Fortschritt-Berichte VDI

VDI

Reihe 10

Informatik/
Kommunikation

Dipl.-Ing. Matthias Reso,
San Francisco

Nr. 861

Temporally Consistent Superpixels

Institut für Informationsverarbeitung
www.tnt.uni-hannover.de

Temporally Consistent Superpixels

Von der Fakultät für Elektrotechnik und Informatik

der Gottfried Wilhelm Leibniz Universität Hannover

zur Erlangung des akademischen Grades

Doktor-Ingenieur

(abgekürzt: Dr.-Ing.)

genehmigte

Dissertation

von

Dipl.-Ing. Matthias Reso

geboren am 9. Januar 1985 in Hannover.

2017

Referent:	Prof. Dr.-Ing. J. Ostermann
Korreferent:	Prof. Dr.-Ing. R. Stiefelhagen
Vorsitzender:	Prof. Dr.-Ing. B. Rosenhahn
Tag der Promotion:	09.08.2017

Fortschritt-Berichte VDI

Reihe 10

Informatik/
Kommunikation

Dipl.-Ing. Matthias Reso,
San Francisco

Nr. 861

Temporally Consistent Superpixels

Institut für Informationsverarbeitung
www.tnt.uni-hannover.de

Reso, Matthias
Temporally Consistent Superpixels
Fortschr.-Ber. VDI Reihe 10 Nr. 861. Düsseldorf: VDI Verlag 2018.
124 Seiten, 54 Bilder, 3 Tabellen.
ISBN 978-3-18-386110-1, ISSN 0178-9627,
€ 48,00/VDI-Mitgliederpreis € 43,20.

Keywords: Superpixels – Temporal Consistency – Supervoxels – Oversegmentation – Occlusion – Interactive Video Segmentation

This thesis addresses the field of early stage video preprocessing in order to improve and accelerate subsequent processing steps like semantic video segmentation or video-based object tracking. A framework is proposed to segment video streams into temporally consistent superpixels in order to create a representation of the video with far less image primitives than the voxel-grid. The proposed energy-minimization-based approach utilizes a novel hybrid clustering strategy for a multidimensional feature space. Techniques are presented to ensure the consistency of the superpixel flow with the image movement while considering visual occlusion and disocclusion effects. The effectiveness of the proposed method is shown by a comparison to state-of-the-art spatio-temporal oversegmentation algorithms using established benchmark metrics. Additionally, its effectiveness is further demonstrated by showing its application on the real-world scenario of interactive video segmentation.

Bibliographische Information der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliographie; detaillierte bibliographische Daten sind im Internet unter www.dnb.de abrufbar.

Bibliographic information published by the Deutsche Bibliothek
(German National Library)
The Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliographie (German National Bibliography); detailed bibliographic data is available via Internet at www.dnb.de.

ISSN 0178-9627

ISBN 978-3-18-386110-1

Acknowledgement

The present thesis was written during my time at the Institut für Informationsverarbeitung (TNT) of the Gottfried Wilhelm Leibniz Universität Hannover.

I owe my deepest gratitude to Prof. Dr.-Ing. Jörn Ostermann for giving me the opportunity to work in his group. His excellent supervision, encouragement and patience during my time at the institute helped to make this thesis a success.

I also would like to thank Prof. Dr.-Ing. Bodo Rosenhahn for his guidance and helpful comments on my work done at the institute as well as for being the chair of the defense committee.

Thanks is also due to Prof. Dr.-Ing. Rainer Stiefelhagen for being the second examiner of my thesis.

Special thanks goes to Jörn Jachalsky from whom I learned a lot during our fruitful discussions and our collaboration on many research papers.

I am very grateful to my dear colleagues and friends at the TNT like Arne Ehlers, Benjamin Jungbluth, Florian Baumann, Björn Scheuermann, Hanno Ackermann, Kai Cordes and Stephan Preihs who contributed their support and made my stay at the institute to one of the greatest times in my life.

