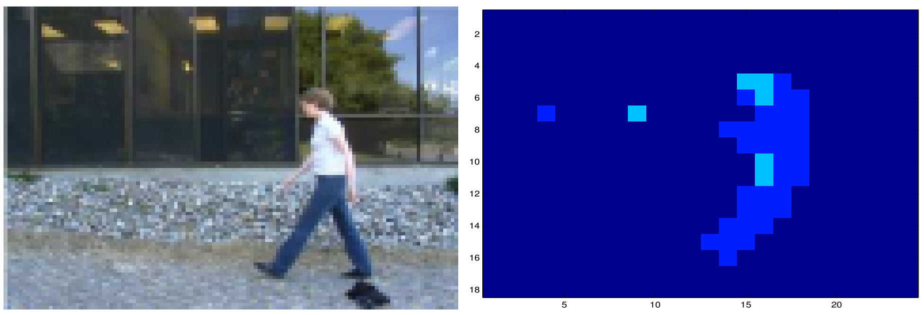


Figure 4.7: Initial estimation of real data based on proper algorithm with normalized messages. You can see the right recognized silhouette of the person - the head, torso and one hand. These parts are moving rigidly. Several inaccurate recognized patches are located mostly at the edges.

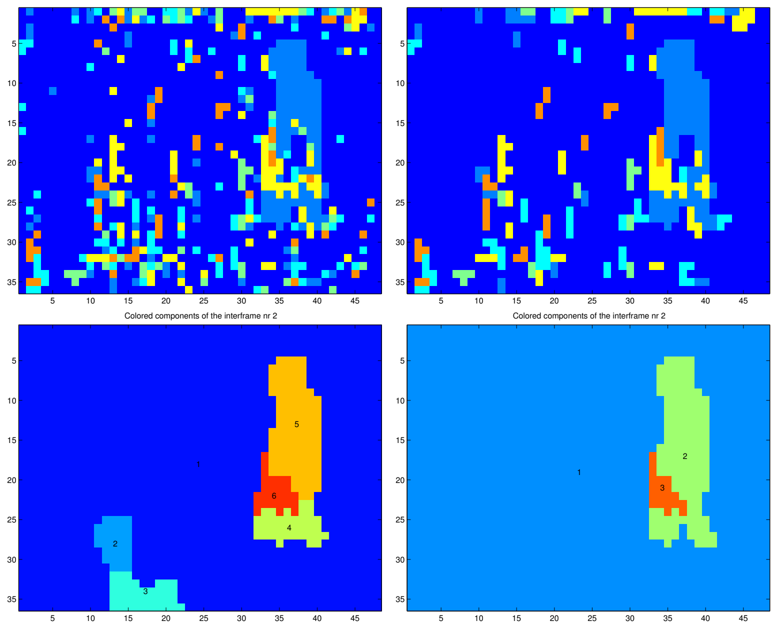


Figure 4.8: Results with (first column) and without (second column) the first stage of merging components. The first row shows colored patches and the second found components. Components 2 and 3 on the left bottom image are only part of the background, component 4 should be part of the component nr 5.

