Technische Universität Ilmenau
Fakultät für Informatik und Automatisierung
Fachgebiet Neuroinformatik und Kognitive Robotik

Optimization-Based Motion Segmentation Using Additional Visual Cues

Diplomarbeit zur Erlangung des akademischen Grades Diplominformatiker

Boris Lau

Betreuer: Dipl.-Inf. Sven Hellbach
Verantwortlicher Hochschullehrer:
Prof. Dr. H.-M. Groß, FG Neuroinformatik und Kognitive Robotik

Die Diplomarbeit wurde am 02.02.2007 bei der Fakultät für Informatik und Automatisierung der Technischen Universität Ilmenau eingereicht.

Inventarisierungsnummer: 2006-11-02/121/IN00/2233

Abstract

Motion segmentation is a fundamental basis for automated image understanding and evaluation of image sequences by computer systems. This thesis presents an optimization-based approach to motion segmentation using Markov Random Fields. Local image motion is estimated with a correspondence matching scheme that incorporates complete sum of squared differences (SSD) surfaces over a search window, rather than best matches only. This way, information from ambigous motion estimates is integrated. The choice of SSD as similarity measure is motivated by experimental comparison of different metrics under noisy conditions.

Additional visual cues like color, depth information or brightness edges are integrated to complement the motion information. Fusion of the cues and segmentation is combined into one cost function. This way, individual cues can compensate the failure of others. The performance of the developed system is evaluated with experiments on rendered images with ground truth information and images obtained from a stereo camera system. Results for commonly used standard sequences are shown to allow comparison to other systems.

Zusammenfassung

Bewegungssegmentation ist eine fundamentale Grundlage für automatisiertes Bildverstehen und die Analyse von Bildsequenzen durch Computer Systeme. Die vorliegende Arbeit präsentiert einen optimierungs-basierten Ansatz zur Bewegungssegmentierung unter Benutzung von Markov Random Fields. Bewegung in Bildern wird mittels Korrespondenzbestimmung unter Verwendung der Quadratfehlersumme als Ähnlichkeitsmaß geschätzt. Anstelle der jeweils besten Matches werden komplette Fehlergebirge als Eingabedaten für die Segmentierung verwendet. Auf diese Weise werden Informationen mehrdeutiger Schätzungsergebnisse integriert. Die Wahl der Quadratfehlersumme als Ähnlichkeitsmaß ergibt sich aus empirischen Vergleichen verschiedener Metriken. Dabei wird die Erkennungsleistung für verrauschte Bilddaten getestet.

Zusätzliche visuelle Merkmale wie Farbe, Tiefeninformationen und Helligkeitskanten werden als Ergänzung zu den Bewegungsdaten bei der Segmentierung verwendet. Die Fusion und Segmentierung erfolgt dabei kombiniert durch Optimierung einer Kostenfunktion. Auf diese Weise können einzelne Merkmale den Ausfall anderer kompensieren.

Die Leistung des vorgestellten Systems wird anhand von gerenderten Bildern mit Ground-Truth Daten sowie Bildsequenzen von einem Stereokamerasystem evaluiert. Ergebnisse für allgemein benutzte Standardsequenzen werden präsentiert, um den Vergleich mit anderen Systemen zu ermöglichen.

Danksagung

Ich danke Dipl.-Inf. Sven Hellbach für die engagierte Betreuung meiner Arbeit, für viele gute Ratschläge, intensive Diskussionen und seine ständige Hilfsbereitschaft. Weiterhin danke ich Prof. Dr. Horst-Michael Groß für die Betreuung und die vielseitige Unterstützung während meiner ganzen Studienzeit.
Mein Dank gilt auch Lars Hörchens und Melanie Keller für das Korrekturlesen der Arbeit, sowie Paul Geisler für Anmerkungen und Tipps zu mathematischen Notationen. Lars Hörchens und Daniel Kondermann haben mich auf wertvolle Quellen hingewiesen, auch dafür meinen Dank.
Die langjährige ideelle und materielle Förderung der Studienstiftung des deutschen Volkes hat mir viele Perspektiven und Möglichkeiten verschiedenster Art eröffnet. Dafür bin ich zutiefst dankbar. Meinem Referenten bei der Studienstiftung, Dr. Gerhard Teufel, und meinem Vertrauensprofessor an der TU Ilmenau, Prof. Dr. Martin Dietzfelbinger, danke ich für die persönliche Begleitung meines Studiums.
Danke an meine Familie und alle Freunde, die während der letzten 26 Jahre für mich da waren. Einen ganz besonderen Dank möchte ich meinen Eltern aussprechen, die mich nicht nur bei allen Vorhaben unterstützt, sondern auch in den letzten Monaten wieder beherbergt und umsorgt haben.

Erklärung: Hiermit versichere ich, dass ich diese Diplomarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe. Alle von mir aus anderen Veröffentlichungen übernommenen Passagen sind als solche gekennzeichnet.

Hamburg, 02.02.2007

Boris Lau

Contents

List of Notations v
List of Figures vii

1 Introduction **1**
1.1 Segmentation and Motion Segmentation 2
1.2 Scope and Fundamental Ideas 4
1.3 Organization and Contribution of this Work 4

2 Background and Previous Work **5**
2.1 Motion Estimation 5
2.1.1 Sparse: Feature-Based 7
2.1.2 Dense: Optical Flow 8
2.2 Representation of Regions 12
2.2.1 Labeling 13
2.2.2 Explicit Contours 14
2.2.3 Level Sets 15
2.3 Segmentation Method 16
2.3.1 Clustering 17
2.3.2 Region Merging 18
2.4 Segmentation by Optimization 18
2.4.1 Markov Random Fields 19
2.4.2 Motion Discontinuities 23
2.4.3 Bayesian Formulations 24
2.4.4 Optimizing the Configuration 25

2.4.5 Optimizing the Prototype Parameters . . . 28
2.4.6 Adjusting the Number of Labels . . . 29
2.5 Multiple Cues . . . 29
2.5.1 Sequential Integration . . . 30
2.5.2 Combined Fusion and Segmentation . . . 31
2.6 Other Properties . . . 32
2.6.1 Motion Models . . . 32
2.6.2 Motion Segments vs. Motion Layers . . . 33
2.6.3 Dominant Motion vs. Clustering . . . 33
2.6.4 Dealing with Occlusions . . . 34
2.6.5 Limitations . . . 36
2.7 Conclusion . . . 36

3 Multi-Cue Motion Segmentation **39**
3.1 Framework . . . 39
3.2 Segmentation Based on Motion Information . . . 42
3.2.1 Detection of Occlusions . . . 43
3.2.2 Optimization of Labels . . . 43
3.2.3 Initialization . . . 43
3.2.4 Discussion . . . 44
3.3 Integration of Additional Visual Cues . . . 44
3.3.1 Categorization . . . 45
3.3.2 Color Information . . . 45
3.3.3 Depth Information . . . 46
3.3.4 Brightness Edges . . . 47
3.3.5 Discussion . . . 48
3.4 Algorithmic Description . . . 48

4 Analysis and Experiments **53**
4.1 Quantitative Evaluation of Segmentations . . . 54
4.1.1 Empirical Goodness Methods . . . 54

4.1.2 Empirical Discrepancy Methods 55
4.2 Experiments with Rendered Images 58
4.2.1 Regularity 59
4.2.2 Occlusions 59
4.2.3 Color Cue 59
4.3 Real Image Sequences 63

5 Discussion **65**
5.1 Towards Real Time Application 65
5.2 Moving Camera and Complex Motion Models 66
5.3 Neural Analogies 67
5.3.1 Motion Detection and Estimation 68
5.3.2 Labeling and Optimization 69
5.4 Conclusion 69

A Comparison of Similarity Measures **71**
A.1 Definition of Measures 71
A.2 Matching Performance 72
A.3 Computation Time 74
A.4 Conclusion 75

Bibliography **77**

List of Notations

Conventions

x, X	Scalar
$f(x), F(x, y)$	Function
$\vec{x}, \vec{X}$	Vector
$\mathbf{f}, \mathbf{F}$	Matrix
$\mathcal{F}$	Set

Frequently Used Symbols

$\mathbf{f}$	Labeling of a whole field
$f(x, y)$	Label of a site (x, y)
$I(x, y)$	Image brightness at pixel (x, y)
$\mathcal{L} = \{1, \ldots, L\}$	Set of all labels
M	Search window size
N	Correlation window size
$\mathcal{N}(x, y)$	Set of sites neighboring (x, y)
$\mathcal{P}_l$	Prototype parameters associated with label l
$\mathcal{S}$	Set of all sites in a labeling field
t	Time step
T	Temperature
(x, y)	Position of a pixel or site
$(\Delta x, \Delta y)$	Motion vector
X, Y	Width and height of an image

List of Figures

2.1 Sequential classification of existing approaches 6
2.2 Sparse and dense motion estimation from two images 7
2.3 Computation of an SSD surface 10
2.4 Occlusions and motion discontinuities 11
2.5 Methods for representing the position and shape of regions 13
2.6 Markov Random Fields: an overview 20
2.7 Neighborhoods and their cliques 21
2.8 Combined vs. sequential multi-cue fusion and segmentation 30

3.1 Schematic overview of the proposed system 40

4.1 Evaluation of segmentation quality based on the confusion matrix $\mathbf{C}_{i,j}$. 57
4.2 Two example images from rendered sequence "seq1" 58
4.3 Segmentation performance depending on the amount of regularization . 60
4.4 Segmentation performance depending on occlusion handling 61
4.5 Segmentation performance depending on integration of a color cue . . . 62
4.6 Segmentation of the "foreman" sequence 63
4.7 Segmentation of the table tennis sequence 63
4.8 Segmentation of camera images 64
4.9 Influence of the depth cue 64

A.1 Performance of similarity measures under noisy conditions 73
A.2 Computation time of similarity measures 75

Chapter 1

Introduction

With the spreading of digital still and video cameras in the last decades, there has been a tremendous increase in the amount of digital pictures and videos being used in industrial, commercial and private applications of all kinds. Multimedia products are present in everyday life. Along with this development, a strong demand for automatic image and video processing has grown. From storage and transmission to annotation and evaluation of image data, many tasks are done by machines in a semi-supervised or unsupervised fashion.

In this context, motion segmentation can be seen as a key to image understanding and many modern image processing applications. In video compression algorithms, the analysis of motion and regions with coherent motion helps to drastically reduce the amount of information that has to be stored and transmitted for each frame. While first generation video codecs only use displacement vectors for whole pixel blocks, the second generation approach comprises object segmentation to achieve better compression results [Torres et al., 1996]. In scene evaluation applications like video indexing [Snoek and Worring, 2005], medical imaging [Noble and Boukerroui, 2006] or camera surveillance [Hu et al., 2004], motion segmentation allows the detection and isolation of moving objects as a basis for higher-level scene understanding. Combined with object recognition and motion pattern analysis, it is part of the way to the sublime goal of building artificial systems that can semantically analyze and explain scenes, answering questions like “what happens?” or “who does what?”. Motion segmentation

Inv.-Nr.: 2006-11-02/121/IN00/2233

and motion understanding also plays an essential role in detecting and/or avoiding obstacles with vehicles or with a mobile robot, whenever one considers the position *and* the velocity of objects as a potential obstacle or threat rather than just the position, which seems desirable in highly dynamic environments.

Similar issues apply to the human vision system: walking in crowded areas or driving a vehicle in high traffic calls for a robust localization of moving obstacles and estimation of their velocities for collision avoidance. As with many problems in computer vision, the tasks of motion detection and segmentation are mastered by the biological vision systems of humans and animals with ease. In mammals the detection of movement for different purposes is part of the low-level stages of the vision system, namely the retina and the primary visual cortex [Squire et al., 2002]. Neurons in the primary visual cortex are tuned to respond exclusively to brightness patterns moving in a certain direction at a certain location in the field of vision [Hubel, 1995]. The system developed in this work employs a similar way of motion detection, in the way that a population of directionally selective units is used to obtain a dense representation of perceived motion. The integration of different sources of information is also an approach that biological vision systems [Grossberg, 2000] and the system presented here have in common: complementary visual cues are fused to gain better results and achieve robustness.

1.1 Segmentation and Motion Segmentation

Segmentation is the exhaustive partitioning of data into different non-overlapping parts called *segments* by a certain criterion. On a 2D image lattice, each segment consists of a region of adjacent pixels. The criteria that distinguish the segments in a good segmentation can be of low-level nature, for example gray-level, color, edges, or of higher-level nature like objects, foreground/background or semantic grouping. Existing approaches employ techniques like clustering, region growing, region merging, neural networks or optimization based methods using Markov random fields or graph cuts to compute the segmentation [Pal and Pal, 1993].

In motion segmentation[1] the partitioning criterion is based on motion information obtained from the comparison of two or more images of a sequence. Such segmentations can aim to distinguish image regions that move in the same way (e.g. translating into the same direction) while neglecting the real object boundaries, or they can aim for actual object segmentation. As mentioned before, motion segmentation can be seen as a step towards a semantic motion interpretation that can be applied on top of the actual segmentation process. In [Bouthemy and Francois, 1993] for example the direction of movement of objects relative to a vehicle is determined. In [Cutler and Davis, 2000] objects are classified based on the periodicity of their motion, and [Yang et al., 2002] proposed a system that recognizes hand gestures of American Sign Language.

An important aspect of different motion segmentation approaches is the type of motion that they are able to detect. Some methods try to split an image into parts of uniform translational or rotational motion, others allow even patches of homogeneous 3D planar or non-planar projective motion.

Motion segmentation and image motion in general are vast fields of research with many publications in the last decades. The first attempts to segment images based on motion information like difference images or optical flow have been made in the 1970s by applying traditional segmentation techniques to this kind of data [Onoe et al., 1973, Potter, 1977]. More recently the integration of feature tracking, trajectory analysis across multiple frames, and the use of multiple cues have been proposed. The approaches differ in characteristic ways, e.g. the number of objects they can detect, the types of estimated motion and how motion is estimated, how the segments are represented, or how problems like occlusions and transparency are handled. All of this shall be reviewed and discussed in Chapt. 2.

[1] In the remainder of this work, the terms segmentation, motion estimation and motion segmentation always refer to operations on 2D images or image sequences, if not stated otherwise.

1.2 Scope and Fundamental Ideas

To restrict the scope of this work, several decisions have been made a priori on the design of the motion segmentation approach developed in this work. The input data for the system are obtained from a single non-moving monocular or stereo camera, but the extension to a dynamic camera will be discussed. Several visual cues extracted from the image data will be integrated to achieve robustness in segmentation. Dense optical flow will be used as motion information. The representation of segmented regions will be realized in an implicit way, as labeling on a 2D lattice.
The author is also inclined to adhere to certain principles of design that are inspired by biological computation found in the human brain: as much available information as possible shall be integrated into the data fusion and segmentation process. This escpecially aims at the use of unsharp or uncertain information. When integrating multiple cues, the supplementary nature of the cues shall be utilized. This requires a framework where different types of information can be easily integrated, rather than discarded after sequential processing. Optimization on Markov Random Fields seems to be suitable to achieve these goals, and it also is an established approach to segmentation [Murray and Buxton, 1987]. Therefore, it has been picked as the method of choice for this work.

1.3 Organization and Contribution of this Work

In this work an approach to motion segmentation based on optimization on a Markov Random Field is proposed. This approach integrates motion estimation, the fusion of multiple visual cues and segmentation in one single optimization process. Chapt. 2 reviews and systemizes related research including past work and the current state of the art, and provides the background for the following sections. Chapt. 3 presents the new system for motion segmentation, based on motion and additional visual cues, and Chapt. 4 presents experiments to analyze the properties of the system and measure its performance. Chapt. 5 concludes with a discussion of the results and possible extensions. The appendix contains an experimental comparison of similarity measures.

Chapter 2

Background and Previous Work

This chapter reviews related previous work on motion segmentation and provides the background for the following chapters. Motion segmentation systems can be distinguished and classified according to several characteristic criteria. These properties are mostly orthogonal, and there is no intuitive hierarchical order that would allow a full graphical taxonomy. Nevertheless, to visualize the relation between the system developed here and previous work, and to organize this chapter, a sequential classification according to the main distinctive features has been done as depicted by the "decision tree" in Fig. 2.1. The text in the following five sections explains these characteristics and reviews the publications listed in the figure. Approaches that do not appear in the figure due to the sequential kind of classification are included in the textual review as well.

The remainder of the chapter describes further characteristics and ends with a conclusion and a summary of the current state of the art.

2.1 Motion Estimation

The estimation of motion in an image comprises the estimation of local displacement at some or all locations in an image from differences between two or more images from an image sequence. This is only possible if certain assumptions about the similarity of the images in the sequence are made. The two most popular ones are formulated in

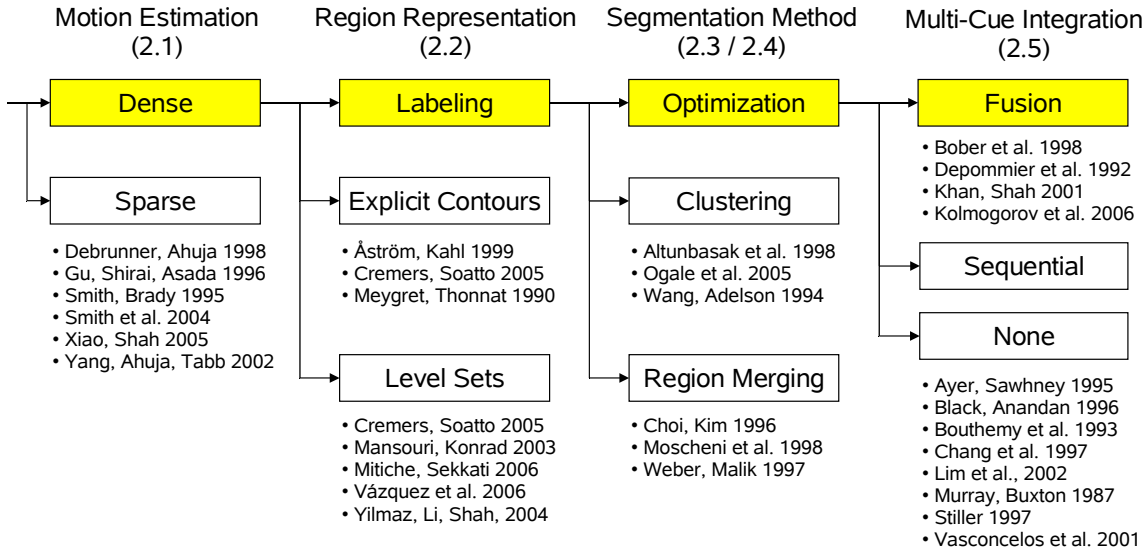
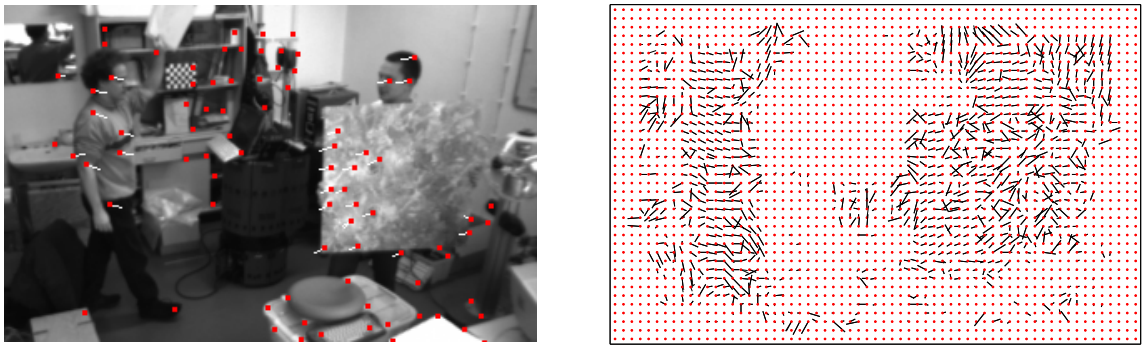


Figure 2.1: Sequential classification of existing approaches. The highlighted boxes represent the properties of the system presented in this work. The numbers of the related sections are given in parentheses.

[Horn and Schunck, 1981]: First, the "brightness constancy assumption" states that the brightness of a particular point in a moving pattern is constant, in other words, brightness changes in an image are due to motion. This assumption is violated by noise and illumination changes in general, causing the necessity of invariants and robustness in estimation. The assumption also fails for semi-transparent objects and in image regions that are newly occluded or uncovered by a moving object. The resulting problems will be discussed in Sect. 2.6.4. Second, it is assumed that motion varies smoothly across the image. Without an assumption like this, two or higher dimensional motion vectors cannot be computed, since images provide only one independent measurement per pixel [Horn and Schunck, 1981].

Two main kinds of motion estimation can be identified: *feature-based* approaches that provide a sparse estimate, and *optical flow* approaches that yield a dense estimate (see Fig. 2.2). The choice of a method is of course crucial for a motion segmentation system. It also determines, how motion information from more than two images of a sequence can be included, if desired.



(a) Sparse: Feature-based

(b) Dense: Optical Flow

Figure 2.2: Sparse and dense motion estimation from two images. The red dots mark the position of motion estimates, the lines attached to them resemble the motion vectors. The motion estimates in (a) were obtained using the KLT feature selection and tracking algorithm. The optical flow display in (b) shows the best SSD match for every fifth pixel.