I also thank Matthias Schuh, Martin Pahl, Thomas Wehberg and the secretaries, Mrs. Brodersen, Mrs. Bank and Mrs. Jaspers-Göring for their commitment and assistance.

A very special thanks goes to my family and particularly my parents Andrea and Henning Reso for enabling my studies and always supporting me in my endeavours. And to my beloved wife Julia who stood by my side even in the darkest hours.

This work has been partially funded by the ERC within the starting grant Dynamic MinVIP. The author gratefully acknowledges the support.

Dedicated to my family

Contents

Abbreviations

2D	two-dimensional
3D	three-dimensional
5D	five-dimensional
6D	six-dimensional
CCD	charge coupled device
CMOS	complementary metal oxide semiconductor
CPU	central processing unit
CRF	conditional random field
FIFO	first-in, first-out
GPU	graphics processing unit
HD	high-definition
KLT	Kanade-Lucas-Tomasi feature tracker
MAP	maximum a posteriori
MRF	Markov random field
OVS	online video seeds
SA	3D segmentation accuracy
sGBH	streaming hierarchical video segmentation
SLIC	simple linear iterative clustering
SOR	successive over-relaxation
TCS	temporally consistent superpixels
TEX	temporal extent
TSP	temporal superpixels
UE	3D undersegmentation error

Symbols and Notation

α	spatial weighting term with an interval of [0,1]
$\bar{\alpha}$	alternative spatial weight with an interval of $[0,\infty)$
B	number of labels i.e. cluster, superpixels, segments
$\boldsymbol{b}$	labeling of a set of pixels
b	label variable
$\hat{\boldsymbol{b}}$	optimal labeling
$\hat{b}$	optimal label
β	weighting term including the feature variance
$\bar{C}$	superpixel compactness
$\boldsymbol{c}$	set of pixel pairs
$D_i(b)$	unary term for assigning label b to pixel i
$E[\cdot]$	expectation value
$E_{\text{asm}}(i,b)$	assignment energy function
$E_{\text{clg}}(\vec{\omega})$	combined local-global optical flow energy function
$E_{\text{clg,nq}}(\vec{\omega})$	non-quadratic combined local-global energy function
$E_{\text{crf}}(\boldsymbol{b})$	energy function of a conditional random field
$E_{\text{hs}}(\vec{\omega})$	Horn-Schunck optical flow energy functional
$E_{\text{ivs}}(\boldsymbol{b})$	energy function for interactive video segmentation
$E_{\text{kms}}(\boldsymbol{b})$	total energy of general k-means
$E_{\text{lk}}(\vec{\omega})$	Lucas-Kanade optical flow energy function
$E_{\text{mrf}}(\boldsymbol{b})$	energy function of a Markov random field
$E_{\text{tcs}}(\boldsymbol{b})$	total energy of a temporally consistent superpixel labeling
$E^{\text{tcs}}_{\text{col}}(i,b)$	color part of the TCS assignment energy
$E^{\text{tcs}}_{\text{pos}}(i,t,b)$	spatial part of the TCS assignment energy
ϵ	misclassification rate
EV	explained variation
F	number of future frames
$G(\vec{y})$	window around a vector $\vec{y}$
$\boldsymbol{g}$	set of pixels contained in a ground truth segment
$\Gamma(\cdot,\cdot)$	indicator function
γ	scale and growth parameter of the penalty function Ψ
$I(x,y,t)$	image functional
i,j,k	pixel indices
$K(\cdot)$	kernel function