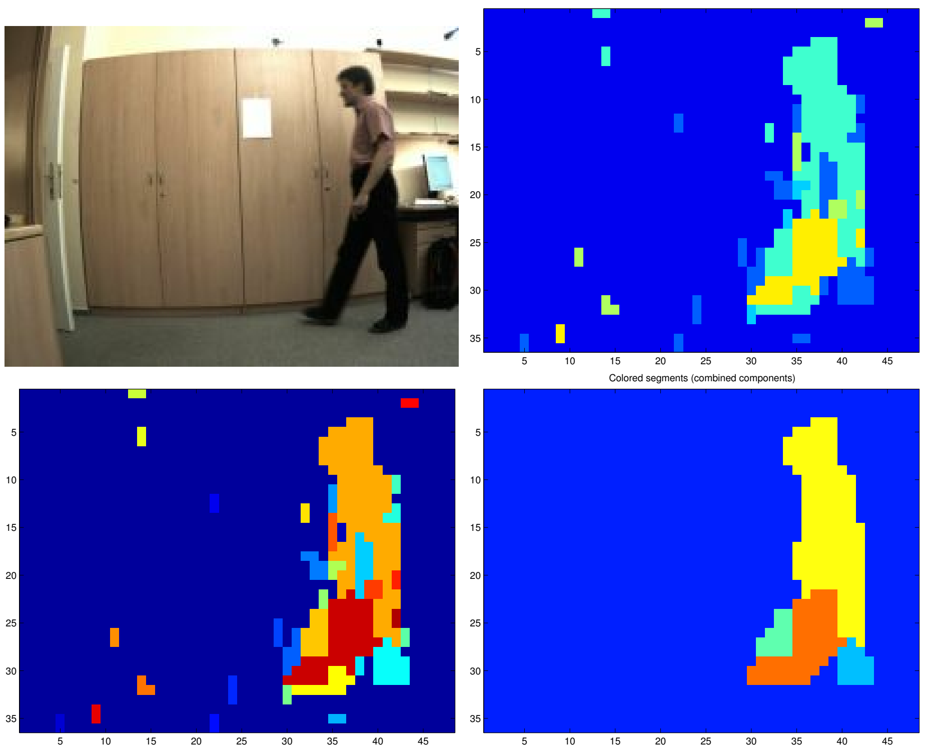


Figure 4.9: Components processing: From colored patches we find connected components and merge the small ones to the surrounding. Right bottom image shows final shape of the find connected components.

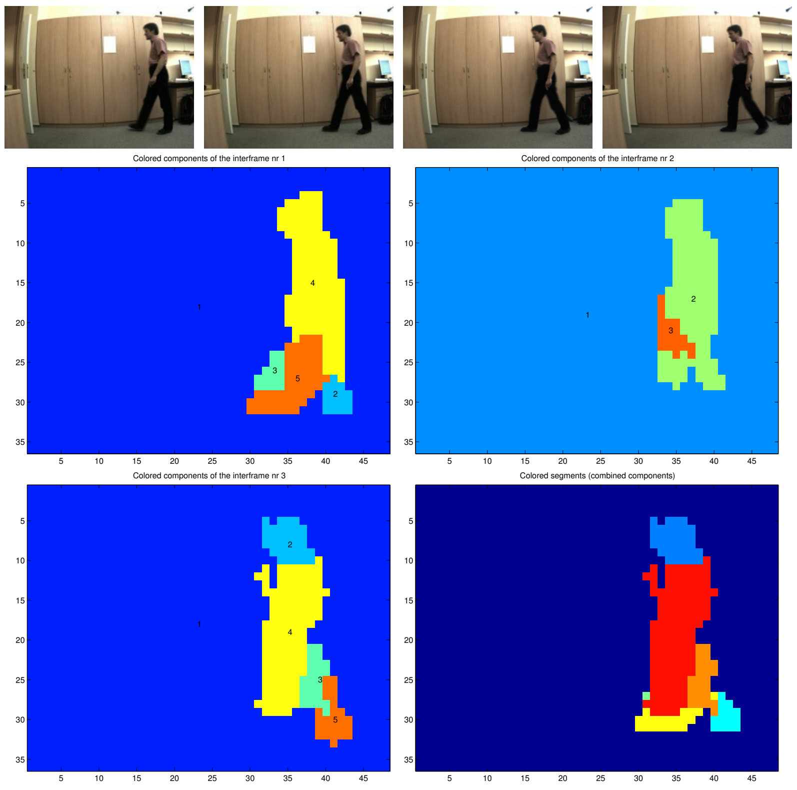


Figure 4.10: Components combining: The first row of pictures shows the video frames, next two frames show found components in each inter-frame and their combination.