2.1.1 Sparse: Feature-Based

If local motion is only estimated for some locations in an image (*sparse estimate*) that show special characteristics, the estimation method is called "feature-based". An example is shown in Fig. 2.2a. Features are image patches or transformed image patches extracted at image locations selected according to certain criteria. These criteria can demand for example the occurrence of corners or more complex properties, e.g. "Good features to track" [Shi and Tomasi, 1994]. Sophisticated feature selection aims for special invariants, e.g. to rotation and scaling [Lowe, 2004]. The features taken from an image frame are searched for in the next or previous frame of an image sequence. Matching can be done in a similar way as for correspondence-based optical flow, which is described further down. Also, more sophisticated *feature tracking* approaches can be applied. With the selection of features, the search is reduced to small parts of the image that seem especially promising for a correct and unambiguous matching result. In [Xiao and Shah, 2005], Harris corners are used as features in combination with the KLT tracking algorithm proposed in [Shi and Tomasi, 1994]. This tracking system has also been used to obtain the motion estimate in Fig. 2.2a.

Feature-based motion is suitable for the tracking of moving parts across multiple image

frames, thus computing a motion trajectory estimate. In [Debrunner and Ahuja, 1998] a motion estimate is obtained by linking feature occurrences to probable paths.
If regions of coherent motion are the desired segmentation result although estimation of motion is done sparsely, the segmentation process has to provide means for bridging the gap between the estimates. In [Smith and Brady, 1995] regions are created by computing the convex hull of tracked feature points with similar motion, and converted into a "radial map" model that describes regions with the position of a centroid, and radial distance measures to the boundary.
Smith et al. track color edges to estimate motion [Smith et al., 2004]. Boundaries are controlled by sample points and interpolation between them. To update the sample points, a search for the edge in a new frame is done for each sample point on a line perpendicular to the edge. Thus, motion is estimated sparsely along the edges, and regions without motion are assigned to the segments according to the boundaries.
In [Gu et al., 1996], spatially stable edge segments (SSESs) are used as features. They consist of connected "stable points" with similar movement, located at positions with high contrast and scale invariancy. Here the segmentation does not try to enclose regions with moving edges to get a segmentation, the connected edges themselves are taken as result.
A hybrid approach is presented in [Yang et al., 2002], where multi-frame sparse region matching is combined with dense 2-frame pixel matching, in order to recognize hand gestures of sign language with time-delay neural networks.

2.1.2 Dense: Optical Flow

If local motion is attempted to be estimated for each pixel (*dense estimate*), the estimate is often called "optical flow"[1]. Optical flow is represented by a 2D vector field,

[1]There are different definitions for the term "optical flow" in the literature: in [Vega-Riveros and Jabbour, 1989] it only refers to gradient-based methods with the argument that only these are based on the brightness constancy assumption. This view is not commonly adopted, [Black and Anandan, 1996] for example describe correspondence-based approaches as "the most direct way to use the brightness constancy assumption". Here the term is used for pixel-based approaches in general as commonly done [Yang et al., 2002].

with displacement vectors for each pixel, as shown in Fig. 2.2b. There are different ways to compute the optical flow, the ones most widely used are either gradient-based or correspondence-based. While the use of gradients provides more accurate estimates, it is only applicable for very small displacements around 1 pixel per frame, unless combined with a multi-resolution strategy as in [Black and Anandan, 1996]. For more details and comparisons see the reviews in [Barron et al., 1994] and [Galvin et al., 1998].

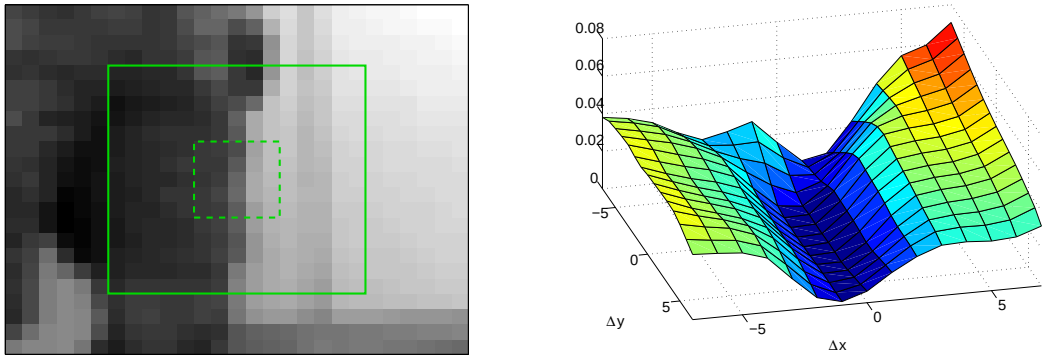
Correspondence-based approaches try to find pixels or patches in an image that correspond to the same spots in another image. To estimate optical flow with correspondences for one pixel at position (x, y) in the image $I(x, y)$ at time step t, a rectangular image patch called *correlation window* of size $N \times N$ (odd N) with (x, y) being the center, is extracted from the image. This patch is compared with image patches of the same size extracted from image $I'(x, y)$ of the next or previous time step at the positions $(x + \Delta x, y + \Delta y),\ \forall \Delta x, \Delta y \in \left[-\frac{M-1}{2}, \frac{M-1}{2}\right]$, thus searching in an area called *search window* of size $M \times M$, with $M > N$. Without loss of generality, $I'(x, y)$ is here assumed to belong to time step $t + 1$, and the brightness values of I and I' are assumed to be scaled between 0 and 1. The comparison is done using a scalar metric that expresses the similarity or difference of the brightness values in the image patches. Popular choices are the sum of the absolute differences (SAD), sum of squared differences (SSD) or a cross-correlation (CORR). With $k = \frac{N-1}{2}$

$$SAD(x, y, \Delta x, \Delta y) := \frac{1}{N^2} \sum_{i=x-k}^{x+k} \sum_{j=y-k}^{y+k} \Big| I(i, j) - I'(i + \Delta x, j + \Delta y) \Big| \tag{2.1}$$

$$SSD(x, y, \Delta x, \Delta y) := \frac{1}{N^2} \sum_{i=x-k}^{x+k} \sum_{j=y-k}^{y+k} \Big(I(i, j) - I'(i + \Delta x, j + \Delta y) \Big)^2 \tag{2.2}$$

$$CORR(x, y, \Delta x, \Delta y) := \frac{1}{N^2} \sum_{i=x-k}^{x+k} \sum_{j=y-k}^{y+k} \Big(I(i, j) \cdot I'(i + \Delta x, j + \Delta y) \Big) \ . \tag{2.3}$$

This matching procedure has to be done for all pixels of the image, yielding a computational complexity of $O(X \cdot Y \cdot N^2 \cdot M^2)$, for an image of width X and height Y. For practical implementation, uniformity, locality and the possibility of parallelization are

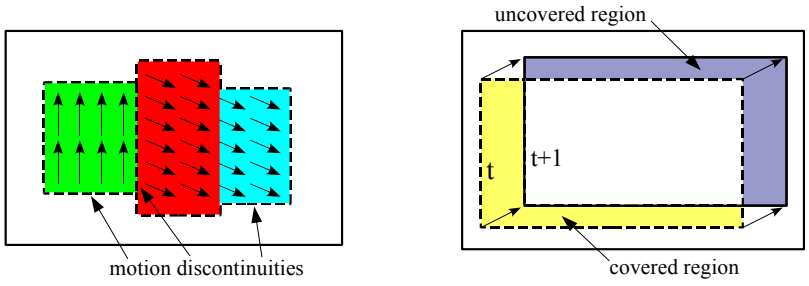


(a) Correlation and search window (b) SSD surface

Figure 2.3: Computation of an SSD surface. In (a) the size of the correlation window is marked with a dashed line. The patch of that size is moved within the boundary of the search window (solid line), and matched using a similarity measure to calculate the supports for the SSD surface (b). The resulting SSD surface has an ambiguous minimum, therefore the motion estimate is ambiguous.

desired properties for these algorithms [Anandan, 1989]. Since the same local matching operation is executed for all pixels, the computation of optical flow using correlation has these properties.

Converting the similarity values for the different displacements into one resulting motion vector is the critical part of the motion estimation process. As a first approximation, the best matching patch according to the used metric is assumed to be the correct corresponding image patch, and its displacement is used as local motion estimate. Whenever there is not enough contrast in the image such that the similarity of the correct match does not deviate significantly from the others, the selection of the correct displacement vector matches becomes problematic. All matches could exhibit the same similarity, or the correct match could be less similar than others, due to disturbances or image noise. Even if there is a strong contrast in an image patch, the matching can be ambiguous due to the *aperture problem*: if the brightness in the image patch varies only in one direction, the matching is ambiguous. An example for such a case with SSD matching is given in Fig. 2.3a. Instead of one distinct minimum that can be used as *the* motion estimate, a number of displacements have the same or a similar low SSD value and could be the true displacement.



(a) (b)

Figure 2.4: Occlusions and motion discontinuities caused by object motion. Motion discontinuities (a), marked with a dashed line, are boundaries between areas of different coherent motions. If such a boundary falls into the correlation window of optical flow computation, motion cannot be estimated correctly. Occlusions (b) are regions covered or uncovered by a moving object, like the rectangle in the figure. For these areas the brightness constancy assumption is violated, and correspondences for optical flow cannot be established.

An effective way to tackle this issue is to introduce confidence measures that are used as weights for additional smoothing of the flow field [Stephan, 2001]. This way, uncertain motion information is discarded. Other approaches use robust statistics [Ong and Spann, 1999] or optimization (e.g. [Konrad and Dubois, 1990], [Sim and Park, 1998]) to achieve good results. [Lai and Vemuri, 1998] proposed to use SSD surfaces with values for all displacements rather than just the best matches for motion estimation in a minimization procedure. This way, the quality of the match can be weighed up against smoothness constraints, and information from ambiguous matching surfaces is used in a sensible way. The system presented in this work transfers this approach to motion segmentation.

Using a correlation window with $N > 1$ brings in the local smoothness assumption introduced above. However, finding the right size correlation window resembles a problematic trade-off: in too large a correlation window the motion can vary within the image patch, causing disturbed results. Too small a size will make the matching ambiguous, rendering correct motion estimation impossible. Of course, the optimal size depends on the actual size of coherent motion patches in the image data. This problem has been discussed in [Szeliski and Shum, 1996].

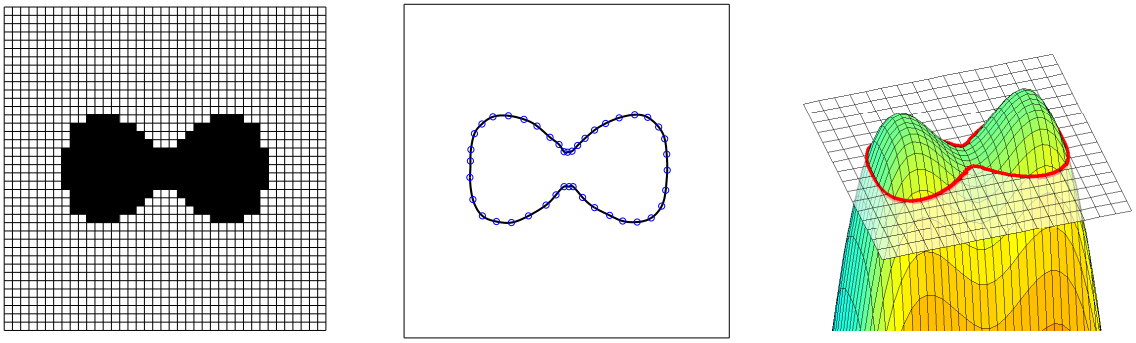
Another problem related to the use of a correlation window occurs at motion boundaries, formed by foreground/background boundaries or where overlapping objects move in different directions, as shown in Fig. 2.4a. Here the local smoothness assumptions is violated and the extracted image patch contains multiple non-coherent motions, yielding bad estimation results. Simple smoothing of the flow field is unsatisfactorial in this case, since the mean of very different motion vectors is often an inappropriate motion vector. Coarse-to-fine strategies in the matching can help to reduce the spatial extent of this problem by using adaptive correlation window sizes or matching on different image resolutions [Anandan, 1989]. Other approaches to this problem in the context of optimization-based motion segmentation use explicit modeling of discontinuities in an otherwise smooth segmentation (see Sect. 2.4.2). Such a discontinuitiy model can be tied to additional cues as well, as discussed in Sect. 2.5.2.

Whenever an object moves, parts of other objects or the background behind the object become occluded, and others are uncovered, as depicted in Fig. 2.4b. In these regions the brightness constancy assumption is violated as well, since there is no correspondence for these pixels in the following or preceding images and the change between two images cannot be explained by motion alone. Accordingly, the similarity measures of the matching process in this regions are defective and can yield bad motion estimation results. Solutions to this problem in the context of motion segmentation are discussed in Sect. 2.6.4.

Considering approaches to motion segmentation with a dense motion estimation scheme, multi-frame motion and segmentation information can be integrated implicitly by propagating estimation results to following frames for initialization of the computation in the next time step. See for example [Black, 1992] and [Bouthemy and Francois, 1993] and the approach developed in this work.

2.2 Representation of Regions

Different segmentation techniques can use different ways to encode the location and shape of the individual regions of a segmentation. Implicit representation with labeling, explicit boundary description like active contours, and which combine properties of



(a) Labeling (b) Explicit Contours (c) Level Sets

Figure 2.5: Methods for representing the position and shape of regions. Labeling (a) is a flexible representation. Segments are formed by adjacent sites carrying the same label (color). Explicit contour representations like snakes (b) have a fixed topology, but allow the use of high-level smoothness constraints and enforce closeness of regions. Level Sets (c) represent boundaries by the set of all locations with zero-value of a 2D surface. They combine topologic flexibility with a more explicit boundary representation.

both have been used for segmentation. Figure 2.5 shows an example of a representation of one region with all three methods. These methods are explained in further detail in the following.

2.2.1 Labeling

Implicit descriptions define a labeling $f(x, y)$ on all the sites $(x, y) \in \mathcal{S}$ of a 2D lattice of size $X \times Y$, which often is of the same size as the input images[2]. $\mathcal{S}$ is the set of all coordinate pairs in the lattice

$$\mathcal{S} = \{1, \ldots, X\} \times \{1, \ldots, Y\} \ . \tag{2.4}$$

The labeling assigns each site a label l from a label set $\mathcal{L} = \{0, ..., L\}$ by

$$f(x, y) = l \ . \tag{2.5}$$

[2]The notation used here is based on [Li, 2001] and adapted to incorporate coordinate pairs for site indexing.

A region $\mathcal{R}$ with its position and shape is implicitly defined by the set of connected sites that carry the same label:

$$\mathcal{R}_i := \left\{(x, y) \in \mathcal{S} \mid f(x, y) = i\right\} . \tag{2.6}$$

Note that this definition does not require the connectivity of all sites in a region. This constraint can be imposed during evaluation of a segmentation, interpreting one region as multiple unconnected segments. Labelings can be visualized easily as an image by using a different color for each label as done in Fig. 2.5a. For motion segmentation, each label l is associated with a parameter set $\mathcal{P}_l$. This set contains *prototype parameters*, e.g. a motion vector. These parameters act as motion models during the segmentation process and resemble an approximation of the input data, e.g. optical flow, of all sites (x, y) that carry the label l.
Implicit representations are very flexible: changes in the topology of regions such as splitting and merging can happen in the same "implicit" way as changing their shape. If the sites at the thin part of the black region in the center of Fig. 2.5a change their label to white, the region can be split up into two. Representing regions with labelings on a 2D lattice is widely used in segmentation. Examples of related previous work are presented throughout the remainder of this chapter.

2.2.2 Explicit Contours

Another approach is to use explicit boundary representations to describe the individual regions, e.g. snakes or other active contour models (see e.g. [Kass et al., 1988] or [Blake and Isard, 1998]). All pixel that lie inside a closed boundary belong to the same region, as shown in Fig. 2.5b. Here each boundary is defined by a spline curve with its control points. The number of snakes used in the representation limits the number of regions that can be described. This approach is limited in that the topology of the regions is mostly fixed during the segmentation process: splitting and merging cannot be achieved by adapting the control points, explicit routines would be necessary. On the other hand, active contours come with certain constraints that can be of advantage in segmentation applications: closed active contours naturally ensure connected regions, and allow elegant higher-level ways of controlling the boundary.

In [Cremers and Soatto, 2005] a closed spline curve with control points as shown in Fig. 2.5b is used for motion segmentation. The control points are evolved with a gradient descent scheme, minimizing a motion energy based on optical flow, and the length of the contour as a smoothness constraint. In [Dubuisson and Jain, 1995], snakes are applied for contour refinement and extraction as a post-processing step after the actual segmentation, which outputs a "motion mask" defined by a labeling. The radial map of [Smith and Brady, 1995] and the boundary model of [Smith et al., 2004] are also approaches with explicit contour representations. A short description has been given in Sect. 2.1.1.

In [Åström and Kahl, 1999] snakes are used to track brightness contours identified by an edge detector. From the motion parameters of the individual regions, the self-motion of the camera is estimated.

A different approach to explicit boundaries is to model separate boundary sections instead of closed curves. As mentioned before, [Gu et al., 1996] track edge segments with a feature-based approach. [Meygret and Thonnat, 1990] build similar "contour chains" based on optical flow, searching for continuous motion gradients along the contours. Combined with stereo vision, the authors obtain 3D segmentations.

In [Black, 1992] labeling is used in a way different from the method described in 2.2.1: the output of the segmentation is a binary edge image, the not necessarily closed boundaries of regions consist of connected pixels with the label that stands for "edge present". Thus the representation of boundaries is explicit, although labeling is used.

2.2.3 Level Sets

Level set methods combine the advantages of active contours with the flexibility of implicit boundary representations. A 2D surface $\phi(x, y)$ is defined over all pixels (x, y), and is adapted with an optimization scheme. The implicit contour $\mathcal{C}$ is defined as the set of points (x, y) where the surface crosses the zero-level:

$$\mathcal{C} = \left\{(x, y) \in \mathcal{S} \mid \phi(x, y) = 0\right\} . \tag{2.7}$$

Figure 2.5c depicts such a surface, the zero-level and the resulting boundary. This boundary splits the area in two regions, called "phases", where the parts with $\phi(x, y) > 0$

are considered to be on the "inside" and parts with $\phi(x, y) < 0$ considered to be on the "outside" of the contour. In contrast to explicit contour representations, a contour defined by a level set can consist of several boundaries: if the saddle point in the center of the surface in the figure is lowered below zero, the contour splits up into two closed boundaries. As with labeling-based representations, a region or phase defined by a level set can consist of multiple unconnected parts, and the topology of regions can change implicitly.

In [Cremers and Soatto, 2005], level sets are applied to motion segmentation. While one contour can define only two possibly splitted regions of coherent motion with its two phases, the authors use a bit-pattern-like combination of multiple boundaries to encode regions, which allows to describe up to n regions with $log_2 n$ surfaces. The surfaces are adapted by minimizing an energy function with the target of homogeneous motion in the phases, and a minimum length for the contours to enforce smoothness. The authors compare the strengths of this implementation to the alternative with active contours. [Vázquez et al., 2006] also use the boundary length as a measure for smoothness of the boundary in their energy function. A second term takes the conformity of estimated motion with image motion into account, and a third one the coincidence of the boundaries with motion discontinuities. In [Mansouri and Konrad, 2003], an energy function is defined that searches for the most probable transformation consisting of motion parameters and motion segmentation, given a pair of images. Brightness edges are included as an additional cue.

[Mitiche and Sekkati, 2006] proposed an approach for simultaneous estimation and segmentation of motion and depth in 3D, with the goal of recovering the 3D structure of a scene. The approach of [Yilmaz et al., 2004] to exploit shape changes of a level set boundary to detect occlusions is discussed in Sect. 2.6.4.

2.3 Segmentation Method

So far, methods for estimating motion from images and different ways of representing a segmentation have been discussed. This section provides the missing link with approaches to how an actual segmentation can be obtained. The review is limited to

systems that use an implicit representation with labeling as described in Sect. 2.2.1, like the system developed in this work does. System with region representations based on active contours or level sets are not covered in further detail, since they differ significantly in their approach.

Many algorithms from classical image segmentation based on brightness, color or other properties can be applied to motion segmentation. This section describes the main approaches. Optimization-based approaches have become quite popular, since they provide effective means to deal with noisy data[3]. These systems are presented in Sect. 2.4.

2.3.1 Clustering

According to Haralick and Shapiro, segmentation is spatial grouping, for example in an image. Clustering on the other hand is grouping in a value domain, e.g. brightness or motion vectors [Pal and Pal, 1993]. Clustering can nevertheless be used for motion segmentation. [Altunbasak et al., 1998] use K-means clustering [MacQueen, 1967] to determine a small number of classes (labels), each associated with a set of affine motion parameters from an optical flow field. Assigning each pixel the label with the best fitting motion parameters yields an initial motion segmentation.

In [Ogale et al., 2005], a 2D translational flow field is obtained from phase correlation of two images. The same phase correlation is used to determine one additional scale and one rotational parameter field from log-polar transformations of the same images. From this 4D motion field, the motion parameters for the background (depending on the camera-motion) are estimated. Now, each pixel is assigned either the background label or an object label, based on the discrepancy of the motion parameters and the optical flow estimated for that pixel. [Wang and Adelson, 1994] also use a label assignment procedure, where each pixel is assigned the best fitting label of a label set.

[3]The noisiness of motion estimates and types of disturbances have been discussed in Sect. 2.1.2.

2.3.2 Region Merging

Region growing is a classical segmentation technique, where pixel adjacent to a region are assigned to that region by changing its label, if the local data of that pixel, e.g. motion vectors, are similar to the prototype data of the region. A *region merging* algorithm processes whole regions in the way that two neighboring regions are assigned the label and thus merged, if their prototype data is similar or equal. If this merging process starts from single pixels, it can be similar to region growing with the additional merging of larger regions.