$\boldsymbol{k}$	set of possible labels
L_b	perimeter of superpixel b
l	triangle index
$\lambda_{\mathcal{F}}$	smoothness regularization constant for the optical flow field
$\lambda_{\mathcal{G}}$	label consistency weight
$\vec{\mu}_b^{\mathcal{C}}$	superpixel color center
$\vec{\mu}_b^{\mathcal{S}}$	superpixel spatial center
$\boldsymbol{\mu}$	set of superpixel centers
$\vec{\mu}_{b,t}^{\mathcal{S}}$	spatial center of a temporally consistent superpixel
$\vec{\mu}_{b,t}^{\mathcal{F}}$	weighted average flow of superpixel b in frame t
N	number of pixels in an image
$\boldsymbol{n}$	set of pixels or sites
$\boldsymbol{n}_b$	set of pixels in superpixel with label b
$\boldsymbol{n}_{b,t}$	set of pixel in superpixel with label b at time t
$\boldsymbol{n}_{b,t}^{\mathcal{H}}$	hidden fraction of superpixel b in frame t
$\boldsymbol{o}$	set of observations
o	observation
$\vec{\omega}$	optical flow vector
$\underline{\vec{\omega}}$	optical flow vector in homogeneous coordinates
P	number of past frames
$P(\cdot)$	probability function
$\Phi(\cdot)$	potential function
$\Psi_i(\cdot)$	non-quadratic penalty function
Q_b	iso-perimetric quotient of superpixel b
$\bar{S}$	average superpixel edge length
SA	3D segmentation accuracy
$\sigma_{\mathcal{F}}^2$	variance of the average flow weighting function
T	total number of frames
t	time index
τ	local frame index
UE	3D undersegmentation error
$V_{i,j}(b,d)$	pairwise energy for assigning labels b, d to the pixels i, j
VoA	variance of area
W	length of the sliding window
$\vec{x}_i^{\mathcal{C}}$	color vector of a pixel
$\vec{x}_i^{\mathcal{S}}$	spatial vector of a pixel
Z	number of iterations
z	iteration index
$\bar{\zeta}_m$	averaged benchmark result m

Abstract

A wide variety of computer vision applications rely on superpixel or supervoxel algorithms as a preprocessing step. This underlines the overall importance that these approaches have gained in recent years. However, most methods show a lack of temporal consistency or fail in producing temporally stable segmentations.

In this regard, this thesis presents a highly competitive approach for temporally consistent superpixels for video content. The approach is based on energy-minimizing clustering utilizing a novel hybrid clustering strategy for a multidimensional feature space. By working in a joint global color space, but keeping the positions of the superpixels localized to the frame level, the framework allows for arbitrary large displacements of the superpixels along the image plane over time. By applying a contour-based optimization the spatial coherency of the pixels of each superpixel is ensured while obeying the optimization target at all times. A sliding window technique enables the approach to process videos of arbitrary length in a streaming fashion. To propagate the superpixel segmentation while shifting the sliding window over the video volume, this thesis proposed two novel propagation methods. While the first approach is trimmed for efficiency and utilizes sparse optical flow vectors in combination with a Delaunay triangulation, the second approach individually propagates the shape of each superpixel. The individual propagation enables the detection of occluded and disoccluded image regions. In order to provide equally sized superpixels, this thesis further proposes a novel approach to handle structural changes in the video volume by utilizing the collected dis-/occlusion information.

For a thorough evaluation, the proposed approach is compared to state-of-the-art spatio-temporal oversegmentation algorithms using established benchmark metrics. The benchmark results show that the proposed framework produces the lowest spatio-temporal segmentation error of all approaches. Thereby, creating longer temporal superpixel trajectories than approaches with a comparable segmentation error. This shows that the proposed method extracts the temporal connections of the image regions inherent in the video volume to a higher extent than previous methods. Simultaneously, its run time scales better than approaches of comparable quality, as it only depends linearly on the number of pixels as well as the number of superpixels.

The effectiveness of the proposed method is further evaluated by showing its application to the task of interactive video segmentation using graph cut techniques. When compared to a voxel level processing of the video material the proposed oversegmentation method decreases the initial segmentation error by over 47 %. Additionally, its application reduces the average run time of the performed graph cut from 31 minutes to under 7 ms per sequence.

Keywords: superpixels, temporal consistency, supervoxels, oversegmentation, occlusion, interactive video segmentation

Kurzfassung

Eine große Anzahl Computer Vision Applikationen basiert auf der Verwendung von Superpixeln oder Supervoxeln als Vorverarbeitungsschritt. Dies unterstreicht die Wichtigkeit, welche diese Ansätze in den letzten Jahren erlangt haben. Viele dieser Methoden erzeugen allerdings zeitlich inkonsistente oder instabile Segmentierungen.