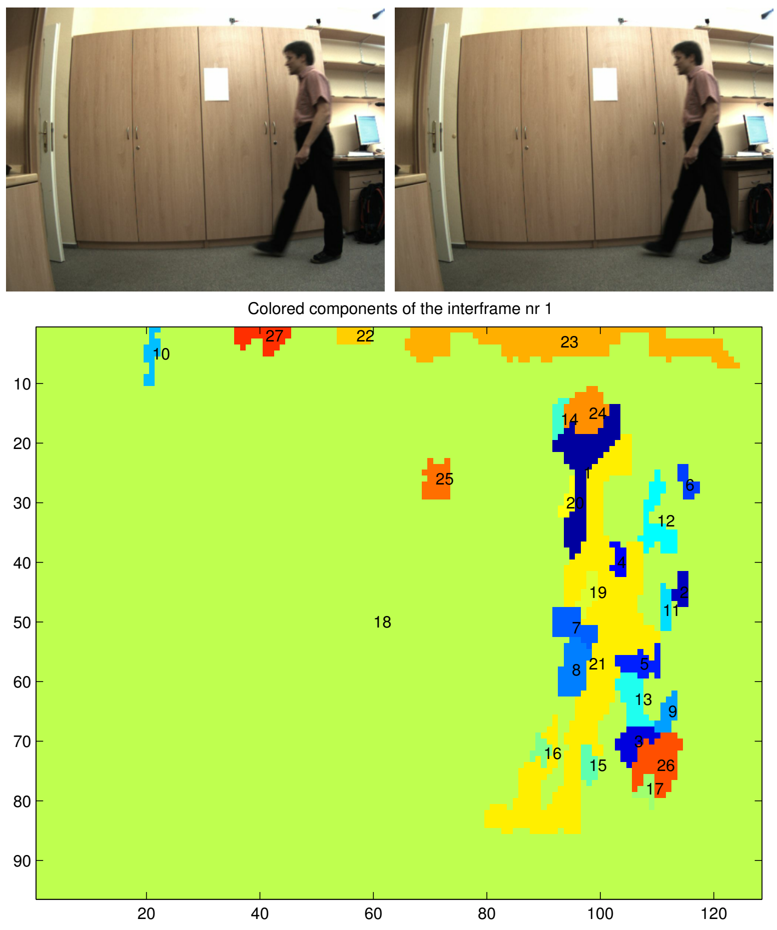


Figure 4.11: Emergence of the virtual components (10, 22, 23, 25 and 27) while using large resolution of the frames. This is caused absence of the texture on the picture.

image (components 10, 22, 23 and 27) and one separated component number 25. Emergence of these components isn't caused due to rigid motion, these components aren't moving. As can be seen on the original frames there is white space without texture on the place of the components. The texture absence causes wrong classification of the patches.

4.6 Dilution caused by one wrongly segmented frame

Grouping and combining components isnt robust to wrong finding components in one inter-frame. As can be seen on the figure 4.12 first 3 inter-frames have correctly found components, but components on the $4th$ inter-frames are too much diluted. This dilution of the one inter-frame leads to grouping these components with correctly found components and diluting final segments into small ones (see figure 4.12, the last image). The first 4 frames and their inter-frames are the same as in the experiment described in Components processing so the result should be similar.

4.7 Experiment with more frames and different articulated object in scene

For this experiment we used 10 frames from video with cow walking in scene [8]. The video is taken from a static analog camera which introduces a lot of noise. The video has resolution 320×200 pixels and resolution of every patch was 5×5 pixels. Every inter-frame brings quite good results of coarse estimation of the motion between frames (legs, body and background segmented). Combination of the all inter-frames shows segments of the legs but without body. It is caused, that the body is not separated from legs in no inter-frame, see Figure 4.13.