[Choi and Kim, 1996] apply this method to motion segmentation in a multi-stage algorithm: at first, neighboring pixels with similar 2D translational flow vectors are grouped together to regions. In the second stage, the similarity of neighboring regions is compared using an affine motion model. The third stage groups regions that are similar according to a quadratic motion model. This way, the model complexity of the motion estimate increases with each step during the segmentation process.

In [Moscheni et al., 1998], region merging is performed on a directed graph. Its nodes represent the individual regions, the weighted edges represent the similarity of adjacent regions according to a uni-directional measure which includes both temporal and spatial properties.

[Weber and Malik, 1997] base region merging decisions on a cost function, which tries to limit the number of labels while keeping the approximation error low. Although this is actually an optimization process, it is different from the kind of optimization approaches that is presented in Sect. 2.4, since it operates on regions rather than on single pixels.

2.4 Segmentation by Optimization

This section covers optimization-based segmentation techniques that are based on a labeling representation (see Sect. 2.2.1) and operate on individual pixels rather

than on regions as a whole (cf. Sect. 2.3.2)[4]. The typical representatives for this group of systems use statistical methods, especially Markov Random Fields (MRFs) or Bayesian frameworks like Maximum Likelihood Estimation (MLE), Maximum A-Posteriori (MAP) estimation or Expectation Maximization (EM). Despite the theoretic differences, algorithms based on these frameworks are often very similar in the context of motion segmentation. [Geman and Geman, 1984] combined MRFs and Bayesian MAP estimation into MAP-MRF labeling (see [Li, 2001]), and several motion segmentation systems are based thereon.

2.4.1 Markov Random Fields

In this work, Markov Random Fields (MRFs) are always defined on regular 2D lattices of sites $(x, y) \in \mathcal{S}$. There are other varieties (see [Li, 2001]), but of no relevance for the system developed in this work and the prior art reviewed here. Figure 2.6 gives a quick pictorial overview about MRFs, their components and the basic optimization strategy in the context of image segmentation. The text of this section gives a more mathematical description. The notation and description given here are adopted from [Li, 2001], and tuned towards this constraint.

Representation by labeling of a 2D lattice has already been described in Sect. 2.2.1. A neighborhood system is introduced that defines which sites are considered to be neighbors of another site. The neighborhood of a site (x, y) is represented as the set $\mathcal{N}(x, y)$ of neighboring sites. Different orders of neighborhoods can be defined. In this work, only first and second order neighborhoods are used and discussed. These are the typical 4- and 8-neighborhood systems used in image processing: The 4-neighborhood $\mathcal{N}_4(x, y)$ of a site (x, y) contains the four sites that are horizontally and vertically adjacent to the site on a 2D lattice, while the 8-neighborhood $\mathcal{N}_8(x, y)$ additionally

[4] Algorithms that use level sets or snakes, and algorithms that operate on whole regions are not directly related to the system developed in this work, and a high-detailed background and review is beyond the scope of this work.

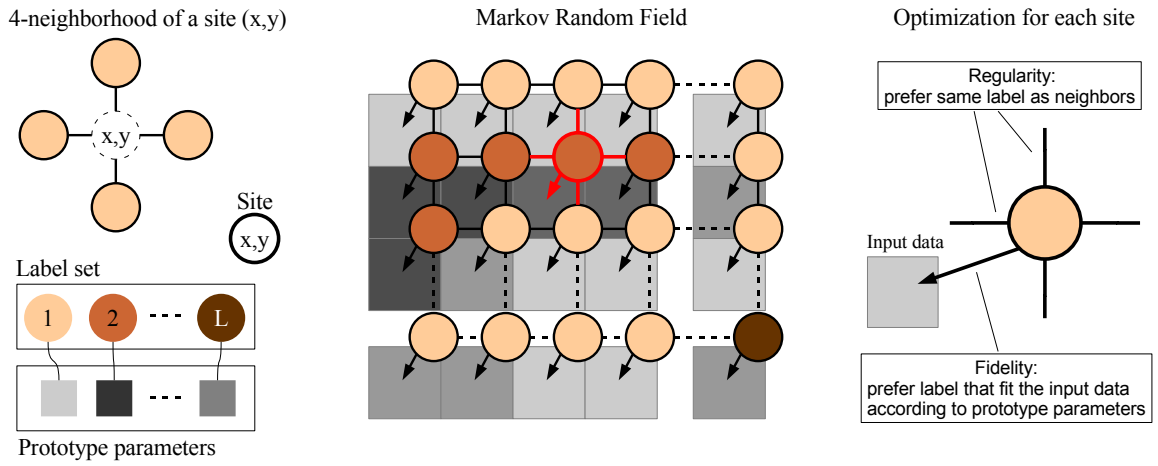


Figure 2.6: Markov Random Fields: an overview. (a) shows the different components, namely sites, a neighborhood definition, and a label set with associated prototype parameters. The optimization (c) visits each site and chooses a label whose prototype parameters are similar to the local image data (fidelity), while trying to achieve a smooth labeling (regularity).

includes the diagonal neighbors

$$\mathcal{N}_4(x,y) = \Big\{(x{+}1,y),\ (x{-}1,y),\ (x,y{+}1),\ (x,y{-}1)\Big\} \tag{2.8}$$

$$\mathcal{N}_8(x,y) = \Big\{(x{+}1,y{+}1),\ (x{-}1,y{+}1),\ (x{+}1,y{-}1),\ (x{-}1,y{-}1)\Big\} \cup \mathcal{N}_4(x,y)\ . \tag{2.9}$$

[Bouthemy and Francois, 1993] and [Lim et al., 2002] use a 8-neighborhood system, and [Wang et al., 2006] consider even higher-order neighborhoods. All other systems reviewed here employ the 4-neighborhood, including the system developed in this work. A set of sites that is fully connected according to the neighborhood criterion is called a *clique* according to graph theory. Different types of cliques are shown in Fig. 2.7: first order neighborhoods contain single-site cliques (a), as well as horizontal and vertical pair-site cliques (b). Second order neighborhoods also contain diagonal pair-site (c), triple-site (d) and quadrupel-site (e) cliques.

The labeling of a lattice with its neighborhood system $\mathcal{N}$ is extended with statistical properties: the event that a site (x,y) has a certain label is seen as a draw from a random variable $F_{x,y}$ associated with that site. $P\big(F_{x,y} = f\,(x,y)\big)$, or in short form $P\big(f\,(x,y)\big)$, is the probability that the random variable $F_{x,y}$ takes the label $f(x,y)$.

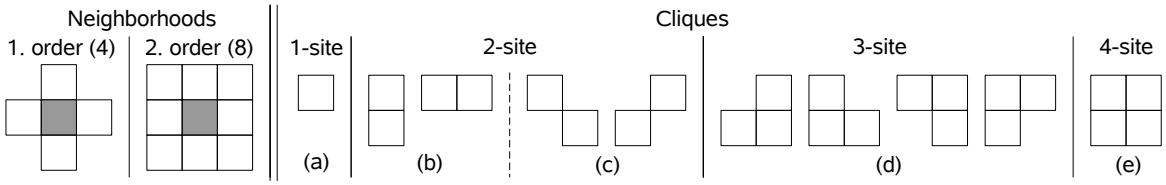


Figure 2.7: Neighborhoods and their cliques. A clique is a group of mutually neighboring sites. The 4-neighborhood contains single-site (a) and horizontal/vertical pair-site cliques (b). The 8-neighborhood also contains diagonal pair-site cliques as well as tripel-site and quadrupel-site cliques (a-e).

The labeling of all sites is grouped in the matrix $\mathbf{f}$, and all random variables are grouped to a random field $\mathbf{F}$ as follows:

$$\mathbf{f} := \begin{bmatrix} f(1,1) & f(2,1) & \cdots & f(X,1) \\ f(1,2) & f(2,2) & \cdots & f(X,2) \\ \vdots & \vdots & & \vdots \\ f(1,Y) & f(2,Y) & \cdots & f(X,Y) \end{bmatrix}, \quad \mathbf{F} := \begin{bmatrix} F_{1,1} & F_{2,1} & \cdots & F_{X,1} \\ F_{1,2} & F_{2,2} & \cdots & F_{X,2} \\ \vdots & \vdots & & \vdots \\ F_{1,Y} & F_{2,Y} & \cdots & F_{X,Y} \end{bmatrix} \tag{2.10}$$

The labeling of all sites, $\mathbf{f}$, is called a *configuration*. The probability of a configuration, $P(\mathbf{f})$, is the joint probability of the labeling of all sites. The set $\mathcal{F}$ contains all possible configurations.

The random field $\mathbf{F}$ is a Markov Random Field (MRF), if and only if these two conditions are fulfilled:

$$P(\mathbf{f}) > 0, \ \ \forall f \in \mathcal{F} \tag{2.11}$$

$$P\big(f\,(x,y) \mid f_{\mathcal{S}-(x,y)}\big) = P\left(f\,(x,y) \mid f_{\mathcal{N}(x,y)}\right) \tag{2.12}$$

where $f_{\mathcal{S}-(x,y)}$ is the set of all labels excluding $f(x,y)$, and $f_{\mathcal{N}(x,y)}$ is the set all labels in the neighborhood of (x,y), not including (x,y) itself.

The *positivity* condition (2.11) implies that $P\big(f(x,y)\big) > 0, \ \ \forall (x,y) \in \mathcal{S}$. The *Markovianity* condition (2.12) demands that the probability for the labeling of any site (x,y) is independent of the labeling of sites that are not in the neighborhood $\mathcal{N}(x,y)$ of that site. This is the equivalent of the Markovianity in Markov processes, where the current state depends on the last state but not on earlier states. The additional property of *homogeneity* requires that $P\big(f\,(x,y) \ \mid \ f_{\mathcal{N}(x,y)}\big)$ is independent of the location (x,y) in

the lattice. This property is true by design for most motion segmentation approaches reviewed here. For more details, see [Li, 2001].

In order to use an MRF for vision applications, the probability distribution of $P(\mathbf{f})$ has to be specified. It must be dependent on the input data, such that the desired outcome modeled by the configuration of the field for a given set of input data is the one with the highest probability. This is done in the design phase. In the application phase, an optimization or search procedure has to determine the most probable configuration for a given input, searching in the usually vast configuration space $\mathcal{F}$.

The property of $\mathbf{F}$ being a Markov Random Field on $\mathcal{S}$ with respect to $\mathcal{N}$ is equivalent to $\mathbf{F}$ being a Gibbs random field (GRF) on $\mathcal{S}$ with respect to $\mathcal{N}$, according to the Hammersley-Clifford theorem [Li, 2001]. Thus, the distribution of $P(\mathbf{f})$ is a *Gibbs distribution* and can be modeled as such:

$$P(\mathbf{f}) = Z^{-1} \cdot e^{-\frac{1}{T}U(\mathbf{f})}, \text{ with } Z = \sum_{\mathbf{f}\in\mathcal{F}} e^{-\frac{1}{T}U(\mathbf{f})} \ . \tag{2.13}$$

The normalizing factor Z is constant and can be omitted, if only the $\mathbf{f}$ that maximizes $P(\mathbf{f})$ needs to be determined, which typically is the case. T is the *temperature* constant and is discussed in Sect. 2.4.4. For now, $T = 1$ is assumed. The energy $U(\mathbf{f})$ is the sum of clique potentials V with respect to the labeling $\mathbf{f}$ for all possible single-site cliques $(x, y) \in \mathcal{C}_1$, pair-site cliques $\big((x, y), (x', y')\big) \in \mathcal{C}_2$ and higher cliques if applicable:

$$U(\mathbf{f}) = \sum_{\mathcal{C}_1} V_1(x, y) + \sum_{\mathcal{C}_2} V_2\big((x, y), (x', y')\big) + ... \tag{2.14}$$

A minimal energy $U(\mathbf{f})$ has the highest probability $P(\mathbf{f})$, thus the optimization procedure has to search for the configuration with the minimal energy.

Now the potential functions V_C have to be defined. They can be dependent on the labeling and the input data, or just the labeling. In a typical motion segmentation application, the single-site potential functions would give low values for sites that have labels associated with motion parameters similar to the input motion estimates for that site. In [Black, 1992] and [Heitz et al., 1991], the "displaced frame difference" is used in the single-site potentials. This difference, also called "motion compensation error" is the brightness difference between a pixel (x, y) in an image $I_t(x, y)$ of time step t

and the same pixel displaced with a motion vector $(\Delta x, \Delta y)$ in the image $I_{t+1}(x, y)$. According to the "brightness constancy assumption" (see Sect. 2.1) this difference should be zero for the correct motion vector. It is directly used in the single-site clique potential function

$$V_1(x, y) = \big(I_t(x, y) - I_{t+1}(x + \Delta x_l, y + \Delta y_l)\big)^2, \text{ with } l = f(x, y) \ . \tag{2.15}$$

The motion vector $(\Delta x_l, \Delta y_l)$ is part of the prototype parameter set $\mathcal{P}_l$, associated with each label l[5]. It can either consist of just two displacement values as shown here, or be based on a higher-level motion model, such as affine transformations. In the single-site potential functions of many proposed systems, e.g. [Bober et al., 1998] or [Bouthemy and Francois, 1993], these motion prototypes are compared to locate image gradients to measure how well a label fits a site. In this work, SSD matching results are used for this purpose.

Pair-site potential functions can be used to enforce a smooth labeling: if a low energy is given for identical labeling of two adjacent sites, and high energy for differing labels, a homogeneous labeling is preferred. This was done for example by [Bouthemy and Francois, 1993], by defining

$$V_2\big((x, y), (x', y')\big) = \mu \cdot \bar{\delta}\big(f(x, y), f(x', y')\big) \tag{2.16}$$

where μ is a penalty constant and $\bar{\delta}(a, b)$ the inverted Kronecker delta function

$$\bar{\delta}(a, b) = \begin{cases} 1, & \text{if } a \neq b \\ 0, & \text{otherwise.} \end{cases} \tag{2.17}$$

The system developed in this work uses a similar formulation.

2.4.2 Motion Discontinuities

If Markov Random Fields are used for segmentation, line processes (LPs) are often used to model discontinuities in a field that is otherwise assumed to be smooth [Li,

[5] If an MRF is used for *motion estimation* without segmentation, the vector $(\Delta x, \Delta y)$ is defined depending on the site (x, y), and contains the motion estimate after convergence.

2001]. A line process is modeled by a binary line field $L\big((x,y),(x',y')\big)$ that provides a binary value for each unordered pair of adjacent sites $\big\{(x,y),\,(x',y')\big\}$ of an MRF. These values can be either switched "ON" or "OFF". An line that is active causes a higher constant cost μ_c, but reduces the cost imposed by smoothing for adjacent sites with a different label. In [Murray and Buxton, 1987] and [Zhang and Hanauer, 1995] such a line process is applied to motion segmentation, modeling motion discontinuities between segment boundaries:

$$V_2'\big((x,y),(x',y')\big) = \begin{cases} \mu_c & \text{if } L\big((x,y),(x',y')\big) \text{ is ON} \\ V_2\big((x,y),(x',y')\big) & \text{otherwise.} \end{cases} \tag{2.18}$$

By replacing V_2 with V_2' in (2.14), the line field is included into the energy function $U(\mathbf{f})$, and its values are obtained from the same optimization procedure that determines the labeling.

2.4.3 Bayesian Formulations

Many authors prefer to model the probability distribution of configurations or labelings $P(\mathbf{f})$ in a Bayesian framework, where the dependency on the observations $\mathbf{o}$ in form of input data $o(x,y)$ is described by the conditional probability $P(\mathbf{f}|\mathbf{o})$. Thomas Bayes postulated in 1763 the theorem that is nowadays known as the *Bayes rule*:

$$P(\mathbf{f}|\mathbf{o}) = \frac{P(\mathbf{o}|\mathbf{f}) \cdot P(\mathbf{f})}{P(\mathbf{o})} \ . \tag{2.19}$$

The first term, $P(\mathbf{f}|\mathbf{o})$ is called the posterior distribution, $P(\mathbf{o}|\mathbf{f})$ the likelihood of the data (observations) given a labeling, and $P(\mathbf{f})$ is the prior distribution. Again, the configuration $\mathbf{f}$ is sought that maximizes the posterior probability. This is called Maximum A-Posteriori (MAP) estimate. The constant term $P(\mathbf{o})$ in the denominator does not effect this search and can be omitted.

If the prior $P(\mathbf{f})$ is "flat", i.e. constant, only the likelihood function $P(\mathbf{o}|\mathbf{f})$ needs to be maximized, and MAP reduces to Maximum Likelihood Estimation (MLE). Such a formulation has been proposed by [Khan and Shah, 2001]. The authors assume that the possible labelings of the sites are of equal probability. The dependence on the input

data is included in their direct formulation of the likelihood function that is maximized through optimization. In other systems, the prior is used for smoothing with definitions based on (2.16). [Vasconcelos and Lippman, 2001] proposed to estimate priors from actual data, rather than just defining them in the design phase.

MAP and Markov Random Fields (MRF) can be combined [Li, 2001, Geman and Geman, 1984]: The posterior distribution of an MRF is modeled as $P(\mathbf{f}|\mathbf{o})$, and the likelihood and the prior are implemented in the clique potential functions. This has also been applied to motion segmentation: a smoothing prior, which is not dependent on the input data, can be implemented in the pair-site clique potentials. The likelihood function, which ties the labeling to the data, is in most cases represented in the potentials of the single-site cliques. The formulation of prior and likelihood function provides an intuitive way to design a segmentation system. Consequently, the actual definition of clique potentials is omitted by several authors, and for the system developed in this work as well. [Murray and Buxton, 1987] follow this approach and formulate a likelihood similar to (2.15) and a prior equivalent to (2.16), but with sums over sites rather than over cliques. [Zhang and Hanauer, 1995] use the same formulation for the likelihood, but extend the prior to model motion discontinuties with a line process.

2.4.4 Optimizing the Configuration

Approaches to design the prior distribution $P(\mathbf{f}|\mathbf{o})$ have been presented in the last subsections. Finding the configuration $\mathbf{f}$ that maximizes this probability for a given input $\mathbf{o}$ is a non-trivial task. The space of possible configurations $\mathcal{F}$ is vast and an exhaustive search for the optimum is computationally intractable: for a labeling on a lattice of the size $X \times Y$ with L labels there are $L^{X \cdot Y}$ combinations.

Whether the prior distribution is defined via clique potentials or via a likelihood and a prior, the segmentation process comes down to minimizing or maximizing a pre-defined energy function, which is a sum of energies computed for each site. Since global optimization is intractable, a local view on MRFs is adopted, which deals with per-site energy optimization.

Simulated Annealing (SA) is a iterative optimization algorithm that operates locally

and nevertheless reaches the global optimum if run for an infinite number of time steps [Geman and Geman, 1984]. It regulates the global temperature parameter T in (2.13) and incorporates it into local optimization. The optimization starts in step $t = 0$ with $T = \infty$, causing $P(\mathbf{f})$ to be a uniform distribution. The temperature is gradually lowered in each iteration cycle, computed by

$$T = \frac{\tau_t}{\log(t+1)} \tag{2.20}$$

where τ_t is a constant factor [Geman and Geman, 1984]. When $T \to 0$ for $t \to \infty$, $P(\mathbf{f})$ is peaked at the optimal result.

In each iteration t, SA visits all sites of the lattice in a certain order[6]. The order can be deterministic, randomized or even asynchronous, as long as no adjacent sites are updated at the same time [Besag, 1986]. During each visit of a site (x, y), the label is l is selected that maximizes this probability:

$$f'(x,y) := \arg\max_{l \in \mathcal{L}} \; P\big(l \mid \mathbf{o}\,(x,y)\big) \tag{2.21}$$

$$= \arg\max_{l \in \mathcal{L}} \; \Big(P\big(\mathbf{o}\,(x,y) \mid l\big) \cdot P\big(l \mid f_{\mathcal{N}(x,y)}\big)\Big)^{\frac{1}{T}} \quad . \tag{2.22}$$

Again, for $T = \infty$, this resolves to the uniform distribution. The lower the temperature is, the more often the best match is picked. For $T \to 0$, this is a greedy selection. The labeling is updated instantly after each visit by setting $f(x, y) = f'(x, y)$.

Among others, [Murray and Buxton, 1987] and [Smith et al., 2004] applied SA to motion segmentation. In theory SA reaches the global optimum, but in practical implementations a lot of computation time is spent on the "burn-in" phase, where labels are flipped randomly. [Besag, 1986] proposed the method of "Iterated Conditional Modes" (ICM), which is equivalent to SA with $T \to 0$ from the very beginning: in each iteration, all sites are visited in a pre-defined order. During each visit of a site (x, y), the site is assigned the label that maximizes (2.21) for $T \to 0$. In contrast to SA it is possible to apply ICM in a fully asynchronous mode, where the update $f = f'$ of the labeling is done after all sites have been visited.

[6] Applied to a discrete lattice, this algorithm is actually a Markov Chain Monte Carlo (MCMC) sampler instead of the "true" SA [Besag, 1986].

Since ICM is a purely greedy selection scheme, it is not necessary to define and compute the likelihood and prior probabilities explicitly as probability distribution functions. The design of an energy function $E(x, y)$ that depends on the input data and the labels of the site (x, y) and its neighbors is sufficient. Typically, this energy function needs to be minimized by varying the label $f(x, y)$, and the label that minimizes $E(x, y)$ is selected for that location.

ICM converges fastly, but can easily be stuck in local minima. Since there is no random flipping in the beginning, the initialization of the field configuration and the label parameters has a strong influence on the time needed for convergence, and on the quality of the result, namely how different the resulting local minimum will be from the global minimum. When images from a longer sequence are segmented, one can use the resulting segmentation and motion parameters of one time step as initialization for the next (e.g. [Bouthemy and Francois, 1993]). This way, information of more than just two frames is implicitly used into the motion segmentation process. Other authors propose to use other cues for this purpose. [Moscheni et al., 1998] initialize the optimization (although a different algorithm) with a color segmentation. [Bober et al., 1998] proposed to use a local robust Hough transform estimator.