Ziel dieser Arbeit ist die Beschreibung eines Systems zur Erzeugung zeitkonsistenter Superpixelsegmentierungen für Videos. Der Ansatz basiert auf einem energieminimierenden Verfahren zur Cluster Analyse und nutzt einen neuen, hybriden Ansatz für den multidimensionalen Merkmalsraum. Dabei kommt ein globaler, zusammengefasster Farbraum zur Anwendung, während die räumlichen Positionen der Superpixel auf den Einzelbildern betrachtet werden. Somit lassen sich beliebig große Bewegungen von Bildregionen entlang der Bildebene durch die Superpixel abbilden. Indem eine konturbasierte Optimierung Anwendung findet, wird der räumliche Zusammenhalt der Pixel jedes Superpixels garantiert, während das Optimierungskriterium zu jedem Zeitpunkt Berücksichtigung findet. Durch den Einsatz einer Fensterungstechnik lassen sich dabei beliebig lange Videosequenzen sukzessiv verarbeiten. Um die Segmentierung während der sukzessiven Verarbeitung zu propagieren, werden in dieser Arbeit zwei neue Ansätze hierfür vorgestellt. Während beim Ersten großes Augenmerk auf die Effektivität gelegt wird und eine Delaunay Triangulation in Kombination mit einzelnen, verfolgten Merkmalspunkten Anwendung findet, propagiert der Zweite jeden Superpixel einzeln. Hierbei lassen sich Rückschlüsse auf verdeckte und aufgedeckte Bildregionen ziehen. Diese Informationen werden im weiteren Verlauf dazu genutzt, um auf strukturelle Änderungen im Videovolumen zu reagieren und hierdurch möglichst gleichgroße Superpixel zu generieren.

In einer umrangreichen Evaluierung mit etablierten Testverfahren wird das vorgstellte System mit aktuellen Verfahren zur Videoübersegmentierung verglichen. Die Ergebnisse zeigen, dass das vorgeschlagene Verfahren den geringsten Segmentierungsfehler aufweist. Gleichzeitig werden zeitlich längere Superpixeltrajektorien erzeugt als von Verfahren vergleichbarer Segmentierungsqualität. Dies zeigt, dass das vorgestellte Verfahren die im Video enthaltenen zeitlichen Verbindungen der Bildregionen besser extrahiert als frühere Ansätze. Gleichzeit skaliert die Laufzeit des Verfahrens besser, da sie nur linear mit der Anzahl der Pixel und Superpixel ansteigt.

Darüber hinaus wid die Leistungsfähigkeit des Verfahren am Beispiel der interaktiven Videosegmentierung mittels des Graph-Cut Algorithmus demonstriert. Verglichen mit einer pixelweisen Verarbeitung des Videomaterials veringert sich der initiale Segmentierungsfehler bei Anwendung des vorgestellten Verfahrens um über 47 %. Zusätzlich verkürzt sich die durchschnittliche Ausführungszeit des Graph-Cut Algorithmus von 31 Minuten auf unter 7 ms pro Sequenz.

Stichworte: Superpixel, Zeitkonsistenz, Supervoxel, Übersegmentierung, Verdeckung, interaktive Videosegmentierung

Chapter 1

Introduction

1.1 Motivation

With the advent of smartphone cameras and similar capturing devices the presence of digital image processing has become ubiquitous in the past years. As these devices provide an easy way of capturing and sharing everyday situations, the amount of video and image data that needs to be processed is growing from day to day. Additionally, the image resolution of the capturing devices is increasing over time which further boosts the scale of available data. To keep up with the growing amount of data, it is therefore essential to further improve the efficiency of the processing and to extract the relevant information at an early processing stage.