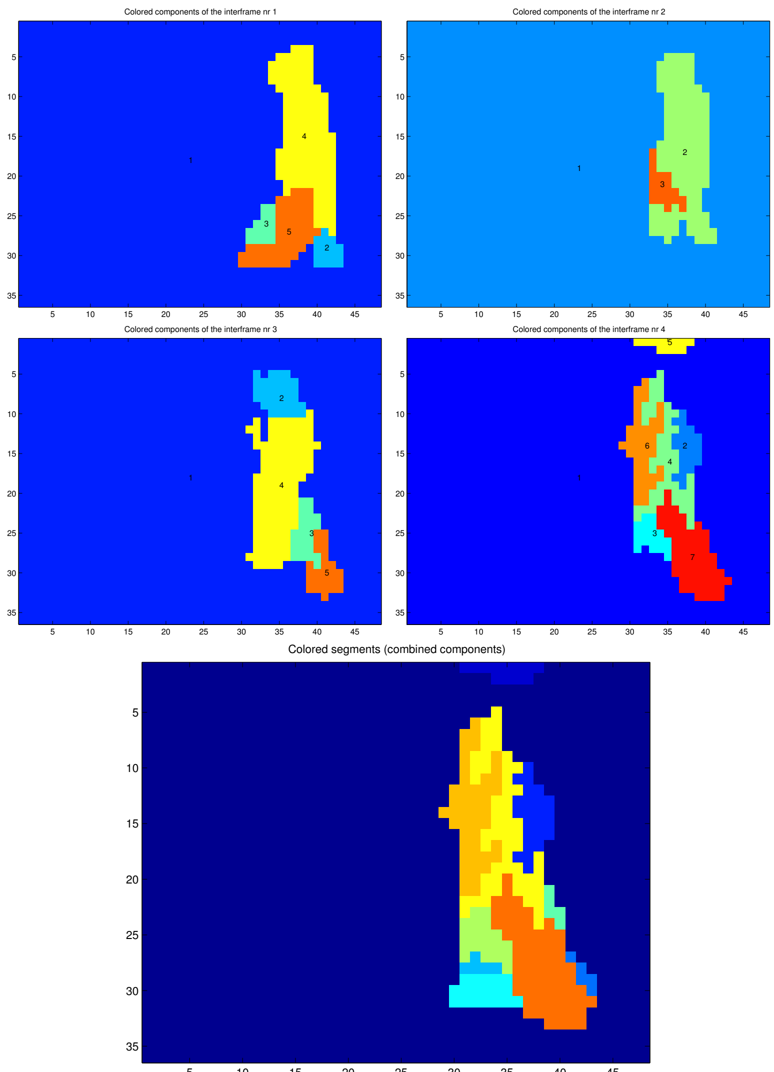


Figure 4.12: Dilution caused by one wrongly segmented frame. The first 4 frames and their inter-frames are the same as in the experiment 4.10 the last one caused the dilution.