ICM has been applied to motion segmentation by several authors, e.g. [Chang et al., 1997, Depommier and Dubois, 1992, Vasconcelos and Lippman, 2001], and in this work as well. [Lim et al., 2002] proposed to run ICM with more visits of sites with motion discontinuities or high motion compensation errors in order to reduce computation time. Other optimization schemes that have been used to improve results and/or to decrease computation time in the context of motion segmentation are "Iterated Conditional Probabilities" (ICP) [Vasconcelos and Lippman, 2001], "Highest Confidence First" (HCF) [Bouthemy and Francois, 1993], "Incremental Stochastic Minimization" (ISM) [Black, 1992], "Simultaneous Over-Relaxation" (SOR) [Black and Anandan, 1996] and the mean field theory [Zhang and Hanauer, 1995].

Multi-scale optimization on a resolution pyramid as proposed by [Stiller, 1997] and [Heitz et al., 1991] can also help to improve the performance. [Bober et al., 1998] presented an approach for scaling the configuration and the cost function based on "super-coupling".

2.4.5 Optimizing the Prototype Parameters

Except for the unlikely case that the prototype parameters $\mathcal{P}_l$ for each label $l \in \mathcal{L}$ are given beforehand, they have to be estimated along with the configuration $\mathbf{f}$. This is a "chicken-and-egg" problem: based on the prototype parameters one can compute a segmentation, and given a good segmentation one can determine the optimal motion parameters.

This cyclic dependendy can be tackled with the iterative two-step Expectation-Maximization (EM) algorithm [Dempster et al., 1977], which has successfully been applied to motion segmentation. In the E-step, the joint probability $P(\mathbf{f}|\mathcal{P}, \mathbf{o})$ of the labeling, conditioned on the input data $\mathbf{o}$ and the current parameter estimate $\mathcal{P}$ is computed. In the M-step, a new parameter set $\mathcal{P}'$ is determined, given the expectation of $P(\mathbf{f})$ conditioned on the input data $\mathbf{o}$. [Vasconcelos and Lippman, 1997] and [Weiss, 1997] explicitly formulated their motion segmentation systems in an EM framework.

In general, the full determination of $P(\mathbf{f})$ is computationally expensive. However, optimization schemes can be used that alternate between configuration and parameter estimation in the same way as the EM algorithm does, but avoid the computation of $P(\mathbf{f})$ by peaking the distribution as in ICM, thus performing a greedy selection [Bouthemy and Francois, 1993, Murray and Buxton, 1987]. With this modification and the assuming that $P(\mathcal{P})$ as a prior is flat, and that the smoothing prior $P(\mathbf{f})$ does not depend on the parameter set $\mathcal{P}$, the E'-step and the M'-step can be computed as follows:

$$\text{E'-step:} \qquad \mathbf{f}' := \arg\max_{\mathbf{f}\in\mathcal{F}} P(\mathbf{f}|\mathcal{P}, \mathbf{o}) = \arg\max_{\mathbf{f}\in\mathcal{F}} \big(P(\mathbf{o}|\mathbf{f}, \mathcal{P}) \cdot P(\mathbf{f})\big) \tag{2.23}$$

$$\text{M'-step:} \qquad \mathcal{P}' := \arg\max_{\mathcal{P}} P(\mathcal{P}|\mathbf{f}', \mathbf{o}) = \arg\max_{\mathcal{P}} P(\mathbf{f}'|\mathcal{P}, \mathbf{o}) \ . \tag{2.24}$$

Several ways of implementing the E'-step have already been discussed in Sect. 2.4.3. For the M'-step, a least squares fit of the motion parameters to the input data can be used. This way an iterative search through the parameter space can be avoided. [Bouthemy and Francois, 1993] proposed this method based on image gradients. In the system developed in this work the motion parameters are fit with least squares to SSD surfaces.

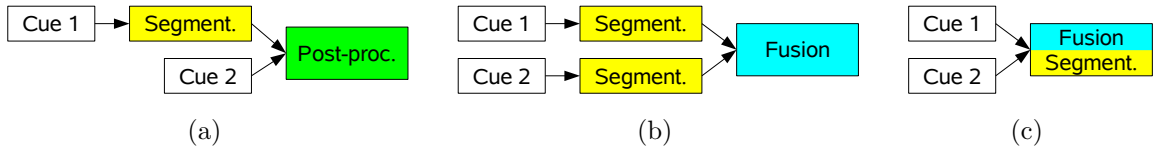
2.4.6 Adjusting the Number of Labels

The number of labels gives the maximum number of different motions in a segmentation. Note that this is not automatically the maximum number of segments, since different disjunct segments can have the same label. If the number of objects present in an image scene is known a-priori, the necessary number of labels can be derived. In most cases these assumptions cannot be made, so there is a need for an estimate. A discussion about ways to estimate the number of different motions from an image sequence is given in [Kanatani and Matsunaga, 2002]. The authors evaluate different measures based on the estimated rank of a matrix containing motion parameters of multiple features tracked over multiple images. Other authors incorporate this adjustment into the optimization process, for example in [Ayer and Sawhney, 1995] by optimizing a MDL based cost-function that sets a trade-off between the encoding costs for the labels and the encoding costs of the residual error. Weiss initializes his mixture estimation framework with "more models than will be needed", and merges them during the estimation process [Weiss, 1997]. More details on this can be found in [Weiss and Adelson, 1996].

Sometimes new labels have to be added during segmentation of a sequence, for example when new objects or motions occur or become uncovered. [Bouthemy and Francois, 1993] introduce a special label with special potential functions, which is assigned to regions that are uncovered according to linear motion prediction. In this work, the initialization procedure is repeated in each frame to detect new motions.

2.5 Multiple Cues

Estimation of motion is critical if problems like occlusions occur or not enough variation of brightness is present in an image region. The latter can easily be the case in interior areas of objects. In these cases, or when more robustness needs to be achieved, the integration of supplementary image information like gray-values, color, edges or stereo-depth is appealing. Several authors proposed to combine different visual modalities (*cues*) in different ways for motion segmentation, as visualized schematically in Fig. 2.8.



(a) (b) (c)

Figure 2.8: Combined vs. sequential multi-cue fusion and segmentation. Sequential approaches use post-processing (a) or fusion of segmentations (b). When fusion and segmentation is combined into one process as depicted in (c), the cues can supplement each other and compensate failures.

2.5.1 Sequential Integration

Multi-cue integration in motion segmentation has mostly been done in a sequential way [Khan and Shah, 2001]. In these cases there is no mutual interaction between information from different cues, only uni-directional dependencies.

One method is to use additional cues for post-processing a segmentation based on pure motion, as depicted in Fig. 2.8a. A first attempt was made in [Thompson, 1980], where grayscale and motion information is combined during a region merging process. In that approach, regions with the same (non-zero) motion vectors are merged. Additionally, regions with ambiguous but non-contradicting motion are merged, if they have a similar contrast. [Yang et al., 2002] try to recognize hand gestures of sign language in image sequences. They apply skin color matching to their motion segmentation result in order to select regions of interest. In [Smith and Brady, 1995] edges are used to refine the boundaries of a segmentation result in a post-processing step. [Meygret and Thonnat, 1990] include stereo depth for 3D grouping of chains (boundary parts) that have already been segmented in 2D. As already mentioned in Sect. 2.4.4, other cues have also been used to initialize the motion segmentation process as a pre-processing step [Moscheni et al., 1998].

Other approaches fuse information of multiple cues after segmentation has been done on them individually, as displayed in Fig. 2.8b. In [Dubuisson and Jain, 1995], a motion mask generated from difference images is combined with color/edge segmentation. The color segmentation is generated independently of the motion segmentation by region merging. It is refined using edges from a Canny edge detector [Canny, 1986]: existing

regions are split if they are broken by an edge. In the following step, regions are added to larger ones if three conditions are fulfilled: their motion parameters are similar, they have a sufficiently long mutual boundary and less than 40% of the pixels on the boundary correspond to edges. Among the candidate regions, the best match according to the first two criteria is selected for merging. The result is fused with a binary motion mask. If more than 90% of a color region is inside the motion mask, it is considered to be completely part of the moving object. If less than 20% is inside, it is completely excluded. Otherwise the decision is left to a smoothing process which is done with a snake. This provides contour refinement as well as boundary extraction (see Sect. 2.2.2).

None of these approaches is based on optimization and labeling, therefore none appears in the classification in Fig. 2.1. Multi-cue approaches that would fall into this category use cue fusion for integration. Examples are given in the next section.

2.5.2 Combined Fusion and Segmentation

The approach of combining fusion and segmentation into one process, as shown in Fig. 2.8c, is beneficial: the possibility of mutual interaction of information from the different cues allows one cue to supplement the other, or to compensate for the failure of another cue in a flexible way.

[Depommier and Dubois, 1992] proposed to couple line processes (LPs) to spatial image gradients (see Sect. 2.4.2). According to their definition, an edge can be inserted into the line field at a lower cost, if it correlates with an intensity step (high gradient) in the image. [Bober et al., 1998] include edges from an edge detection module into motion estimation without introducing an extra line field. When an edge is present between two sites, a cost raised for deviating motion parameters of the sites is reduced. A similar approach has been used in the system developed in this work.

[Kolmogorov et al., 2006] combine stereo matching results with color and contrast. All cues are integrated into one cost function for foreground/background segmentation. For color, Gaussian mixture models are learned from the labeling for both the foreground and the background. The probability for a color to appear, depending on the selection

of one of the two labels, is included in the cost function. Stereo matching quality is used as a measure to identify areas of occlusion, the actual disparity values do not contribute to the segmentation.

[Khan and Shah, 2001] use every pixel's 2D position, 2D motion and YUV color parameters as input data. The parameter sets $\mathcal{P}_l$ of each label l contain not only a motion vector, but color and position parameters as well. All of these are included in the cost function for label assignment. This way, regions are assumed to be not only of homogeneous motion, but of a typical color and location in the image as well. Adaptive weights for the color and motion information are used to suppress the motion term at object boundaries where it is unreliable. The reliability is determined from the fit value of the best matching motion parameter set. The system developed in this work has a similar approach to multi-cue integration, although the adaptive influence of the cues is formulated in a different way.

2.6 Other Properties

2.6.1 Motion Models

The implemented motion models range from simple 2D translational motion to 2D affine [Jepson and Black, 1993] to full 3D rigid body motion [Mitiche and Sekkati, 2006]. Gu et al. propose to use a mixture of basic models like standing, moving with constant velocity or constant acceleration [Gu et al., 1996]. In [Chang et al., 1997] motion is represented as the sum of parametric 2D motion and a non-parametric residual field. Choi and Kim estimate motion stage-wise with increasing complexity [Choi and Kim, 1996].

The motion model that is used for segmentation has a great impact on the results that can be achieved: if big rotational motion is supposed to be segmented into regions with coherent translational motion parameters, many small parts with different motion will be created instead of a single patch with the correct motion parameters. To account for small 3D motions from one frame to the other, 2D affine transformations are generally good enough for segmentation purposes [Smith et al., 2004].

2.6.2 Motion Segments vs. Motion Layers

As defined in the introduction, a segmentation of an image consists of non-overlapping regions that together fully cover an image. These regions are usually assumed to contain opaque objects or parts of objects that move in a coherent, rigid way. This way, occluded parts of objects cannot be represented. In motion segmentation these regions are often referred to as motion layers. In contrast to the regions of an image segmentation, some authors use motion layers that actually do overlap. This is the case for example in [Wang and Adelson, 1994] where the actual segmentation is obtained by a compositing process using depth ordering and opacity information in form of a binary alpha channel of the motion layers. Other examples are approaches where the position and shape of occluded parts of objects is estimated and tracked over time, see Sect. 2.6.4.

The relationship between motion segments and objects varies in different approaches. Some authors want to achieve a segmentation that corresponds to the boundaries of real objects. Others are interested in the number of motions, which can be smaller if multiple objects are moving in the same way, or greater if objects are split up into multiple regions of coherent motion. Besides from erroneous over-segmentation (multiple segments for one object), the latter can be the case whenever the applied motion model cannot account for the motion of an object as a whole. This is the case if rotation is approximated by local translations, non-rigid motion occurs or if objects with significant depth discontinuities are modeled in a 2D motion scheme, e.g. [Weber and Malik, 1997]. In [Kanatani and Matsunaga, 2002] the problem of estimating the number of different motions vs. estimating the number of different objects is discussed.

2.6.3 Dominant Motion vs. Clustering

In [Smith et al., 2004], two different groups of techniques for layered motion estimation are distinguished: One is based on the estimation of “dominant motion”. Here the dominant (background) motion is determined with robust estimators, and outliers are either assigned to one single foreground layer or iterativly divided into multiple motion layers by the same technique, layer by layer. The opposite are “clustering approaches”,

which estimate all motion layers at the same time. Note that this use of the term "clustering" differs from the definition used in the rest of this work (cf. Sect. 2.3.1). For a further discussion of the advantages of both strategies see [Moscheni et al., 1998]. These authors make the same differentiation as above using the terms "top-down" and "bottom-up".

2.6.4 Dealing with Occlusions

In stereo algorithms research, occlusions have been of strong interest in the last two decades [Brown et al., 2003]. In the context of motion segmentation this has been an issue of importance for roughly 10 years. In the literature the term occlusion is used in two different ways: On the one hand it refers to real object occlusion over longer periods of time, which is relevant for tracking the real shape of objects, e.g. [Yilmaz et al., 2004] or [Xiao and Shah, 2005]. On the other hand, occlusion (and disocclusion) is used for newly covered or uncovered areas when advancing from one image frame to the next [Lim et al., 2002, Ogale et al., 2005], as shown in Fig. 2.4b. Here the term is used in the latter definition, since tracking in this work is done only implicitly. If a patch of an image $I(t)$ is compared with displaced patches of the next frame $I(t+1)$, the problem occurs in form of areas that are newly covered in the new frame and do not have a correspondent match. If the matching is done by searching matches for areas in $I(t)$ in the previous frame $I(t-1)$, the problem concerns uncovered areas that have previously been occupied by an object.

As mentioned in Sect. 2.1, occluded areas cause bad matching results in correspondence computation, thus yielding bad motion estimation results. The assignment of those areas to the correct motion layer poses a problem in motion segmentation, independent of the actual method used for motion estimation. The detection of occlusion makes it possible to explicitly deal with them. On top of that, knowledge about areas of occlusion can also help during motion segmentation or in a post processing stage for depth ordering of the motion layers [Wang and Adelson, 1994, Bergen and Meyer, 2000].

Different methods have been presented to detect regions of occlusions. Stiller proposes

to approximate covered and uncovered regions directly from the displacement field of the previous image frame [Stiller, 1997]. If the displacements for two positions point to the same pixel, occlusion is likely to occur. And, respectively, a pixel that has a displacement vector different from zero but is not target of any displacement, is considered to be part of an uncovered region. Of course, this assumes a correct motion estimation for the previous frame in the presence of occlusions.

In [Zhang and Hanauer, 1995] the estimation of a binary occlusion field is included into an energy minimization process of a Markov Random Field for motion estimation: the *motion-compensation error*, which is the squared difference of the brightness of a pixel and the brightness of the pixel regarded as the corresponding one is computed for every pixel and summed up. Areas that are labeled as occlusion areas are excluded from this summation, and are assigned a constant penalty in the prior energy. This way, bad-matching areas will be labeled as regions with occlusions. A similar idea is realized in [Kolmogorov and Zabih, 2001], where pixels that have no matching partner according to a threshold are classified as occluded.

To detect both covered and uncovered areas, matching in both temporal directions is necessary. Depommier and Dubois use a similar method as the above, but with a tri-state field with different labels for occlusion, disocclusion, and "normal" fixed or moving areas. The applied energy function contains the motion compensation errors in respect to the previous and the following image frame [Depommier and Dubois, 1992]. This way, motion information from three images is used.

In [Lim et al., 2002] it is argued that a high motion-compensation error can also be due to other problems in the matching (e.g. noise, illumination changes, motion within the correlation window due to non-rigid motion or motion discontinuities), and low motion-compensation errors can result from false matching in homogeneous background regions. They propose to use matching forwards and backwards in time, as in the approach presented above, but only on two image frames. Binary fields are used for labeling uncovered areas backwards and forwards in time, which is the same as one occlusion and one disocclusion field. They also employ two displacement vector fields for motion estimation from forward and backward matching. These are only defined for the areas that are not affected by occlusion or disocclusion. Again, for pixels labeled

as occlusion there is a constant penalty. The authors claim that a higher robustness can be achieved with their bi-directional optimization.
A very different approach is presented in [Yilmaz et al., 2004]. Here a boundary-based representation of motion regions is used. The authors propose to classify covered and uncovered regions based on abrupt changes in the boundary size or shape.

2.6.5 Limitations

All methods and algorithms that come to use underlie certain limitations like the number, shape and topology of objects that can be represented, or the required computation time. Some base on special restrictive application-specific assumptions, e.g. the planarity of the background in aerial applications [Pless et al., 2000] or a simplified camera model [Weber and Malik, 1997]. A considerable amount of publications concentrates on segmentation with just two motion layers, namely for foreground/background segmentation, e.g. [Kolmogorov et al., 2006]. Others search for a single moving object [Dubuisson and Jain, 1995], which is a similar but not equivalent constraint.

2.7 Conclusion

This chapter presented previous work on motion segmentation, combined with background information. Several criteria have been identified that can be used to characterize approaches to motion segmentation. A classification of the system developed in this work is depicted in Fig. 2.1.
The selection of a motion estimation scheme (dense vs. sparse) has a great impact on the segmentation task. If motion is estimated based on sparse but reliable features, multi-frame trajectories can be used to estimate complex motion models. The segmentation system has to provide means for region formation in between the feature points. If motion estimates are given in a dense form like optical flow, the segmentation system has to deal with noisy data and false estimates caused by occlusions and motion discontinuities. Additional constraints like smoothing can help to obtain a good segmentation.

Optimization-based approaches allow balancing of several information sources and constraints. Bayesian labeling and Markov Random Fields are suitable frameworks for this purpose. The application of Markov Random Fields to motion segmentation represented by labeling on a 2D lattice became popular with the work of [Murray and Buxton, 1987]. In subsequent publications, the combination of motion with edge information and line processes to model motion discontinuities has been proposed. The integration of color and depth information into the optimization procedure as discussed in Sect. 2.5.2 has been attempted more recently by [Khan and Shah, 2001] and [Kolmogorov et al., 2006] and is subject of current research.

A big part of the current state of the art and recent publications regarding motion segmentation employs level sets for boundary representation [Cremers and Soatto, 2005, Mitiche and Sekkati, 2006, Vázquez et al., 2006], as discussed in Sect. 2.2.3. Level sets provide a combination of more explicit boundary representations with the flexibility of labeling approaches. The advantage of optimizing a labeling with only local interdependencies rather than a boundary or level set phase is the possibility of parallelization. If real-time or near-real-time computation is an issue, this might be a crucial factor.

Chapter 3

Multi-Cue Motion Segmentation

The system developed in this work uses labeling on a 2D lattice to represent a segmentation. The labeling is obtained by optimization on a Markov Random Field. The approach to motion estimation proposed in [Lai and Vemuri, 1998] is adopted and integrated into the segmentation procedure. Color, depth and edge information is fused with motion estimates in one cost function during segmentation. False motion estimates caused by occlusions and motion discontinuities are detected and excluded from optimization.

This chapter describes the developed segmentation system. First, a description of the general framework is given. Secondly the implementation for motion segmentation is presented, followed by details on initialization. The approach is then extended to integrate additional visual cues. At last, an algorithmic description is given.

3.1 Framework

The segmentation of an image is specified by a labeling called the "configuration" of a Markov Random Field (MRF) with one label $f(x, y) = l \in \{1, ..., L\}$ for each site (x, y). Related background information is given in Sect. 2.2.1 and Sect. 2.4. The number of labels L can change during optimization and from one frame to the other. Each label l is associated with a prototype parameter set $\mathcal{P}_l$. If the used information is restricted to motion, $\mathcal{P}_l$ only contains a motion vector $(\Delta x_l, \Delta y_l)$. Thus, sites with the same label

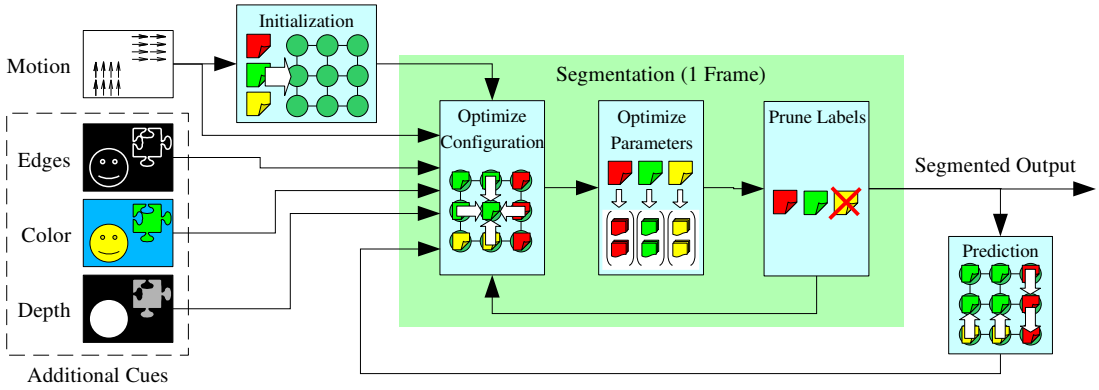


Figure 3.1: Schematic overview of the proposed system. Motion is used for initialization and segmentation. Additional visual cues are integrated. The segmentation of each frame in an image sequence is done with an iterational optimization that alternates between optimizing the configuration and the label parameters. The data resulting from the segmentation in one time step is used as initialization for the following.

are considered to show the same translational motion. As defined in (2.10), $\mathbf{f}$ is the matrix of all labelings $f(x, y)$. The group of motion parameters $\mathcal{P}_l$ of all labels $l \in \mathcal{L}$ is called $\mathcal{P}$.