To this end, this thesis deals with the problem of video oversegmentation as a preprocessing step in a video processing pipeline. A common processing pipeline consists of a capturing stage, where video data is recorded, and a processing stage that extracts some information of interest from the captured footage. In a digital system the video footage is captured and stored in the familiar pixel grid structure.

The regular structure is inherited from the underlying capturing sensor (commonly a CCD or CMOS array) and is used throughout the pipeline as the basic image representation. While this representation provides an efficient way of organizing and addressing pixel data, it does not provide additional structural information to further processing stages such as visible shapes or textures. Simultaneously, the recorded image data is in general not in alignment with the grid structure of the capturing device. This is due to the fact that natural contours of objects are often irregularly shaped and in general do not obey the orthogonality of the pixel grid.

Therefore, Ren and Malik proposed in [60] to complement the efficient organization of pixel information with a more content adaptive, data driven structure. The structure is created through the grouping of spatially coherent pixels by using low-level features such as color or texture. Thereby, an image segmentation is created which in general contains more segments than visible object parts. Simultaneously, it contains far less segments than the original number of pixels. The underlying assumption is that pixels which are spatially coherent and share their low-level features

are part of the same object and thus can be grouped and handled together. These so called superpixels can then be utilized as primitives for further content analysis and processing. The pixel grouping leads to a major reduction of image primitives which results in an increased computational efficiency for subsequent processing steps. Further, it allows for more complex algorithms computationally infeasible on pixel level [60] as well as the creation of a spatial support for region-based features [37].

The applications of superpixels are widely spread and include e.g. video segmentation [48, 31], tracking [91], multi-view object segmentation [22], scene flow [87], 3D layout estimation of indoor scenes [94], interactive scene modeling [83], image parsing [77], and semantic segmentation [39, 67]. In particular applications processing video data benefit from the usage of superpixels. This is due to the reduction of raw pixel data that needs to be processed in later stages. But superpixel algorithms such as [1, 25, 47, 50, 57, 58, 74, 85, 90] are mainly designed for the application on single images. Therefore, the results show volatile and flickering superpixel contours when applied to video sequences. Even if there are only slight changes between consecutive frames. Moreover, by design no temporal connections between superpixels in successive video frames are determined. Consequently, the same image regions in consecutive frames are not consistently labeled, but get assigned a different superpixel label in each frame. An example for a consistent superpixel labeling is depicted in Figure 1.1. Here, the superpixels of different frames occupying the same corresponding image region, i.e. they are part of the same spatio-temporal segment, are labeled with the same color.

In addition to the further reduction of the overall number of image primitives, a temporally consistent labeling can be very beneficial for many applications. An example could be a manual video segmentation task, as shown in Figure 1.1, where a human operator divides regions into foreground and background. For the sake of simplicity, it is assumed that the separation is done by pointing at the superpixels. In the case of a temporally inconsistent segmentation, the operator would have to select the superpixels of the soccer players in Figure 1.1 in every single frame to assign them to the foreground. In the temporally consistent case, the selection in a single frame would be sufficient in most places as the superpixels in other frames would be automatically selected via the temporal connections. Thus, less manual interaction by the operator would be needed to yield an equivalent final segmentation result.