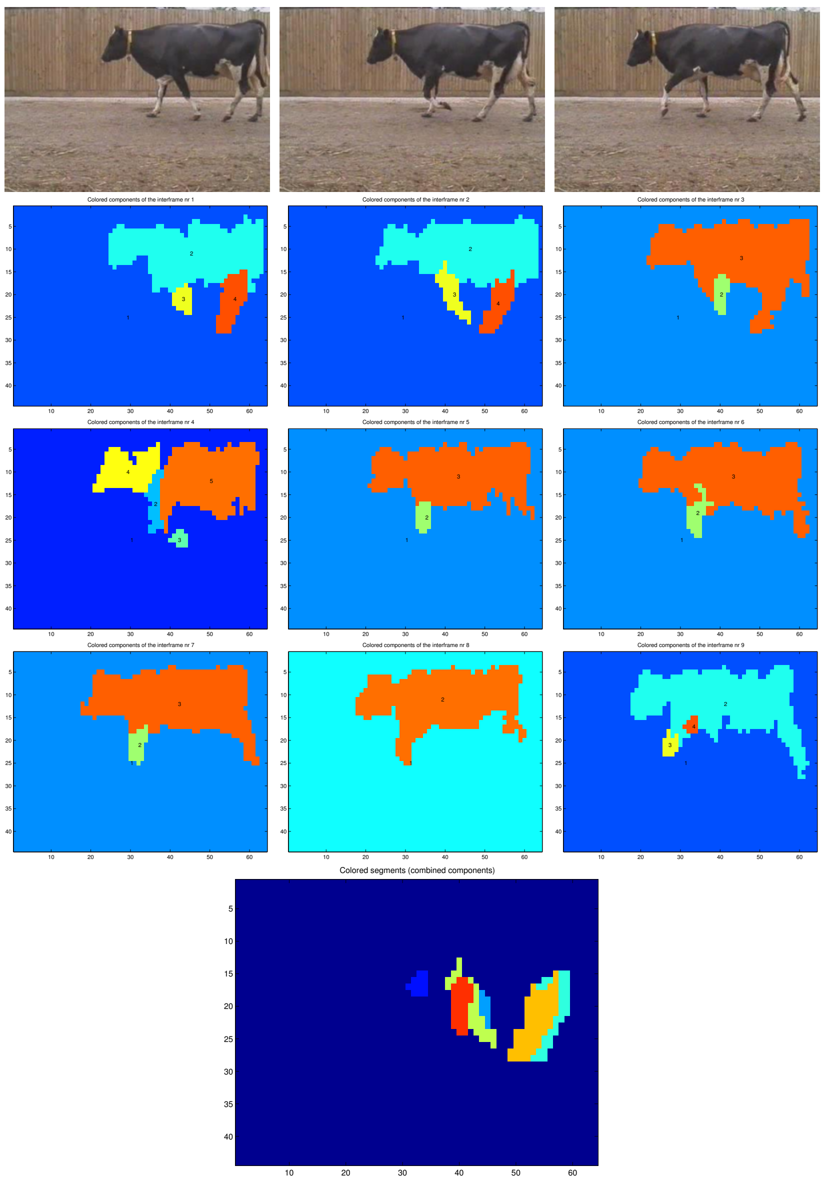


Figure 4.13: Experiment with more frames and different articulated object in scene: 10 frames from video with cow walking in scene [8]. The video is taken from a static analog camera which introduces a lot of noise. The first row of the pictures shows the first, middle and last frame of the video, next 9 images show components found in every inter-frame and last image is combination of all inter-frames. Although all inter-frames are quite good segmented, final combination is short of segment denoting body.

Chapter 5

Conclusion

Our implementation can recognize moving components in video and combine these components into segments over more frames, but we did not test it by large images and bigger set of putative transformation due to time and memory complexity. We improve components combination using similarity matrix with weighted values and use coarse to fine strategy to obtain shorter calculation times. We identify main issues of this algorithm: time complexity of the calculation of the likelihood and mainly time and memory complexity of the loopy belief propagation.

The most of the algorithm was successfully implemented, but not entire algorithm. Unfortunately we did not succeed in the implementation of last step of the algorithm, the refining shape. Despite that fact preliminary results can be used for recognition applications

Appendix A

Notes to article [5]

The article [5] is dense and contains a few unclearness, what we must have consulted with M. Pawan Kumar. We note these inaccuracies for better understanding in future. There is a typo in equation (10) in [5], there should be plus instead of minus. Sign minus leads to opposite dependence. Equation (12) in [5] has wrong index letter by first s, there should be s_l (see proper equation 2.5). The part of the algorithm Combining components is in [5] described very briefly and imperfectly for implementation, proper explanation is described above.

Better description with corrected mistakes can be found in enlarged version of the article. This enlargement leads to better understanding of the messages passing algorithm of the LBP, has more experiments, but still is short of description of combining components with similarity matrix.

Understanding of this method also needs more knowledge from computer vision, i.e. about loopy belief propagation (LBP), Markov random field (MRF) etc.

Appendix B

Hidden Markov Models

Hidden Markov Models are used in a wide variety of applications where a sequence of observable events is correlated with or caused by a sequence of unobservable underlaying states. In our example the observable events are likelihoods of very site (and every transformation) and the unobservable states are their beliefs. A major obstacle in scaling Hidden Markov Models up to larger spaces is the computational cost of implementing the basic primitives associated with them: given an n-state Hidden Markov Model and a sequence of T observations, determining the probability of the observations, or the state sequence of maximum probability, takes time $O(Tn^2)$ using the forward-backward and Viterbi algorithm. The quadratic dependence on the number of states is a bottleneck that necessitates a small state set, often artificially coarsed.