To determine the labeling and the motion parameters of the labels, an energy function is introduced which needs to be minimized in order to achieve what is assumed to be the optimal segmentation. The global energy is the sum of local energies computed for each site. These energy functions consist of two terms: first the "fidelity" FID, which expresses how well the prototype parameters of a label l fit the local image properties given by $o(x, y)$ (e.g. motion estimates) at each site, as described in the next section. Second a regularization term REG that brings in a smoothness constraint. The fidelity corresponds to the likelihood, and the regularity to the prior probability in a Bayesian framework (see Sect. 2.4.3). For each site $(x, y) \in \mathcal{S}$, the regularization assigns a penalty for each adjacent site (x', y') in the 4-neighborhood $\mathcal{N}(x, y)$ of (x, y) that has a different label than $f(x, y)$:

$$\mathrm{REG}(x, y, l) := \left| \left\{ (x', y') \in \mathcal{N}(x, y) \mid f(x', y') \neq l \right\} \right| . \tag{3.1}$$

With the inverted Kronecker delta function $\bar{\delta}(a, b)$, which is 1 if $a \neq b$ and 0 otherwise

as defined in (2.17), it can be rewritten as follows:

$$\mathrm{REG}(x,y,l) := \sum_{x'=x\pm1} \bar{\delta}\big(l, f\,(x',y)\big) + \sum_{y'=y\pm1} \bar{\delta}\big(l, f\,(x,y')\big) \ . \tag{3.2}$$

Minimization is achieved by varying both the configuration $\mathbf{f}$ and the label parameters $\mathcal{P}$:

$$(\mathbf{f}',\mathcal{P}') := \arg\min_{\mathbf{f},\mathcal{P}} \sum_{(x,y)\in\mathcal{S}} \mathrm{FID}\big(x,y,f\,(x,y)\,,\mathcal{P}_{f(x,y)}\big) + \alpha_{\mathrm{reg}} \cdot \mathrm{REG}\big(x,y,f(x,y)\big) \ . \tag{3.3}$$

The estimates $\mathbf{f}'$ and $\mathcal{P}'$ are used as the new configuration and prototype parameter sets in the next time step. Finding the global minimum by doing an exhaustive search in the configuration space and the label parameter space at the same time is computationally intractable. Instead, the optimization process is split up in alternating phases: in step (a) the configuration is optimized for given label parameters, and in step (b) the label parameters are adapted given the current configuration. These steps are iterated until convergence is reached, yielding an 2-phase optimization in a fashion similar to the Expectation Maximization (EM) algorithm described by [Dempster et al., 1977].
This procedure is executed for each frame in an image sequence, corresponding to the sequence shown in Fig. 3.1 and the algorithmic description in Sect. 3.4:

1. Initialize configuration and labels
2. Iterate until convergence or maximum step number is reached:
   2.1. Optimize configuration (a)
   2.2. Optimize motion parameters (b)
   2.3. Remove unused and duplicate labels
3. Apply motion model for prediction
4. Output the configuration as resulting segmentation

The actual optimization steps employ a greedy local optimization scheme called *iterated conditional modes* (ICM) as described in Sect. 2.4.4, which is identical to *Simulated Annealing* with a minimal temperature from the very beginning [Besag, 1986].
The update for the configuration in each iteration is done for all sites in a random order, always selecting the label for a site which minimizes the energy function for that

site:

$$f'(x,y) := \arg\min_{l} \left(\mathrm{FID}\,(x,y,l,\mathcal{P}_l) + \alpha_{\mathrm{reg}} \cdot \mathrm{REG}\,(x,y,l) \right) \quad . \tag{3.4}$$

The update for the label parameters in each iteration is done by selecting the parameters for each label that minimize the sum of the energies of all sites carrying that label. Since the regularity term does not depend on the label parameters, it can be omitted:

$$\mathcal{P}_l := \arg\min_{\mathcal{P}} \sum_{\mathcal{R}_l} \mathrm{FID}\,(x,y,l,\mathcal{P}_l) \quad . \tag{3.5}$$

3.2 Segmentation Based on Motion Information

In this work, local image motion is estimated by analyzing correspondences with a straight-forward similarity matching scheme broadly used for optical flow estimation. As discussed in Sect. 2.1, the problematic selection of the "winning" match is deferred to the segmentation process. Therefore, matching results for all displacements inside a defined search window are obtained and included in the optimization framework, as proposed in [Lai and Vemuri, 1998] for pure motion estimation.

Popular similarity measures for correspondence analysis are the sum of absolute difference (SAD), sum of squared differences (SSD) and cross-correlation-based measures, with different types of normalization. In principle, all of them can be used in the segmentation system proposed in this work.

For the remainder of this work, the sum of squared differences (SSD) has been chosen as the measure with both good performance in the presence of image noise and fast computation time. This choice is based on experimental comparisons, the description of the procedure and results of the experiments are given in Sect. A as part of the appendix.

SSD surfaces directly resemble the motion fidelity

$$\mathrm{FID}_{\mathrm{mot}}(x,y,l) := SSD(x,y,\Delta x_l, \Delta y_l) \quad . \tag{3.6}$$

If no other cues are being used, the fidelity for the optimization is defined as

$$\mathrm{FID} := \alpha_{\mathrm{mot}} \cdot \mathrm{FID}_{\mathrm{mot}} \tag{3.7}$$

where α_{mot} is a constant weight factor.

3.2.1 Detection of Occlusions

To detect areas of occlusion where image changes cannot be explained by motion, the minimum of an SSD surface is used as a measure for its validity. Only if there is a low minimum in an SSD surface, the respective motion can account for the brightness changes in the correlation window at that particular location. When the minimum is higher than a threshold τ_{occ}, the matching is assumed to be bogus. Such SSD surfaces are completely set to zero, thus they do not affect motion segmentation:

$$\min_{u,v}\{SSD(x,y,u,v)\} > \tau_{occ} \Longrightarrow SSD(x,y,\Delta x,\Delta y) := 0, \;\; \forall \Delta x, \Delta y \;\; . \tag{3.8}$$

3.2.2 Optimization of Labels

The motion parameters of the labels have to be updated along with the configuration in each iteration step. For each label l the displacement values are chosen that belong to the minimum in the sum of all SSD surfaces of sites with label l. Hence, the new motion parameters are the least-squares fit for the involved motion estimates:

$$(\Delta x_l, \Delta y_l) := \arg \min_{(\Delta x, \Delta y)} \sum_{(x,y)\in\mathcal{R}_l} SSD(x,y,\Delta x,\Delta y) \;\; . \tag{3.9}$$

Labels that are not assigned to any site are deleted from the list. Labels with identical motion parameters are unified. This way the total number of labels L can decrease during iteration.

3.2.3 Initialization

Besides a generic background label with $\Delta x = \Delta y = 0$, the initial labels are determined by searching for "good" motion estimates in the SSD surfaces. These have to meet all of the following criteria: (a) Motion needs to be clearly present, which is indicated by bad matching results for $\Delta x = \Delta y = 0$. This condition is considered to be fulfilled at all positions (x, y) where $SSD(x, y, 0, 0) > \tau_{mov}$ with $\tau_{mov} = 5$. (b) Occlusion areas are excluded, they are detected as described by (3.8). (c) The correspondence matches have to be unambiguous, which corresponds to a peaked minimum in an SSD surface.

The configuration of the sites is initialized by randomly assigning labels to the sites. When segmenting a sequence of images, the label parameters are carried over to the next frame. To account for dramatic changes or new occurrences of motion, the initialization procedure described above is recurred in each frame to include new additional labels. The configuration is also used in the next frame as a prediction for the positions and shape of the segments. The motion parameters of the labels are applied to each site, yielding a linear predictive motion model:

$$\forall x, y : f_{t+1}(x + \Delta x_l, y + \Delta y_l) := f_t(x, y), \text{ with } l = f_t(x, y) \ . \tag{3.10}$$

3.2.4 Discussion

This section has presented a system for motion segmentation based on SSD matching. By including full SSD surfaces instead of just the best match, uncertain information in the case of ambiguities is integrated. Areas where no motion information is present due to missing contrast cannot be assigned to the correct segment based on the fidelity term alone. To a certain degree, the regularization causes a filling-in of those areas. This is demonstrated experimentally in Sect. 4.2.1. This filling-in process is rather slow – in the worst case the labeling propagates with only one site per iteration on the whole field, depending on the update order. For large areas without motion, the filling-in can fail completely. Also, if a moving objects stops its motion, the segmentation is lost for the new frames. To deal with these problems and to achieve a higher general robustness, supplementary visual cues carrying complementary information are introduced in the next section.

3.3 Integration of Additional Visual Cues

All visual information that comes to use here besides motion estimation is assumed to be provided on a low-level basis as raster images. Specifically, color information, edge images, and disparity maps obtained from a stereo camera are integrated. However, the approach is easily extentable to include further cues like for example gray level or color edges.

3.3.1 Categorization

Two different kinds of cues can be distinguished. The system developed in this work is able to include the cues in each of the two categories in a uniform manner: "areal cues" and "edge cues". The properties and integration concept of these two classes is presented in the following. The remainder of this section deals with the implementation of concrete representatives of these classes.

Areal Cues

These cues are assumed to vary smoothly across the image, and have representative properties for objects, e.g. color, motion and depth information. Cues in this category are integrated into the fidelity term of the energy functions as additional summands, with a factor α that controls their influence. For each cue, a prototype parameter vector is added to the label parameter sets $\mathcal{P}_l$. These parameters need to be adapted during optimization. The prototype parameters for each label as a whole represent a kind of basic object knowledge.

Edge Cues

These cues contain explicit information about discontinuities, which are expected to indicate object boundaries. In relation to areal cues they are of a *differential* nature, and can mostly be computed from corresponding areal cues, e.g. edges in grayscale, color or depth maps. Edges are integrated in the regularity term of the energy functions: the penalty for unequal adjacent labels is reduced at edges, since they are assumed to be coherent with object boundaries, ergo with desired segment boundaries. Again, a factor α controls the influence of each cue.

3.3.2 Color Information

Image data for this cue are given by hue and saturation values $H(x, y)$ and $S(x, y)$ for each pixel (x, y). These data are assumed to be discretized into $b = 10$ equally distributed bins. Prototype color parameters for each label are given by a normalized

two-dimensional $b \times b$ color histogram $c_l(h, s)$ of hue and saturation values. Thus, sites with the same label are considered to show similar coloring or a similar choice of colors. For every pixel, the histogram value corresponding to the pixel's color is subtracted from the energy. This way, low energy is achieved for labels whenever the pixel's color falls into histogram bins with a high value. Thus, labels with typical color information for that pixel are preferred when the configuration gets optimized:

$$\mathrm{FID}_{\mathrm{col}}(x, y, l) = -c_l\left(H\left(x, y\right), S\left(x, y\right)\right) \quad . \tag{3.11}$$

When adapting the label parameters, a normalized color histogram $c_l(h, s)$ of any label l is computed by accumulating the hue and saturation values $H(x, y)$ and $S(x, y)$ of all sites in the region with label l:

$$c_l(h, s) = \frac{\left|\left\{(x, y) \in \mathcal{R}_l \;\middle|\; \left(H(x, y) = h\right) \wedge \left(S(x, y) = s\right)\right\}\right|}{\left|\mathcal{R}_l\right|} \quad . \tag{3.12}$$

Here the numerator resembles the number of cells in the region $\mathcal{R}_l$, with hue h and saturation s. The denominator stands for the total number of cells in the region $\mathcal{R}$ with the label l. This cue uses histograms instead of single float values as prototypes, in order to accommodate color mixtures within one label. Otherwise, regions with coherent motion could be split up into multiple segments, if they show different coloring. For a first initialization, the histograms are set to be flat. This way, they are build up after the first iteration, exploiting segmentation of coherent motion estimates. When advancing to the next frame in a sequence, the color histograms are preserved with the rest of the prototype parameters.

3.3.3 Depth Information

The distance measures for this cue are given by $d(x, y)$. They are assumed to be proportional to the distance from the camera to the object seen at each pixel. From disparity values, these are obtained by simply computing the reciprocal value. Locations (x, y) where no distance information is available, e.g. due to low brightness contrast in stereo computation, are marked with $d(x, y) := -1$. For each label, a prototype distance

d_l is added to the parameter set. Like for all area cues, depth information is integrated into the fidelity term by comparing the local image features to the prototypes:

$$\mathrm{FID}_{\mathrm{dep}}(x,y,l) = \begin{cases} \left(d\,(x,y) - d_l\right)^2 & \text{if } d(x,y) \geq 0 \text{ and } d_l \geq 0 \\ \tilde{d} & \text{otherwise .} \end{cases} \tag{3.13}$$

The case selection in this equation has to account for two problems: first, if there is no depth information present at a position (x, y), $\mathrm{FID}_{\mathrm{dep}}(x, y, l)$ is the same for all labels and does not make a difference in the minimization process. Second, there may be labels l without a valid depth prototype, indicated by $d_l := -1$. This is the case when only sites without a valid depth estimate are assigned with these labels. For these, FID_{dep} has to be set to a default value $\tilde{d} = 1$, while it is the normal squared difference for the others.

During optimization of the label parameters, the prototype distances d_l are set to the mean of the distance values of associated sites:

$$d_l = \frac{\sum\limits_{(x,y)\in\mathcal{R}_l} d(x,y)}{\left|\mathcal{R}_l\right|} \quad . \tag{3.14}$$

The distance prototypes of labels that have no sites associated with them are set to $d_l = -1$. As with the color and motion prototypes, the distance parameters in the labels from one frame are carried over as initialization to the following frame. If the depth cue is used, the distance parameters play a role in the pruning of labels as well: only if the motion parameters of two labels i and j are equal *and* the absolute distance of their depth prototypes is smaller than the threshold τ_d, the labels are unified.

3.3.4 Brightness Edges

Edge information is obtained with the Canny edge detector [Canny, 1986]. It is represented with a binary edge map $e(x, y)$, which shows 1 for 'edge', 0 for 'no edge'. The penalty for a neighboring site (x', y') is reduced if a brightness edge goes through exactly one of the two sites with image coordinates (x, y) and (x', y') respectively. This way, it is possible to favor diverting labels at edges where object boundaries are likely

to occur. To include the edges, the regularization term is redefined as follows:

$$\begin{aligned}\mathrm{REG}(x,y,l) = &\sum_{x'=x\pm1} \bar{\delta}\big(l, s\,(x',y)\big) \cdot \Big(1 - \alpha_{\mathrm{edg}} \cdot \bar{\delta}\big(e\,(x,y)\,, e\,(x',y)\big)\Big) \\ &+ \sum_{y'=y\pm1} \bar{\delta}\big(l, f\,(x,y')\big) \cdot \Big(1 - \alpha_{\mathrm{edg}} \cdot \bar{\delta}\big(e\,(x,y)\,, e\,(x,y')\big)\Big) \ . \end{aligned} \tag{3.15}$$

3.3.5 Discussion

Applicable visual cues have been categorized into areal and edge cues. Representatives and the implementation of their integration from both categories, namely color, distance and brightness edges, have been presented here. While the edge cues are included in the regularity, the area cues are integrated via the fidelity term of the cost function:

$$\mathrm{FID} = \alpha_{\mathrm{mot}} \cdot \mathrm{FID}_{\mathrm{mot}} + \alpha_{\mathrm{col}} \cdot \mathrm{FID}_{\mathrm{col}} + \alpha_{\mathrm{dep}} \cdot \mathrm{FID}_{\mathrm{dep}} \ . \tag{3.16}$$

3.4 Algorithmic Description

This section gives an algorithmic description of the proposed system. The steps correspond to the boxes in Fig. 3.1.

Input

/ Scalar parameters with typical values */*

1 $X = 160,\ \ Y = 120$ *// Image width and height*

2 $N = 5$ *// Correlation window size*

3 $M = 15$ *// Search window size*

4 $T = 30$ *// Maximum number of time steps*

/ Image data of size $X \times Y$ */*

5 $\mathtt{SSD}(\Delta x, \Delta y, x, y)$ *// SSD surfaces of size $M \times M \times X \times Y$*

6 $H(x,y)$, $S(x,y)$ *// Hue and saturation image, quantized into b bins*

7 $d(x,y)$ *// Depth image*

8 $e(x,y)$ *// Binary edge image*

9 $f(x,y)$ *// Initialized labeling*

/ Prototype parameters of labels */*

10 $\Delta x_l, \Delta y_l$ *// Initialized motion vectors*

11 $c_l(h,s)$ *// Color histograms*

12 d_l // *Depth values*

13 $\tau_d = 0.02$ // *Similarity margin for depth values*

Algorithm

Initialization

14 $f_h(l) = 0,$ `for` $l = 1, \ldots, L$ // *Label histogram for pruning*

15 $E(l) = 0,$ `for` $l = 1, \ldots, L$ // *Local energy*

16 $t = 0;$ // *counting the time steps*

/* *Generate coordinate arrays* z_x *and* z_y */

17 $i = 0;$

18 `for` $x = 1, \ldots, X$

19 `for` $y = 1, \ldots, Y$

20 $z_x(i) = x;$

21 $z_y(i) = y;$

22 $i = i + 1;$

23 `end`

24 `end`

/* *Create array order with random permutation of indices for coordinate arrays* */

25 $\texttt{order} = \texttt{rndpermute}(1, \ldots, X \cdot Y);$

Optimization of configuration

Label 1:

26 $\texttt{changes} = \texttt{false};$ // *indicates changes for convergence test*

27 `for` $i = 1, \ldots, X \cdot Y$ // *for all sites*

28 $x = z_x\big(\texttt{order}(i)\big);$

29 $y = z_y\big(\texttt{order}(i)\big);$

30 `for` $l = 1 \ldots L$ // *for all labels*

31 $\texttt{FID}_{\texttt{mot}} = \texttt{SSD}(\Delta x_l, \Delta y_l, x, y);$

32 $\texttt{FID}_{\texttt{col}} = -c_l\big(H(x,y), S(x,y)\big);$

33 `if` $d(x,y) \geq 0$ `and` $d_l \geq 0$ // *check for valid depth estimates*

34 $\texttt{FID}_{\texttt{dep}} = \big(d_l - d(x,y)\big)^2;$

35 `else`

36 $\texttt{FID}_{\texttt{dep}} = \tilde{d};$ // $\tilde{d} = 1$

37 `end`

38 $\texttt{REG} = 0;$

39 `for` $x' = x - 1, x + 1$ // *for both horizontal neighbors*

40 $\texttt{REG} = \texttt{REG} + \bar{\delta}\big(l, f(x', y)\big) \cdot \Big(1 - \alpha_{\texttt{edg}} \bar{\delta}\big(e(x,y), e(x',y)\big)\Big);$

41 `end`

```
42          for y' = y − 1, y + 1                                   // for both vertical neighbors
43              REG = REG + δ̄(l, f(x, y')) · (1 − α_edg δ̄(e(x, y), e(x, y')));
44          end
45          E(l) = FID_mot + α_col · FID_col + α_dep · FID_dep + α_reg · REG;
46      end
47      l_new = arg min_l E(l),  with  l = 1, ..., L;
48      if  f(x, y) ≠ l_new
49          changes = true;                                         // label of a site has changed
50      end
51      f(x, y) = l_new;
52      f_h(l) = f_h(l) + 1;                                        // count the label assignment
53  end
```

Optimization of parameters

```
54  d_l = 0;                                                        // Reset mean depth values
55  c_l(h, s) = 0;                                                  // Reset color histograms
56  for  i = 1, ..., X · Y                                          // for all sites
57      x = z_x(i);
58      y = z_y(i);
59      l = f(x, y);
60      for  Δx = −k, ··· , k                                       // Accumulates the SSD surfaces
61          for  Δy = −k, ··· , k
62              SSD_cum(Δx, Δy, l) = SSD_cum(Δx, Δy, l) + SSD(Δx, Δy, x, y);
63          end
64      end
65      d_l = d_l + d(x, y);                                        // Accumulate depth value for mean
66      h = H(x, y);
67      s = S(x, y);
68      c_l(h, s) = c_l(h, s) + 1;                                  // Accumulate for histogram
69  end
70  (Δx_l, Δy_l) = arg min_(Δx,Δy) SSD_cum(Δx, Δy),  with  Δx, Δy = −k, ..., +k;
71  d_l = d_l / f_h(l);                                             // Divide by number of members to compute mean
72  c_l(h, s) = c_l(h, s) / f_h(l),  for  h, s = 1, ··· , b;        // Normalize histograms
73  end
```

Prune labels

```
74  u = 0;                                                          // counts unused labels
75  s(l) = 0,  with  l = 1, ..., L;                                // stores label numbers for substitution
76  dupe = false;                                                   // indicates detection of a duplicate label
```

```
      /* find unused and duplicate labels */
77    for i = 1,... L
78       if f_h(i) == 0
79          u = u + 1;                                          // label i is unused
80       else
81          for k = 1,..., i - 1                                // check for duplicates
82             if l_h(k) > 0 and Δx_i == Δx_k and Δy_i == Δy_k
83                if α_dep == 0 or abs(d_i - d_k) < τ_d
84                   u = u + 1;
85                   s(i) = s(k);                              // plan to substitute label i with label k
86                   dupe = true;
87                   break;
88                end
89             end
90          end
91       end
92       if dupe == false
93          s(i) = i - u;
94       end
95       dupe = false;
96    end
97    if u > 0
98       changes = true;
99    end
      /* update prototype parameters */
100   k = 0;
101   for i = 1,..., L - u
102      while s(k) < i
103         k = k + 1;
104      end
105      Δx_i = Δx_k;
106      Δy_i = Δy_k;
107      d_i = d_k;
108   end
109   c'_l(h, s) = 0,  for h, s = 1, ··· , b;                   // new color histograms
110   f'_h(l) = 0,  for l = 1,..., L - u;
111   for i = 1,..., L
```

```
112         if l_h(i) > 0
113             f'_h(s(i)) = f'_h(s(i)) + f_h(i);
114             c'_{s(i)}(h,s) = c'_{s(i)}(h,s) + f_h(i) · c_i(h,s),  for h,s = 1,··· ,b;    // cumulate histogr.
115         end
116     end
117     for i = 1,...,L − u
118         c_i(h,s) = c'_i(h,s)/f'_h(l),  for h,s = 1,··· ,b;    // Overwrite and normalize histograms
119     end
120     L = L − u;
        /* update labeling */
121     for i = 1,...,X · Y                                                    // for all sites
122         x = z_x(i);
123         y = z_y(i);
124         f(x,y) = s(f(x,y));
125     end
126     t = t + 1;
127     if changes == true and t < T
128         GOTO 1;
129     end
```

Apply motion model

```
130     f'(x,y) = 0,  for all (x,y);
131     for (x,y)
132         dx = Δx_{f(x,y)};
133         dy = Δy_{f(x,y)};
134         if f'(x + dx, y + dy) == 0
135             f'(x + dx, y + dy) = f(x,y);
136         end
137     end
```

Output

```
138     f(x,y)                                                          // Labeling
139     f'(x,y)                                                         // Prediction of labeling
140     Δx_l, Δy_l                                                      // Motion parameters of labels
141     c_l(h,s)                                                        // Color histograms
142     d_l                                                             // Depth values
```

Chapter 4

Analysis and Experiments

This section presents experiments evaluating the performance of the proposed system. The influence of regularization and occlusion handling on the result is shown, and the advantage of including additional cues is demonstrated. The system is tested on rendered images, stereo camera images and standard video sequences.