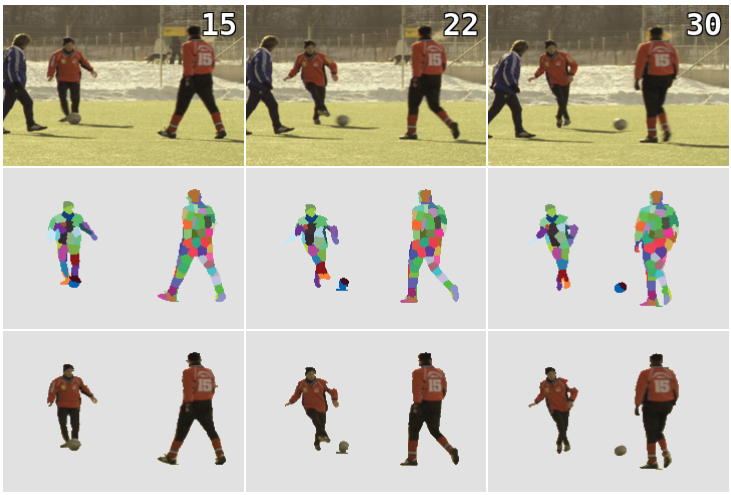


Figure 1.1: Top row: original sequence with frame numbers. Middle row: a subset of superpixels is manually selected in frame 15 and shown as color-coded label map. The superpixels are generated by a variant of the approach proposed in this thesis and same colors indicates temporal consistency. Bottom row: a video segmentation is created by cutting out the soccer players based on the selected superpixels. Images taken from [63].

1.2 Related Works

The available approaches for spatio-temporal oversegmentation can be classified as either generating superpixels with temporal consistency (e.g. [14, 82, 46]) or supervoxels (e.g. [34, 85, 1]). The former approaches produce a valid superpixel segmentation for each frame with spatially coherent segments and a consistent labeling over multiple frames. The latter conceive the video as a 3D voxel volume and produce a grouping of the voxels, where each group is connected through the spatio-temporal connections between the voxels. The voxel-grid is either build up using the 6- or 26-connected neighborhood in the video volume or through connections derived through optical flow vectors between frames [34].

The relation between supervoxels and superpixels with temporal consistency can be described in the following way: Superpixels with temporal consistency can be stacked up to build supervoxels. Similarly, a superpixel representation with temporal consistency can be obtained by slicing a supervoxel representation at frame instances. The latter conversion is only valid as long as the cross section of a super-

voxel at a frame instance does not split up into spatially non-contiguous segments. The reason is that spatially non-coherent segments on the image plane are valid for a supervoxel segmentation if the spatio-temporal regions are spatially connected through another frame. Simultaneously, such a split results in an invalid superpixel segmentation of this frame.

In the following, a brief overview of available approaches for supervoxels and superpixels with temporal consistency will be given. An early example of this kind of algorithms, which is not explicitly labeled as a superpixel or supervoxel approach but shares a similar idea, is proposed by Zitnick et al. in [95].

1.2.1 Supervoxel Methods

In [85], Veksler et al. propose a first supervoxel termed approach. For their creation the video volume is covered with overlapping cuboids, whereas each cuboid corresponds to one supervoxel in the final segmentation. The volume of each cuboid determines the maximum volume of the supervoxel to be generated. Thus, longer cuboids encourage higher temporal consistency. The assignment of an exclusive supervoxel label to each voxel is achieved by formulating an energy function incorporating image gradients and minimizing this energy function using graph cut.

Grundmann et al. [34] propose an approach for hierarchical video segmentation that is based on the graph-based image segmentation method introduced by Felzenszwalb and Huttenlocher in [26]. To leverage the benefits of color histograms, the optimization procedure of [26] is applied twice on the voxel graph of the video volume. In a first run, neighboring voxels are merged into small voxel groups from which color histograms are computed. Based on the chi-square distances of their color histograms the voxel groups are further merged into larger spatio-temporal regions. By keeping track of the mergers a hierarchical video segmentation is created. In [93], Xu et al. add streaming capabilities to [34] by applying a Markovian assumption on chunks of the segmented video stream.

In [1], Achanta et al. describe a supervoxel extension of their clustering-based superpixel approach by extending the clustered data points with a temporal dimension. Thereby, each voxel is viewed as a data point in a six-dimensional feature space consisting of three color, two spatial, and one temporal dimension. The supervoxels form clusters of data points in the feature space which are revealed by a standard cluster analysis technique. To improve the efficiency of the cluster analysis, a technique is proposed to limit the extent of the search space during the analysis.