An Hidden Markov Models can be represented by a 5-tuple $\lambda = (S, V, A, B, \pi)$, where $S = (s_1, s_2, \cdots, s_n)$is a finite set of (hidden) states, $V = (v_1, v_2, \cdots, v_n)$ is a finite set of observable symbols, A is an $n \times n$ matrix with entries a_{ij} corresponding to the probability of going from state i to state j, $\{B = (b_i(k))\}$ where $b_i(k)$ specifies probability of observing symbol v_k in state s_i, and π is an n-vector with each entry π_i corresponding to the probability that the initial state of the system is s_i. The vector π is not important in our application. Let g_t denote the state of the system at time t, while o_t denotes the observed symbol at time t. Given a sequence of observations $O = (o_1, o_2, \cdots, o_T)$ there are three standard estimation (or inference) problems that have wide application:

- Find a state sequence $Q = (q_1, q_2, \cdots, q_T)$ maximizing $P(Q \mid O, \lambda)$.
- Compute $P(O \mid \lambda)$, the probability of an observation sequence being generated by λ
- Compute the posterior probabilities of each state, $P(q_t = s_i \mid O, \lambda)$.

Our application of the Hidden Markov Model is the first task of the itemization: find state sequence (labels) of the patches, which maximize the equation 2.1.

Bibliography

[1] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast aproximate energy minimization via graph cuts. *IEEE*, 2001.

[2] R.T. Collins and et al. A system for video surveillance and monitoring: Vsam. Technical Report CMU-RI-TR-00-12, Carnegie Mellon University., 2000.

[3] D. Cremers and S. Soatto. Variational space-time motion segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 886–892. IEEE Computer Society Press, 2003.

[4] I. Haritaoglu, D. Haewood, and L. S. Davis. A real time system for detecting and tracking people. In *Pattern Anal. Mach. Intell*, volume 2, pages 809–830, 2000.

[5] M. P. Kumar, P. H. S. Torr, and A. Zisserman. Learning layered motion segmentations of video. In *Proceedings of the International Conference on Computer Vision*, volume 1, pages 33–40, 2005.

[6] Pawan Kumar. Personal communication, 2006.

[7] Bastian Leibe, Edgar Seemann, and Bernt Schiele. Pedestrian detection in crowded scenes. *Multimodal Interactive Systems*, 2005.

[8] D.R. Magee and R.D. Boyle. Detecting lameness using re-sampling condensation and multi-stream cyclic hidden markov models. In *IVC*, volume 8, pages 581–594, June 2002.

[9] A. Meygret and M. Thonnat. Segmentation of optical flow and 3d data for the interpretation of mobile objects. In *International Conference on Computer Vision*, December 1990.

[10] Judea Pearl. *Probabilistic reasoning in intelligent systems : networks of plausible inference*. The Morgan Kaufmann series in representation and reasoning. Morgan Kaufmann, San Francisco, 1988.

[11] H Sidenbladh and M. J. Black. Learning the statistics of people in images and video. In *IJCV*, volume 1, pages 181–207, September 2003.

[12] C Stauffe and W. E. L. Grimson. Adaptive background models for real-time tracking. In *Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2242–2252, June 1999.

[13] A. Verri, S. Uras, and E. DeMicheli. Motion segmentation from optical flow. In *Fifth Alvey Vision Conference*, pages 209–214, 1989.

[14] Yair Weiss. Beliefs propagation and revision in networks with loops. Technical Report A.I. Memo No. 1616, Massachusets Institute of Technology, Artificial Intelligence Laboratory, MA, November 1997.

[15] J. Wills, S. Agarwal, and S. Belongie. *What went where.* IEEE Computer Society, 2003.

.