The rendered images have been generated by the author of this work. They show colored objects moving in front of a structured background. Ground truth data is available in the form of a reference segmentation output by the render system. Besides the availability of ground truth information, rendered images have the advantage of being controllable: the amount of contrast in the image and velocity of motion can be easily adjusted.

Camera images are used to test the system's performance under real-world conditions. The images have been taken by the author using a stereo camera system[1] that computes disparity maps at 25 frames per second in hardware on the camera.

To allow comparison with other systems, commonly used standard video sequences from an MPEG encoding benchmark set are used as well[2]. These sequences do not provide ground truth information.

[1] Videre Design STH-DCSG-STOC (STereo On a Chip), 640×480 pixels, monochrome or color

[2] available at http://videonet.ece.missouri.edu/download.htm

4.1 Quantitative Evaluation of Segmentations

In most publications on motion segmentation, the empirical evaluation of algorithms consists of subjective judgement of segmentation results, e.g. [Altunbasak et al., 1998, Bober et al., 1998, Cremers and Soatto, 2005, Khan and Shah, 2001, Moscheni et al., 1998, Ogale et al., 2005, Sim and Park, 1998, Yilmaz et al., 2004]. For general image segmentation there exist databases with images and ground truth data, e.g. the Berkeley Segmentation Dataset[3], but these databases only contain still images. To obtain comparable results, several MPEG reference sequences[4] are used for evaluation without ground truth data, e.g. more recently by [Khan and Shah, 2001, Lim et al., 2002, Smith et al., 2004, Vasconcelos and Lippman, 2001, Xiao and Shah, 2005].
To compare the performance of different approaches and to empirically determine good parameter settings, an automatic quantitative evaluation of segmentation results is desirable. A review and characterization of different quantitative methods for evaluation of general image segmentation systems has been presented by [Zhang, 1996] and updated in [Zhang, 2001]. A theoretical analysis of segmentation algorithms can provide information about their computational complexity and convergence behaviour. Further, the author distinguishes two types of data driven evaluation: empirical goodness methods that rate actual segmentation results, and empirical discrepancy methods that compare segmentation results to ground truth data. These two groups are discussed in the remainder of this section, and an empirical discrepancy method is introduced that is used for the experiments in this work.

4.1.1 Empirical Goodness Methods

The group of empirical goodness evaluation methods comprises quantitative measures that evaluate segmentation results for given test data. In [Zhang, 1996], intra-region conformity, inter-region contrast and region shape are proposed. These measures are an integral part of most optimization-based motion segmentation systems and are usually not used for evaluation of the same.

[3]available at http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping
[4]available at http://videonet.ece.missouri.edu/download.htm

To compare results with different parameter settings for one segmentation system, the global energy after optimization can be used as a quality measure. This method is appropriate if different initialization or optimization strategies are compared. If however the design or parameterization of the energy function itself is evaluated, the use of this method is of course problematic.
[Vasconcelos and Lippman, 1997] used the convergence time to evaluate their segmentation results for different parameter settings. Of course this has to be combined with other quality measures, the convergence time alone is not sufficient if the segmentation results vary for different parameter settings.

4.1.2 Empirical Discrepancy Methods

Evaluating the empirical discrepancy means to compare segmentation results to ground truth data. In the context of motion estimation systems, ground truth data is often integrated in the form of error values for 2D or 3D rotation and translation, obtained with an electromagnetic tracker or from known motor actions. This evaluation system has been applied to motion segmentation in cases where the number of motion layers is small, for example in combination with feature-based motion estimation in [Debrunner and Ahuja, 1998, Huang et al., 1995, Veenman et al., 2001, Wang and Duncan, 1996] and level set representations [Mitiche and Sekkati, 2006].
If ground truth motion information is not available, the quality of a motion estimate can be judged by the *image reconstruction error*. To compute this measure for a motion estimate, a prediction image $\hat{I}_{t+1}(x, y)$ based on an image $I_t(x, y)$ of time step t and the corresponding motion estimate is generated, as defined in (3.10). The quality is determined by the mean square error MSE of the prediction:

$$\mathrm{MSE} = \frac{1}{X \cdot Y} \cdot \sum_{x=1}^{X} \sum_{y=1}^{Y} \left(\hat{I}_{t+1}\left(x, y\right) - I_{t+1}\left(x, y\right) \right)^2 \ . \tag{4.1}$$

This measure has been applied to motion segmentation by [Lim et al., 2002] in the form of a *peak signal-to-noise ratio* defined by

$$\mathrm{PSNR} = 10 \cdot \log_{10} \frac{1}{\mathrm{MSE}} \ . \tag{4.2}$$

Performance measures based on the image reconstruction error only give information about the quality of the corresponding motion estimate. To evaluate motion segmentation this alone is not sufficient, since the congruence of segments with real entities is not taken into account.

To evaluate the actual segmentation quality, reference labelings can be used as ground truth data. For synthetic data these labelings can be generated automatically, for real images they have to be created by human experts. The latter introduces a subjectivity, since different experts will most likely create different segmentations. [Unnikrishnan et al., 2005] proposed an approach to evaluate segmentations using multiple reference labelings.

From a segmentation $f(x, y) \in \{1, \ldots, L\}$ and the corresponding ground truth data $g(x, y) \in \{1, \ldots, L_g\}$, a confusion matrix $\mathbf{C}_{i,j}$ can be constructed

$$\mathbf{C}_{i,j} := \left| \left\{ (x, y) \in \mathcal{R}_i \mid g(x, y) = j \right\} \right| . \tag{4.3}$$

This matrix contains in each cell (i, j) the number of sites with segmentation label i and ground truth label j.

If the mapping $m(l)$ from ground truth labels to segmentation labels is known, the percentage of mis-classified sites $q(l)$ for each ground truth label l is a straightforward measure, and can be computed directly from the confusion matrix as discussed in [Zhang, 1996], by summing over the columns with

$$q(l) := 100 \cdot \frac{\left(\sum_{i=1}^{L} C(i, l) \right) - C_{m(l),l}}{\sum_{i=1}^{L} C_{i,l}} \tag{4.4}$$

where the mapping $m(l)$ provides the segmentation label corresponding to the ground truth label l. Equivalently, the percentage of misses for each label can be computed by summing over the rows.

In many cases, however, this method is problematic: if the segmentation is based on optimization and the prototype parameters of the labels are not given a-priori, the mapping $m(l)$ is not known and has to be determined manually or with heuristics for each resulting labeling individually.

Input

1 $C_{i,j}$ `with` $i = 1, \ldots, L$ `and` $j = 1, \ldots, L_g$ *// Confusion matrix*

2 X, Y *// Size of the segmentation*

Algorithm

3 $m(l) := 0,$ `for` $l = 1, \ldots, L_g$`;` *// mapping array*

4 $q' := 0$`;`

5 `while` $\sum_{i,j} C(i,j)$`>0`

6 $(i', j') := \arg\max_{i,j} C(i,j)$`;`

7 $q' := q' + \sum_i C(i,j') + \sum_j C(i',j) - 2 \cdot C(i',j')$`;`

8 $C_{i',j} := 0,$ `for` $j = 1, \ldots, L_g$`;`

9 $C_{i,j'} := 0,$ `for` $i = 1, \ldots, L$`;`

10 $m(j') := i'$`;`

11 `end`

12 $q = q'/(X \cdot Y)$`;`

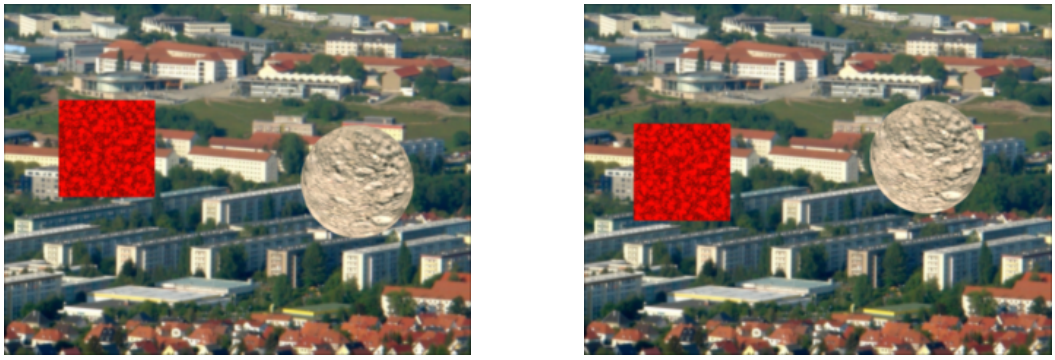
Output

13 $m(l)$ *// mapping*

14 q *// number of mis-classified sites*

Figure 4.1: Evaluation of segmentation quality based on the confusion matrix $\mathbf{C}_{i,j}$.

The algorithm in Fig. 4.1 determines such a mapping $m(l)$ and computes the relative number of mis-classified sites q for a given confusion matrix $\mathbf{C}_{i,j}$. It chooses the cell (i', j') with the maximum value in $\mathbf{C}_{i,j}$ and assumes that label i' corresponds to the ground truth label j'. The sum of all values in the row and column of the cell (i', j') excluding (i', j') itself is the number of sites with label i' that have a different ground truth label than j' plus the number of sites that have the ground truth label j' but carry a different label than i'. This sum is added cumulatively to yield the total number of mis-classified sites q', which is divided by the number of sites $X \cdot Y$ to obtain the relative number of mis-classified sites q. The values of row i' and the column j' are set to zero because they have already been identified either as correct or wrong, and the process is repeated until the matrix is empty.



(a) Frame 6 (b) Frame 9

Figure 4.2: Two example images from the rendered test sequence "seq1" used for tests with regularization (Fig. 4.3) and occlusions (Fig. 4.4).

This algorithm is equivalent to reordering the rows and columns of $\mathbf{C}_{i,j}$ to bring the highest values onto the main diagonal[5], and summing over all cells excluding the main diagonal.

The normalized number of mis-classified sites q is used to quantitatively evaluate the segmentation performance in the experiments.

4.2 Experiments with Rendered Images

In this set of experiments, segmentation is done using motion alone. The prototype parameters of the labels contain a 2D translation vector, and the fidelity term of the energy functions only includes SSD surfaces, as described in Sect. 3.2. A fixed weight $\alpha_{\text{mot}} = 100$ is used to bring the SSD values up to scale. All other cues have been disabled. The SSD surfaces are obtained using a 5×5 pixel correlation window, and a 15×15 pixel search window.

In each experiment with quantitative evaluation, results are averaged over 10 trials. In each trial, the system is run on a rendered sequence, 10 frames in a row. For each frame, the segmentation error q, i.e. percentage of mis-classified sites, is computed when optimization has converged. Two images from the sequence are shown in Fig. 4.2.

[5] If $\mathbf{C}_{i,j}$ is not quadradic, the main diagonal here is the set of cells $\mathbf{C}_{k,k}$ with $k = 1, \ldots, \min(L, L_g)$ anyway.

4.2.1 Regularity

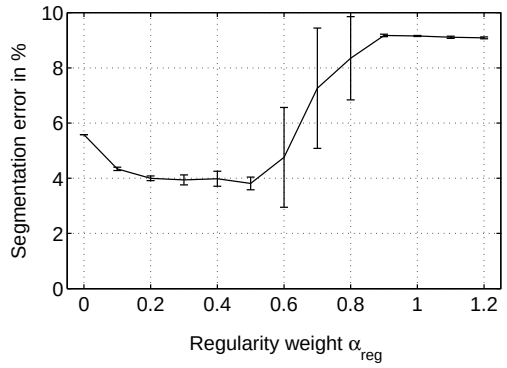
This experiment tests the system's performance for different amounts of regularization, which is controlled by varying the weight parameter α_{reg}. All other cues and occlusion handling are disabled. The results for this experiment are presented and discussed in Fig. 4.3.

4.2.2 Occlusions

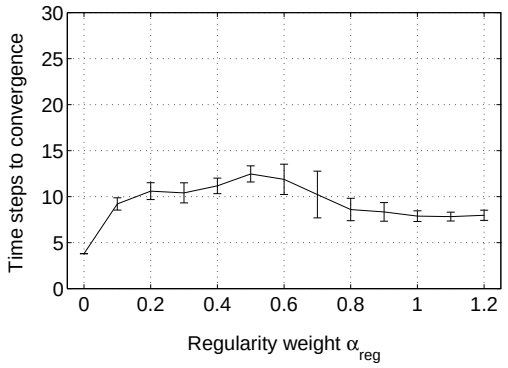
In this experiment, segmentation performance is tested with different settings for the occlusion detection threshold τ_{occ} (see Sect. 3.2.1). The regularity weight α_{reg} has been set to the value with the best performance in the last experiment, $\alpha_{\text{reg}} = 0.5$. The results of this experiment are presented and discussed in Fig. 4.4.

4.2.3 Color Cue

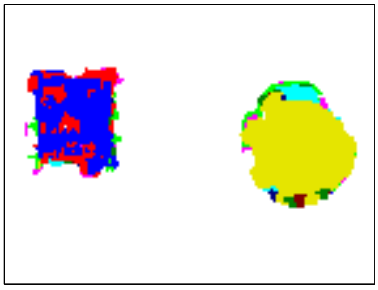
In this experiment the influence of the color cue on segmentation is analyzed. To demonstrate the ability of the color cue to compensate for the failure of the motion cue, the system is tested using images of a second sequence, where the textures of the moving objects in the scene have been removed. The result of this experiment is shown and discussed in Fig. 4.5.



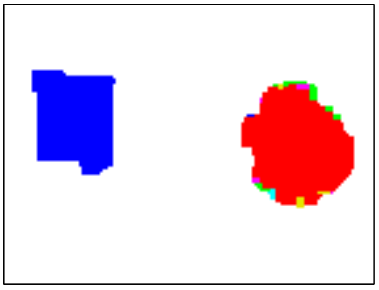
(a) Quality of segmentation



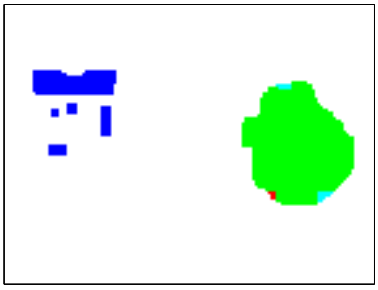
(b) Convergence time



(c) $\alpha_{\text{reg}} = 0.0$



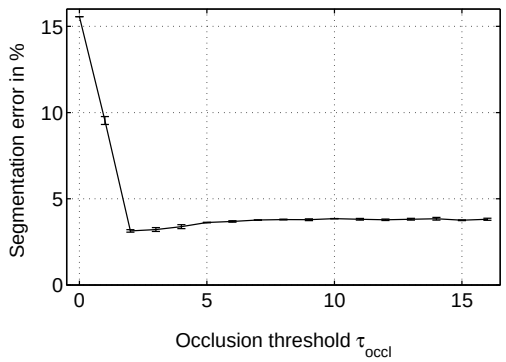
(d) $\alpha_{\text{reg}} = 0.4$



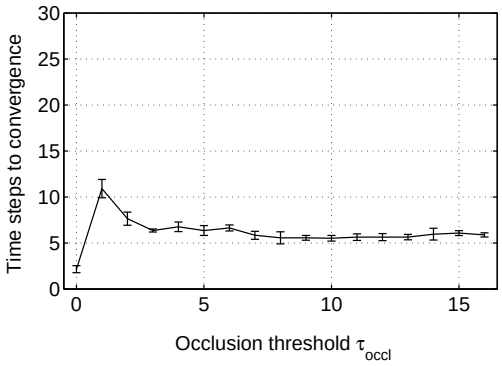
(e) $\alpha_{\text{reg}} = 0.8$

Figure 4.3: Segmentation performance depending on the amount of regularization controlled by the weight α_{reg}. Values are averaged over 10 trials with 10 frames each. The error bars show the standard deviation over the trials.

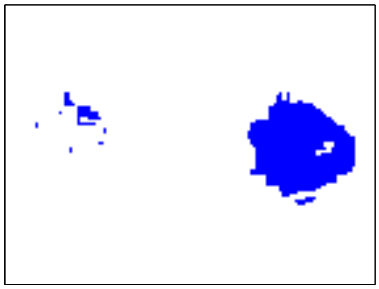
The segmentation error (a) decreases as regularization becomes stronger up to the point where $\alpha_{reg} = 0.5$. For higher weights the error steps up again because "over-smoothing". The example segmentations document the system's behaviour: without regularization (c), no smoothness is enforced on the labeling. The label of every site only depends on the motion data. If the motion estimate is ambigous, arbitrary labels are chosen. As a result, the segmentation looks "grainy" in the lower contrasted object on the left. With $\alpha_{reg} = 0.4$, the segments are smooth (d). If the regularization is stronger than the fidelity, the labeling spreads across object boundaries (e). With regularization the convergence time increases (b), since label changes based on smoothing propagate through the field with one site per iteration. For the following experimens, $\alpha_{reg} = 0.5$ is used.



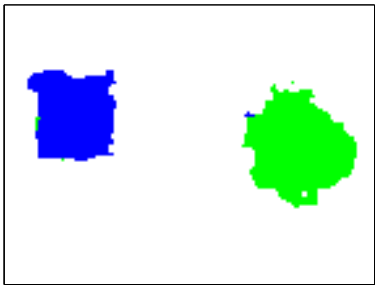
(a) Quality of segmentation



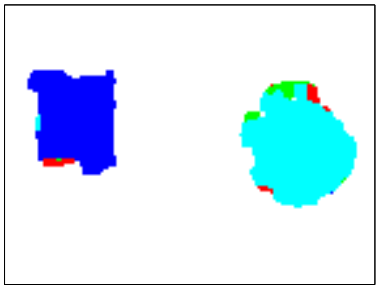
(b) Convergence time



(c) $\tau_{\text{occ}} = 1$

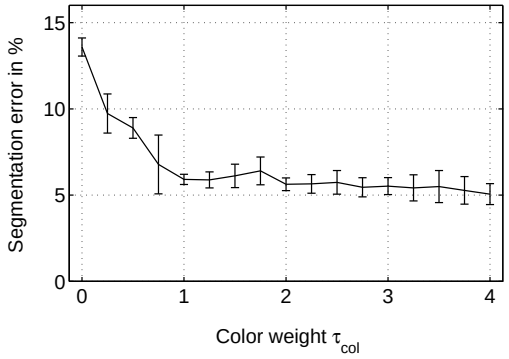


(d) $\tau_{\text{occ}} = 3$

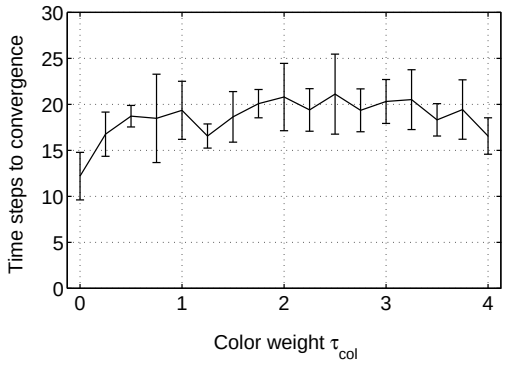


(e) $\tau_{\text{occ}} = 16$

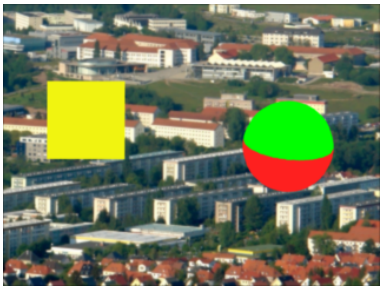
Figure 4.4: Segmentation performance depending on occlusion handling, controlled by the occlusion threshold τ_{occ}: if the minimum value in a SSD surfaces is greater than τ_{occ}, the SSD surface is excluded from the cost function. The graph (a) shows the segmentation error depending on the parameter choice for τ_{occ}. Values are averaged over 10 trials with 10 frames each. The error bars show the standard deviation over the trials.
If the threshold τ_{occ} is too high to be reached (c), the occlusion handling does not work. With adequate settings $3 \leq \tau_{occ} \leq 5$, the false segments at the front border of the moving objects are supressed. If τ_{occ} is too low, good motion estimates are discarded and the segmentation deteriorates. For the next experiments, $\tau_{occ} = 4$ is used.



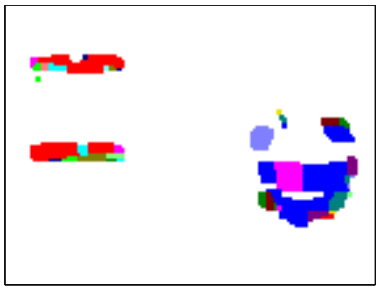
(a) Quality of segmentation



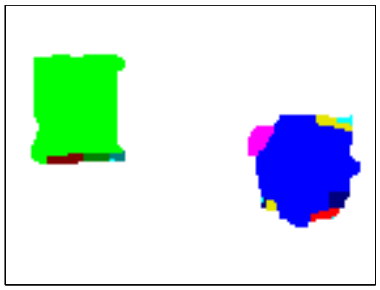
(b) Convergence time



(c) Seq2 - Frame 6



(d) $\alpha_{\text{col}} = 0$

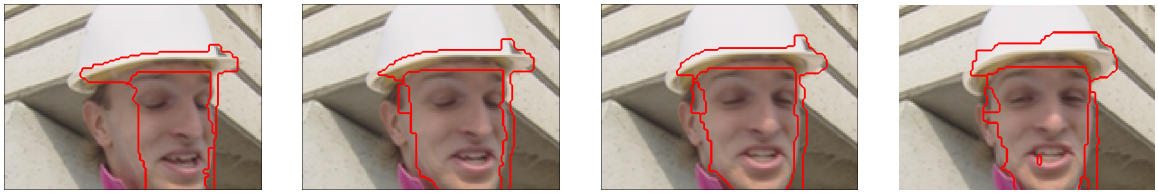


(e) $\alpha_{\text{col}} = 2$

Figure 4.5: Segmentation performance depending on integration of a color cue, controlled by the weight parameter α_{col}. This experiment was conducted on image sequence "seq2", which is identical to "seq1" except that object textures have been removed from this one (c). Without the color cue, i.e. $\alpha_{col} = 0$, the segmentation error is significantly higher compared to the segmentation error with object textures (see Fig. 4.3). This is due to the falsely labeled parts in the homogeneous interior area of the object surfaces (d), where no motion can be estimated. If the weight of the color cue is raised to $\alpha_{col} \geq 1$, the equality of color of the interior homogeneous area and the contrast regions at the boundary allows correct segmentation (e). Note that the labeling of the right object, which has two colors, is not split at the color boundary. This is possible due to the color mixture model in the form of histograms. The use of the color cue increases the convergence time. This can be explained with the higher number of prototype parameters that need to be estimated along with the configuration.

4.3 Real Image Sequences

Real image sequences are used to further evaluate the performance of the system, and to reveal problems that did not occur in the segmentation of clean rendered images. Results for standard sequences are shown and discussed in Fig. 4.6 and Fig. 4.7, and for a sequence of self-recorded camera images in Fig. 4.8. The experiment of Fig. 4.9 demonstrates how the depth cue can compensate for the failure of the motion cue. The weight parameters have been set to the values determined before. The segmentations are displayed as contourplots overlayed over the actual images. In these plots the red contours resemble the segment boundaries of the labeling.



(a) Frame 77 (b) Frame 78 (c) Frame 79 (d) Frame 80

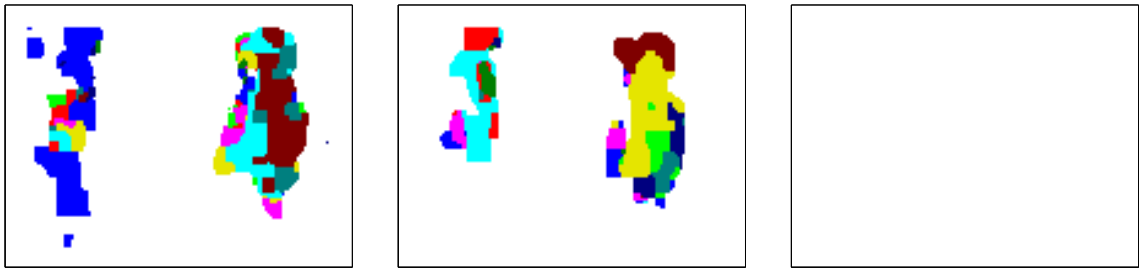
Figure 4.6: Segmentation of the "foreman" sequence from the MPEG benchmark set. Here the 3D rotation of the head is approximated with homogeneous 2D translation of the regions. In the projection, the edge of the ear and the helmet is not moving. Thus the motion in these parts does not match the translational motion model.

(a) Frame 7 (b) Frame 8 (c) Frame 9 (d) Frame 10

Figure 4.7: Segmentation of the sequence "ttenis" from the MPEG benchmark set. In each frame a false segment appears in the location that will be covered by the moving ball in the next frame. The background has low contrast in the vertical direction, and a downwards displacement of the segment does not yield high matching errors in the SSD surfaces. Thus this area is not detected as an occlusion area, the ball "pushes" the segment downwards.

(a) Frame 107 (b) Frame 108 (c) Frame 109 (d) Frame 110

Figure 4.8: Segmentation of camera images from the sequence "walk3". The system failed to associate each of the moving persons with one region. Instead, different parts of the bodies have been assigned with different labels. Even with a more complex motion model, this problem would still remain: the arms move in different directions than the bodies and clothes move in an unrigid way. In the top left corner, the movement of the person in the mirror has been detected.



(a) Frame 109, $\alpha_{\text{dep}} = 2$ (b) Frame 109', $\alpha_{\text{dep}} = 2$ (c) Frame 109', $\alpha_{\text{dep}} = 0$

Figure 4.9: Influence of the depth cue. In this experiment a duplicate of frame 109 of the "walk3" sequence has been used to replace frame 110. Thus, absolutely no motion occurs from frame 109 to 109'. The system was started in frame 107. If the depth cue is used, disparity information is stored in the prototype data and integrated into segmentation. This is a basic kind of object knowledge aquired from the frames where motion has been present (a). If the motion stops, a pure disparity-based segmentation remains (b). If the depth cue is not used (c), segmentation fails completely if motion disappears.

Chapter 5

Discussion

This chapter discusses various aspects of the proposed system and suggests possible extensions. Parallels and analogies between this technical system and biological vision systems are outlined, and a conclusion summarizes the contributions of this work.

5.1 Towards Real Time Application

The motion segmentation systems reviewed in this work are in general not yet suitable for real time application. Computation times vary between several seconds and several hours per frame, depending on the image resolution.

The system developed in this work is currently implemented with MATLAB. For the computationally intensive parts, MEX functions have been generated from C functions. On 160x120 images, the motion estimation for one frame takes around 5 seconds, and the optimization procedure needs 0.5–8 seconds, depending on the number of integrated cues and the number of labels. Another 1–2 seconds are spent on loading images, preparing the data, initialization, applying the motion model and other minor tasks, yiedling a total time of 6.5–15 seconds per frame.

To speed up the motion estimation process, resolution pyramids could be used with a coarse-to-fine strategy as previously done by [Anandan, 1989]. This would allow to reduce the number of unsuccessful matches attempted during motion estimation. Hardware solutions are appealing as well, the computation of the SSD surfaces can

be parallelized and distributed across multiple processors. The current optimization scheme visits all sites in each iteration. A more selective strategy can be chosen to reduce computational effort, as mentioned in Sect. 2.4.4.
Currently, motion is estimated by estracting a patch from an image at time step t and searching for the corresponding patch in the following image from time step $t+1$. Thus the motion estimate and the segmentation corresponds to the image at time step t and not to the most recent one. For real time applications, it might be desirable to search backwards in time to obtain a segmentation for time step $t+1$ instead.

5.2 Moving Camera and Complex Motion Models

The motion segmentation system presented and tested here was designed with the assumption of the camera to be stationary. The label $l = 0$ is interpreted as background label with $\Delta x = \Delta y = 0$. If ego-motion of the camera is present, this is not applicable anymore.
Rotation of the camera around its vertical axis in the nodal point merely induces a homogeneous translation of the background in the image. In this case any label other than $l = 0$ can take over the role of the background label, or one could abandon the constraint of having a zero-motion background label. If the camera translates horizontally, the direction of all background motion in the image is the same, while the velocity depends on the distance of objects from the camera. In all other cases, e.g. the camera moving forwards, the direction and the velocity of background motion in the image depend on both the location in the image and the distance.
To correctly represent this non-homogeneous background motion with only one label, the underlying motion model of the prototype parameters needs to be enhanced. As a first step, a new special background label with a 3D camera motion model could be introduced, while keeping the general motion model unchanged. To estimate camera motion from local motion estimates in the presence of moving objects without knowledge of a segmentation, an approach is needed that can deal with a high number of outliers caused by object motion. Robust estimators with high break-down points [Sim and Park, 1998] or the RANSAC method [Fischler and Bolles, 1981] can be used for this

purpose. Kalman filtering [Kalman, 1960] or condensation [Isard and Blake, 1998] can be applied to smooth the camera motion estimate, and to integrate odometry data if available.

The general use of a more complex motion model is a possible extension to the present system as well. Full 2D affine motion parameters will help to compute good segmentations if large rotations or distance changes occur. The use of full motion space surfaces in the fidelity term would probably have to be replaced with fitting of a motion model, since the time needed to compute the supports of the surfaces grows exponentially with the dimensionality of the motion model. Estimating the labels' motion parameters can still be done with a least squares estimate.

5.3 Neural Analogies

The system presented in this work was not explicitly designed to be biologically plausible or to model biological processes. Nevertheless, parts of the system and its ways of computation are similar to structures found in human or mammal brains.

One example is the integration of edges into the segmentation process. In technical applications, the fusion of motion and object boundary information has been a strong focus of research [Moscheni et al., 1998].From a general segmentation perspective, edges can help to determine a correct filling-in of regions. This has also been investigated in the neuro sciences: [Cohen and Grossberg, 1984] proposed a perceptual model for the primary visual cortex that realizes the filling-in of surface areas by spreading of neural activity. This diffusion process stops at boundaries marked by brightness edges. The system developed in this work shows a similar behaviour. Distinctly assigned labels spread to sites with ambiguous motion estimates because of the regularization term. At brightness edges the smoothing is reduced or disabled, thus the filling-in stops at these locations.

5.3.1 Motion Detection and Estimation

Sensitivity to motion can be found in different parts of the vision system of mammals, namely the retina, the Lateral Geniculate Nucleus (LGN), and the visual cortex. The motion selectivity in the retina and LGN does not provide detailed spatial resolution, and is assumed to be responsible for triggering eye motion, e.g. the vestibo-ocular reflex. In the primary visual cortex V1 (striate cortex), directional and spatial selective neurons obtain their selectivity independently from the motion detection systems in the retina and LGN [Squire et al., 2002].

Simple cells in the V1 show excitatory *on* and inhibitory *off* responses for stimuli in different areas of their receptive fields. Their receptive fields are not rotationally symmetric with an on-center and off-periphery or vice versa, but are tuned to only respond to brightness edges with a certain orientation [Hubel, 1995].

Complex cells are also selective to orientation, but mainly respond to moving brightness edges. They are assumed be excited by simple cells with the same orientation selectivity and slightly displaced receptive fields. Different theories explaining the selectivity of complex cells for motion have been proposed. One of them employs intermediate cells as delay element, another one supposes delay phenomena caused by varying positions of simple cells' synapses on the dendrites of complex cells. Both theories assume that a complex cell responds to a moving stimuli, if the phase differences of simple cell responses correspond to the delay times [Hubel, 1995], such that multiple responses fall together temporally.

An SSD surface as used for motion estimation in this work (see Fig. 2.3b) can be seen as the equivalent of responses of a population of complex cells, tuned to different directions and velocities. If correlation is used as similarity measure, the values of a similarity surface correspond to the correlation of a pattern appearing at slightly different positions in two time steps, which is considered to be stimulating the response of complex cells as well.

5.3.2 Labeling and Optimization

A labeling of sites assigns each site a label from a set of labels, and all sites with the same label are implicitly grouped together to a region. Classical concepts of firing rates of neurons in topology preserving maps are not sufficient to encode a multi-class labeling, since different magnitudes of activation typically represent one fitness value, e.g. the strength of affiliation with one label, rather than a 1-in-n coding for multiple labels.

Simultaneous recordings of individual cells in mammal brains provided insights in the phenomena of neural synchrony [Castelo-Branco et al., 2000]. In the context of perceptual grouping it has been found that besides outputting electrical spikes at different frequencies (firing rates), neurons can synchronize their spike patterns. The effect of neural synchrony appears between spatially close neurons as well as between neurons in different areas of the brain, and seems to occur if neurons encode the same perceptual event or object. This suggests that the actual spike patterns encode additional grouping information. To what extend this actually happens is subject of a scientific debate and still part of ongoing research. However, the interpretation of different spike patterns as labels and neural synchrony for grouping of neurons creates a plausible analogy between labeling representations and neural effects found in mammal brains.

The segmentation in the system presented in this work is achieved by optimization on a Markov Random Field. The optimization of the configuration is a local process, the sites are laterally connected for smoothing and the fidelity of each site only depends on local image properties. This architecture is similar to Neural Field Dynamics [Amari, 1977], but again the implementation of multi-class labeling requires additional representation concepts.

5.4 Conclusion

This work presented a systematical review of existing approaches to motion segmentation by analyzing proposed systems using characteristic properties. As main criteria,

the method of motion estimation, represention of regions, the kind of the actual segmentation algorithm and the approach to include multi-cue information have been selected.

A motion segmentation system framework has been proposed that is able to integrate multiple cues in a consistent way. A general classification of cues into edge and area cues has been presented. Cues of these types can be integrated in the specified way. In general, cue fusion and segmentation is combined into one optimization process. This way, individual cues can compensate for the failure of others and contribute to the general robustness of the system. In the current implementation of the system, motion, color, depth and brightness edge information has been used. The advantage of integrating the individual cues has been demonstrated in experiments.

Motion information is included in the form of SSD matching surfaces, rather than optical flow that only provides the best matches. This way, data from evenly good matches in ambiguous cases is not discarded. The same SSD surfaces provide useful information for the initialization of the optimization process. The importance of occlusion handling in the context of motion estimation has been shown.

The choice of a similarity meausure has been based on experimental comparison of different metrics under varying amounts and types of noise. This experiments revealed that SSD outperforms SAD in matching performance, and the computation of SSD is faster than SAD on a modern CPU. It also showed that cross-correlation-based measures function only when normalization is applied.

Color prototype information in the system is represented by 2D hue-saturation histograms that resemble color mixture models. This way, regions are not split erroneously at intra-object color boundaries, as demonstrated in an experiment.

Despite the flexibility of multi-cue integration and the good segmentation results, the current system is limited in some ways. Motion is represented as local translational motion. The use of 2D affine or 3D motion models would improve the performance in cases where large rotations, changes in distance or non-rigid body motion take place. The general approach with its special characteristics however is suited to employ other motion models.

Appendix A

Comparison of Similarity Measures

Various metrics have been defined to express the similarity or the difference of image patches. Especially for cross-correlation based measures several definitions can be found in the literature, with differences in the details of normalization. SSD measures exist in different normalized versions as well. This section compares the matching performance and computational requirements of different similarity measures under noisy conditions, i.e. in the presence of Gaussian sensor noise and global brightness offsets due to illumination changes.

A.1 Definition of Measures

The similarity measures compared here are split into two different groups. The first group consists of measures that are based on the sum of absolute differences (SAD) or the sum of squared differences (SSD). The best match according to these measures is the one with the minimum value. The second group is based on cross-correlation (CORR), here the best match is the one with the maximum value.
To normalize a measure, the sum of the brightness values are computed for the compared image patches. The normalized derivate (Nx) of a measure $x \in \{\text{SSD, CORR}\}$ divides x by the square-root of the product of both patch-sums. The square-normalized versions (N2x) use the squares of the patch-sums. The square-normalized zero-mean versions (N2Zx) compute the (N2x) on mean-free image patches

SAD, SSD and CORR have already been defined in (2.1-2.3). In the following, the different forms of normalization are given for CORR. The derivates of SSD are computed respectively. Constant normalization factors like $\frac{1}{N^2}$ are omitted, they do not influence the optmization. All sums in these equations are computed over the correlation window as in (2.3):

$$CORR(x, y, \Delta x, \Delta y) := \sum \Big(I(i, j) \cdot I'(i + \Delta x, j + \Delta y) \Big) \tag{A.1}$$

$$NCORR(x, y, \Delta x, \Delta y) := \frac{\sum \Big(I(i, j) \cdot I'(i + \Delta x, j + \Delta y) \Big)}{\sqrt{\sum I(i, j) \cdot \sum I'(i + \Delta x, j + \Delta y)}} \tag{A.2}$$

$$N2CORR(x, y, \Delta x, \Delta y) := \frac{\sum \Big(I(i, j) \cdot I'(i + \Delta x, j + \Delta y) \Big)}{\sqrt{\sum I(i, j)^2 \cdot \sum I'(i + \Delta x, j + \Delta y)^2}} \tag{A.3}$$

$$N2ZCORR(x, y, \Delta x, \Delta y) := \frac{\sum \Big(\tilde{I}(i, j) \cdot \tilde{I}'(i + \Delta x, j + \Delta y) \Big)}{\sqrt{\sum \tilde{I}(i, j)^2 \cdot \sum \tilde{I}'(i + \Delta x, j + \Delta y)^2}} \tag{A.4}$$

where $\tilde{I}$ is a zero-mean image patch

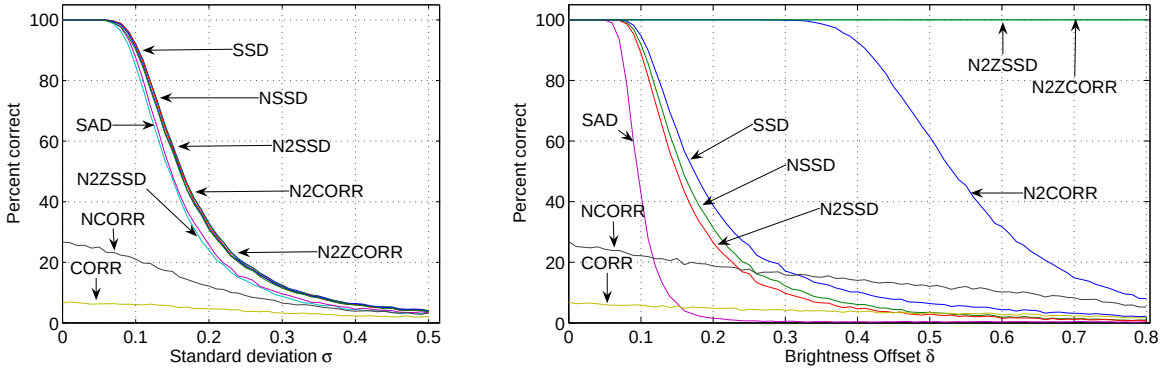
$$\tilde{I}(i, j) = I(i, j) - \sum_{i,j} I(i, j) \tag{A.5}$$

and $\tilde{I}'$ respectively.

A.2 Matching Performance

The matching performance of the similarity measures defined in Sect. A.1 is evaluated by experimental comparison in several trials. In this experiments, similarity surface are computed, as described in Sect. 2.1.2. In each trial t an independent and identically distributed (i.i.d.) random image $I_t(x, y) \sim N(\mu, \sigma_I)$ of size $M \times M$ ($M = 15$) is generated, with mean $\mu = 0.5$, standard deviation $\sigma_I = 0.1$ and $x, y \in \{-8, \ldots, 8\}$. From this image, a patch of size $N \times N$ with $N = 5$ is extracted from the center and combined with additive noise of different types:

$$I'_t(x, y) := I_t(x, y) + n_t(x, y), \;\; \forall x, y \in \{-2, \cdots + 2\} \;\; . \tag{A.6}$$



(a) Gaussian noise

(b) Constant offset

Figure A.1: Performance of similarity measures under noisy conditions, indicated by the percentage of correct distinct best matches. In (a) additive Gaussian noise is used to model sensor noise. In (b) a constant offset is added to model illumination changes. N2ZCORR performs best under all conditions, followed by N2ZSSD and N2CORR. NSSD and N2SSD have no advantage over SSD, which yields better results than SAD. CORR and NCORR perform unacceptable even without noise.

This image patch is compared to the original image using each of the similarity measures. The performance of a similarity measure is rated by the percentage of trials where the correct match (i.e. $\Delta x = \Delta y = 0$) is the distinct best match.

The same experiments have been done with equally distributed noise images as well. The results are similar to the results presented here, both in quality and quantity, if the contrast $c = \max\big(I(x,y)\big) - \min\big(I(x,y)\big)$ of the brightness values is chosen to be $c = 3\sigma_I = 0.3$.

To model sensor noise, the term $n_t(x,y)$ is chosen to resemble i.i.d. Gaussian noise $N(0,\sigma)$. Figure A.1a shows results for different standard deviations σ. Each data point is the average hit rate over 10000 trials. CORR and NCORR show unacceptable performance with hitrates below 10% and 30% respectively, even without noise at $\sigma = 0$. The performance of the other measures is very good up to $\sigma = 0.1$, which is equal to the standard deviation σ_I of the Gaussian noise generating the original image, or 1/3 of the contrast if equally distributed noise is used to generate the images. SAD and N2ZSSD perform slightly worse than the rest. The hit rates of all measures is affected by the additive Gaussian noise.

Global illumination changes in this test are modeled with a constant brightness offset $n_t(x, y) := \delta$. Figure A.1b shows results for different levels of this offset. Each data point is the average hit rate over 10000 trials. Again, CORR and NCORR show unacceptable performance. The performance of the measures N2ZSSD and N2ZCORR that operate on mean-free image patches is not affected by the global brightness offset. SSD, NSSD and N2SSD perform similarly good up to the offset $\delta = 0.1$ which is equal to the standard deviation σ_I for Gaussian noise images or 1/3 of the contrast for equally distributed noise images. Their performance decreases for greater offsets, while the hit rate of N2CORR is longer at 100%. The performance of SAD decreases earlier and faster than the SSD hit rate.

A.3 Computation Time

In this section, the computational performance of the similarity measures defined in Sect. A.1 is compared. All of them have been implemented in seperate MEX functions (C-Code), and executed 10^6 times on random image patches of the same size as in the previous tests for all possible displacements. A dummy measure without any actual computation has been implemented to measure the overhead time spent in the function for parameter fetching and initalization of variables. If the code of a metric is directly integrated in a C program, this time will decrease.

The MATLAB profiler has been used to measure the computation time, the test has been run on a IBM Thinkpad with a Pentium M Processor with 1.4 GHz.

The results for this experiment are shown in Fig. A.2. The computation time of a similarity measure clearly depends on the type of normalization that comes to use. The unnormalized measures SAD, SSD and CORR are the fastest, the normalized and square-normalized measures NSSD, N2SSD, NCORR and N2CORR need 2.5 times as long, and the measures operating on mean-free image patches N2ZSSD and N2ZCORR take 3.5 times longer.

Some authors prefer to use the sum of absolute difference (SAD) over the sum of squared difference (SSD) measure for performance reasons: the operations to compute the absolute value in the SAD equation are assumed to be computationally cheaper than

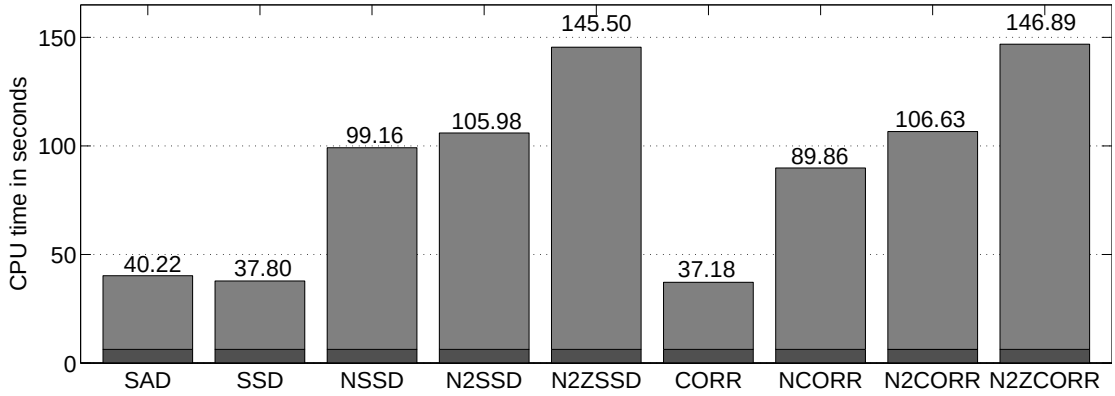


Figure A.2: Computation time of similarity measures for 10^6 iterations. The dark parts of the bars depict the functions' approximate overhead time of 6.29 s as measured with a dummy metric. This time is included in the numeric results. The normalized measures NSSD, N2SSD, NCORR and N2CORR need around 2.5 times the CPU time of the unnormalized measures SAD, SSD and CORR, and the mean-free measures N2ZSSD and N2ZCORR need around 3.5 times as long. SSD is slightly faster than SAD.

the multiplication needed for the SSD. This might be true for older CPUs and smaller processors used in embedded systems, that cannot execute floating point multiplications in one clock cycle. The results in Fig. A.2 show that SSD is faster than SAD on a modern Pentium CPU. Although the unnormalized CORR is even faster, it is not suitable for use due to bad matching performance.

A.4 Conclusion

Nine similarity measures have been tested on their performance under additive Gaussian noise and additive brightness offsets. The CPU time requirements have been anaylized as well.

Unnormalized correlation (CORR) and the normalized correlation (NCORR) are not usable for correspondence matching, since their ability to detect the correct best match is under 30% even without noise. The sum of squared differences (SSD) metric performs better than the sum of absolute differences (SAD), plus it takes slightly less time to compute on modern CPUs.

The metrics operating on mean-free image patches N2ZSSD and N2ZCORR are not affected by constant brightness offsets. N2ZCORR performs slightly better than N2ZSSD in the presence of Gaussian noise. With the invariance to constant offsets, these measures are assumed to be more robust to abrupt illumination changes. This advantage has to be paid for with significantly higher computation cost.
The square-normalized zero-mean cross-correlation N2ZCORR (also called covariance measure) is appealing to be used for the proposed system because of the good performance under noisy conditions. The sum of squared differences (SSD) performs equally good under Gaussian noise and can deal with a considerable brightness offset as well. Also, its requirements of computation time are significantly lower. Therefore SSD has been selected for the system developed in this work.

Bibliography

[Altunbasak et al., 1998] Altunbasak, Y., Eren, P. E., and Tekalp, A. M. (1998). Region-based parametric motion segmentation using color information. *Graphical Models and Image Processing*, 60(1):13–23.

[Amari, 1977] Amari, S. (1977). Dynamics of pattern formation in lateral-inhibition type neural fields. *Biol. Cybernetics*, 27:77–87.

[Anandan, 1989] Anandan, P. (1989). A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2(3):283–310.

[Åström and Kahl, 1999] Åström, K. and Kahl, F. (1999). Motion estimation in image sequences using the deformation of apparent contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(2):114–127.

[Ayer and Sawhney, 1995] Ayer, S. and Sawhney, H. (1995). Layered representation of motion video using robust maximum-likelihood estimation of mixture models and mdl encoding. In *Proc. of the Fifth International Conference on Computer Vision*.

[Barron et al., 1994] Barron, J. L., Fleet, D. J., and Beauchemin, S. S. (1994). Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77.

[Bergen and Meyer, 2000] Bergen, L. and Meyer, F. (2000). A novel approach to depth ordering in monocular image sequences. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 536–541.

[Besag, 1986] Besag, J. (1986). On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society. Series B (Methodological)*, 48(3):259–302.

[Black, 1992] Black, M. J. (1992). Combining intensity and motion for incremental segmentation and tracking over long image sequences. In *Proceedings of the Second European Conference on Computer Vision*, volume 588 of *Lecture Notes In Computer Science*, pages 485–493. Springer.

[Black and Anandan, 1996] Black, M. J. and Anandan, P. (1996). The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1):75–104.

[Blake and Isard, 1998] Blake, A. and Isard, M. (1998). *Active Contours: The Application of Techniques from Graphics,Vision,Control Theory and Statistics to Visual Tracking of Shapes in Motion*. Springer-Verlag New York.

[Bober et al., 1998] Bober, M., Petrou, M., and Kittler, J. (1998). Nonlinear motion estimation using the supercoupling approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):550–555.

[Bouthemy and Francois, 1993] Bouthemy, P. and Francois, E. (1993). Motion segmentation and qualitative dynamic scene analysis from an image sequence. *Intl. Journal of Computer Vision*, 10(2):157–182.

[Brown et al., 2003] Brown, M., Burschka, D., and Hager, G. (2003). Advances in computational stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8):993 – 1008.

[Canny, 1986] Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698.

[Castelo-Branco et al., 2000] Castelo-Branco, M., Goebel, R., Neuenschwander, S., and Singer, W. (2000). Neural synchrony correlates with surface segregation rules. *Nature*, 405:685–689.

[Chang et al., 1997] Chang, M., Tekalp, A., and Sezan, M. (1997). Simultaneous motion estimation and segmentation. *IEEE Transactions on Image Processing*, 6(9):1326–1333.

[Choi and Kim, 1996] Choi, J. and Kim, S. (1996). Multi-stage segmentation of optical flow field. *Signal Processing*, 54(2):109–118.

[Cohen and Grossberg, 1984] Cohen, M. and Grossberg, S. (1984). Neural dynamics of brightness perception: features, boundaries, diffusion, and resonance. *Percept Psychophys.*, 6(5):428–456.

[Cremers and Soatto, 2005] Cremers, D. and Soatto, S. (2005). Motion competition: A variational framework for piecewise parametric motion segmentation. *International Journal of Computer Vision*, 62(3):249–265.

[Cutler and Davis, 2000] Cutler, R. and Davis, L. (2000). Robust real-time periodic motion detection, analysis, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):781–796.

[Debrunner and Ahuja, 1998] Debrunner, C. and Ahuja, N. (1998). Segmentation and factorization-based motion and structure estimation for long image sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(2):206–211.

[Dempster et al., 1977] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B*, 39:1–38.

[Depommier and Dubois, 1992] Depommier, R. and Dubois, E. (1992). Motion estimation with detection of occlusion areas. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 3, pages 269–272.

[Dubuisson and Jain, 1995] Dubuisson, M. and Jain, A. (1995). Contour extraction of moving objects in complex outdoor scenes. *International Journal of Computer Vision*, 14(1):83–105.

[Fischler and Bolles, 1981] Fischler, M. and Bolles, R. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. of the ACM*, 24:381–395.

[Galvin et al., 1998] Galvin, B., McCane, B., Novins, K., Mason, D., and Mills, S. (1998). Recovering motion fields: An evaluation of eight optical flow algorithms. In *Proceedings of the Ninth British Machine Vision Conference (BMVC)*, volume 1, pages 195–204, Southampton, UK.

[Geman and Geman, 1984] Geman, S. and Geman, D. (1984). Stochastic relaxation, gibbs distributions and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):564–584.

[Grossberg, 2000] Grossberg, S. (2000). Filling-in the forms: Surface and boundary interactions in visual cortex. Technical Report CAS/CNS-2000-018, Boston University Center for Adaptive Systems and Department of Cognitive and Neural Systems.

[Gu et al., 1996] Gu, H., Shirai, Y., and Asada, M. (1996). MDL-based segmentation and motion modeling in a long image sequence of scene with multiple independently moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(1):58–64.

[Heitz et al., 1991] Heitz, F., Perez, P., and Bouthemy, P. (1991). Parallel visual motion analysis using multiscale markov randomfields. In *Proceedings of the IEEE Workshop on Visual Motion*.

[Horn and Schunck, 1981] Horn, B. K. and Schunck, B. G. (1981). Determining optical flow. *Artificial Intelligence*, 17:185–203.

[Hu et al., 2004] Hu, W., Tan, T., Wang, L., and Maybank, S. (2004). A survey on visual surveillance of object motion and behaviors. *IEEE Transactions on systems, man and cybernetics. Part C : Applications and reviews*, 34(3):334–352.

[Huang et al., 1995] Huang, Y., Palaniappan, K., Zhuang, X., and Cavanaugh, J. (1995). Optic flow field segmentation and motion estimation using a robust ge-

netic partitioning algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(12):1177–1190.

[Hubel, 1995] Hubel, D. H. (1995). *Eye, Brain, and Vision*, chapter 4. Scientific American Library.

[Isard and Blake, 1998] Isard, M. and Blake, A. (1998). Condensation - conditional propagation for visual tracking. *Internation Journal on Computer Vision*, 29(1):5–28.

[Jepson and Black, 1993] Jepson, A. and Black, M. (1993). Mixture models for optical flow computation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 760–761, New York, NY, USA.

[Kalman, 1960] Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45.

[Kanatani and Matsunaga, 2002] Kanatani, K. and Matsunaga, C. (2002). Estimating the number of independent motions for multibody motion segmentation. In *Proceedings of the 5th Asian Conference on Computer Vision (ACCV)*.

[Kass et al., 1988] Kass, M., Witkin, A., and Terzopoulos, D. (1988). Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331.

[Khan and Shah, 2001] Khan, S. and Shah, M. (2001). Object based segmentation of video using color, motion and spatial information. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages II–746 – II–751.

[Kolmogorov et al., 2006] Kolmogorov, V., Criminisi, A., Blake, A., Cross, G., and Rother, C. (2006). Probabilistic fusion of stereo with color and contrast for bilayer segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1480–1492.

[Kolmogorov and Zabih, 2001] Kolmogorov, V. and Zabih, R. (2001). Computing visual correspondence with occlusions using graph cuts. In *Eighth International Conference on Computer Vision (ICCV)*, volume 2.

[Konrad and Dubois, 1990] Konrad, J. and Dubois, E. (1990). Comparison of stochastic and deterministic solution methods in bayesian estimation of 2D motion. *Image and Vision Computing*, 8(4):304–317.

[Lai and Vemuri, 1998] Lai, S.-H. and Vemuri, B. C. (1998). Reliable and efficient computation of optical flow. *International Journal of Computer Vision*, 29(2):87–105.

[Li, 2001] Li, S. Z. (2001). *Markov random field modeling in image analysis*. Springer-Verlag, 2nd edition edition.

[Lim et al., 2002] Lim, K. P., Das, A., and Chong, M. N. (2002). Estimation of occlusion and dense motion fields in a bidirectional bayesian framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):712 – 718.

[Lowe, 2004] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.

[MacQueen, 1967] MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, Berkeley. University of California Press.

[Mansouri and Konrad, 2003] Mansouri, A.-R. and Konrad, J. (2003). Multiple motion segmentation with level sets. *IEEE Transactions on Image Processing*, 12(2):201–220.

[Meygret and Thonnat, 1990] Meygret, A. and Thonnat, M. (1990). Segmentation of optical flow and 3D data for the interpretation ofmobile objects. In *In Proceedings of the 3. Intl. Conference on Computer Vision*.

[Mitiche and Sekkati, 2006] Mitiche, A. and Sekkati, H. (2006). Optical flow 3D segmentation and interpretation: A variational method with active curve evolution

and level sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1818–1829.

[Moscheni et al., 1998] Moscheni, F., Bhattacharjee, S., and Kunt, M. (1998). Spatio-temporal segmentation based on region merging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(9):897–915.

[Murray and Buxton, 1987] Murray, D. and Buxton, B. (1987). Scene segmentation from visual motion using global optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(2):220–228.

[Noble and Boukerroui, 2006] Noble, J. and Boukerroui, D. (2006). Ultrasound image segmentation: a survey. *IEEE Transactions on Medical Imaging*, 25(8):987–1010.

[Ogale et al., 2005] Ogale, A., Fermuller, C., and Aloimonos, Y. (2005). Motion segmentation using occlusions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6):988–992.

[Ong and Spann, 1999] Ong, E. and Spann, M. (1999). Robust optical flow computation based on least-median-of-squares regression. *International Journal of Computer Vision*, 31(1):51–82.

[Onoe et al., 1973] Onoe, M., Hamano, N., and Ohba, K. (1973). Computer analysis of traffic flow observed by subtractive television. *Journal of Computer Graphics and Image Processing*, 2:377–392.

[Pal and Pal, 1993] Pal, N. R. and Pal, S. K. (1993). A review on image segmentation techniques. *Pattern Recognition*, 26(9):1277–1294.

[Pless et al., 2000] Pless, R., Brodsky, T., and Aloimonos, Y. (2000). Detecting independent motion: the statistics of temporal continuity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):768–773.

[Potter, 1977] Potter, J. (1977). Scene segmentation using motion information. *Computer Graphics and Image Processing*, 6:558–6581.

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

[Sim and Park, 1998] Sim, D.-G. and Park, R.-H. (1998). Robust reweighted map motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(4):353–365.

[Smith et al., 2004] Smith, P., Drummond, T., and Cipolla, R. (2004). Layered motion segmentation and depth ordering by tracking edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(4):479–494.

[Smith and Brady, 1995] Smith, S. and Brady, J. (1995). Asset-2: real-time motion segmentation and shape tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):814–820.

[Snoek and Worring, 2005] Snoek, C. G. and Worring, M. (2005). Multimodal video indexing: A review of the state-of-the-art. *Multimedia Tools and Applications*, 25(1):5–35.

[Squire et al., 2002] Squire, Bloom, McConnell, Roberts, Spitzer, and Zigmond (2002). *Fundemental Neuroscience*, chapter 27. Academic Press.

[Stephan, 2001] Stephan, V. (2001). *Visuomotorische Antizipation: eine handlungsorientierte Sicht auf die visuelle Wahrnehmung*. PhD thesis, TU Ilmenau.

[Stiller, 1997] Stiller, C. (1997). Object-based estimation of dense motion fields. *IEEE Transactions on Image Processing*, 6(2):234–250.

[Szeliski and Shum, 1996] Szeliski, R. and Shum, H.-Y. (1996). Motion estimation with quadtree splines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(12):1199–1210.

[Thompson, 1980] Thompson, W. (1980). Combining motion and contrast for segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(6):543–549.

[Torres et al., 1996] Torres, L., Kunt, M., and Pereira, F. (1996). Second generation video coding schemes and their role in MPEG-4. In *European Conf. on Multimedia Appliactions, Services, and Techniques*, pages 799–824.

[Unnikrishnan et al., 2005] Unnikrishnan, R., Pantofaru, C., and Hebert, M. (2005). A measure for objective evaluation of image segmentation algorithms. In *Proceedings of the 2005 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '05), Workshop on Empirical Evaluation Methods in Computer Vision.*

[Vasconcelos and Lippman, 1997] Vasconcelos, N. and Lippman, A. (1997). Empirical bayesian EM-based motion segmentation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR).*

[Vasconcelos and Lippman, 2001] Vasconcelos, N. and Lippman, A. (2001). Empirical bayesian motion segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(2):217–221.

[Veenman et al., 2001] Veenman, C., Reinders, M., and Backer, E. (2001). Resolving motion correspondence for densely moving points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(1):54–72.

[Vega-Riveros and Jabbour, 1989] Vega-Riveros, J. and Jabbour, K. (1989). Review of motion analysis techniques. *Communications, Speech and Vision, IEE Proceedings I*, 136(6):397–404.

[Vázquez et al., 2006] Vázquez, C., Mitiche, A., and Laganière, R. (2006). Joint multiregion segmentation and parametric estimation of image motion by basis function representation and level set evolution. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(5):782–793.

[Wang and Adelson, 1994] Wang, J. and Adelson, E. (1994). Representing moving images with layers. *IEEE Transactions on Image Processing*, 3(5):625–638.

[Wang and Duncan, 1996] Wang, W. and Duncan, J. (1996). Recovering the three-dimensional motion and structure of multiple moving objects from binocular image flows. *Computer Vision and Image Understanding*, 63(3):430–446.

[Wang et al., 2006] Wang, Y., Loe, K.-F., and Wu, J.-K. (2006). A dynamic conditional random field model for foreground and shadow segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(2):279–289.

[Weber and Malik, 1997] Weber, J. and Malik, J. (1997). Rigid body segmentation and shape description from dense optical flow under weak perspective. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):139–143.

[Weiss, 1997] Weiss, Y. (1997). Smoothness in layers: Motion segmentation using nonparametricmixture estimation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 520–526.

[Weiss and Adelson, 1996] Weiss, Y. and Adelson, E. (1996). A unified mixture framework for motion segmentation: incorporating spatial coherence and estimating the number of models. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 321–326.

[Xiao and Shah, 2005] Xiao, J. and Shah, M. (2005). Motion layer extraction in the presence of occlusion using graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1644–1659.

[Yang et al., 2002] Yang, M.-H., Ahuja, N., and Tabb, M. (2002). Extraction of 2D motion trajectories and its application to hand gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(8):1061–1074.

[Yilmaz et al., 2004] Yilmaz, A., Li, X., and Shah, M. (2004). Contour-based object tracking with occlusion handling in video acquired using mobile cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1531–1536.

[Zhang and Hanauer, 1995] Zhang, J. and Hanauer, G. (1995). The application of mean field theory to image motion estimation. *IEEE Transactions on Image Processing*, 4(1):19–33.

[Zhang, 1996] Zhang, Y. J. (1996). A survey on evaluation methods for image segmentation. *Pattern Recognition*, 29(8):1335–1346.

[Zhang, 2001] Zhang, Y. J. (2001). A review of recent evaluation methods for image segmentation. In *Sixth International Symposium on Signal Processing and its Applications*, volume 1, pages 148–151, Kuala Lumpur, Malaysia.

Thesen

- Bei der Bewegungsschätzung zeigt das SSD-Maß bessere Performance als SAD bei verrauschten Bildern.
- Auf modernen Hauptprozessoren wird SSD schneller berechnet als SAD.
- Kreuzkorrelationsbasierte Ähnlichkeitsmaße bedürfen einer Normalisierung.
- Regularisierung ermöglicht das korrekte Segmentieren innerer Teile von homogenen Flächen.
- Wenn visuelle Merkmale (Cues) in einem Segmentierungsprozess fusioniert werden, können sie Ausfälle von anderen Merkmalen, z.B. Bewegungsinformation, kompensieren.
- Visuelle Merkmale (Cues) können in Kanten- und Flächen-Cues unterteilt werden. Merkmale der gleichen Klasse können auf einheitliche Weise in den Segmentierungsprozess eingebunden werden.
- Die Einbindung und Nutzung von mehrdeutigen Ergebnissen einer Bewegungsschätzung verbessert die Segmentierungsleistung.

Hamburg, 02.02.2007

Boris Lau