1.2.2 Superpixel Methods

A first approach towards superpixels with temporal consistency is introduced by Levinshtein et al. [46]. The approach is based on the TurboPixel algorithm, proposed in [47], which uses level-set techniques to grow equally distributed seed points

into non-overlapping superpixels. To derive a temporally consistent segmentation, Levinshtein et al. propose to propagate the central point of each superpixel to initialize the seeds for the superpixels in the next frame. The propagation is performed by using optical flow information [46]. Given the propagated seeds, the superpixels are then grown on frame level only. While achieving a more temporally stable superpixel segmentation and determining their temporal connections, the approach defines no mechanism to handle structural changes in the video such as occlusion or disocclusion.

In [14], Chang et al. propose a generative probabilistic framework for temporal superpixels. Here, each pixel of the segmentation is modeled as a random variable which can take values from the set of superpixel labels. The inference is done in a frame-by-frame fashion by optimizing a joint log-likelihood function defined over all random variables given the observed pixel data. For the optimization, label changes are proposed which are only accepted if they preserve the spatial coherence of the superpixels and improve the value of the objective function. The superpixel movement from frame to frame is modeled by a Gaussian process initialized using optical flow. To address the problem of structural changes the authors propose split, merge, and switch moves, where superpixels can be split up into two, merged together, or take the label of a superpixel that was merged in the past, respectively. The moves are proposed randomly and are accepted if the new resulting segmentation increases the joint log-likelihood function.

Van den Bergh et al. extend their superpixel approach, proposed in [81], to be applied on video streams in [82]. The approach uses color histograms to represent superpixels and sets-up an objective function which is maximized if the number of populated bins per histogram is minimized. The proposed hill-climbing algorithm optimizes the segmentation by proposing the reallocation of single pixels or pixel blocks from one superpixel to a neighboring superpixel and accepting these changes if they maximize the objective function. Influenced by a parameter some frames are selected for termination and splitting of superpixels. To keep the amount of superpixels constant over time, for every terminated superpixel a new superpixel is created by splitting off a part from another superpixel. The decision on which superpixels to be selected for both cases is done according to which termination and splitting has the least influence on the objective function.

1.3 Challenges

On frame level the demands on a temporally consistent superpixel segmentation are basically the same as for a superpixel segmentation that is performed on a single image. Meaning that the superpixel contours should follow object boundaries and be sensitive to fine-grained details. On the other hand, especially in the absence of boundaries, their shape should be as compact as possible. Additionally, the video

frames should be divided into equally sized regions as subsequent processing steps can benefit from a homogeneous size and shape, as it has been pointed out previously e.g. in [47, 60, 85].

Furthermore, a temporally consistent superpixel segmentation should not violate object boundaries across different frames. This means a superpixel should not flip from being part of an object in one frame to being part of another object in another frame. Instead, each superpixel should follow its underlying image region when the region moves over time. An example of a temporally consistent superpixel is shown in Figure 1.1, where e.g. the soccer ball is occupied by the same superpixel over multiple frames. To completely extract the temporal connections of the image regions which are inherent in the video stream the temporal trajectory of the superpixel should ideally be as long as the time for which the image region is visible.

Scenes, as the one depicted in Figure 1.1, often involve moving objects or camera motion. These in general introduce structural scene changes in the some form of occlusion or disocclusion of image regions. Since many superpixel algorithms for videos sequentially work on the single frames of the video and propagate the previous segmentation (in some form) onto the next frame, they have to explicitly handle these structural changes. For a segmentation to represent the structural changes in a correct way, its segments should disappear as soon as the corresponding image regions get occluded. Similarly, new superpixels should be created, where a new image region gets disoccluded. If these structural changes are not handled (as e.g. in [46]) a temporally consistent segmentation will inevitably lead to superpixels of inhomogeneous size. An example for such a case is visualized in Figure 1.2. In contrast to the superpixel approaches, supervoxel methods conceive the video as a 3D volume and therefore no propagation is needed in their case. But as these approaches often come without the notion of compactness on the frame level, they inherently are more likely to produce superpixels of inhomogeneous size when sliced on the frame instance. Supervoxel approaches which include a notion of compactness, such as [1] and [85], limit the temporal duration of the supervoxels in an implicit or explicit way, respectively [65]. Thereby, these approaches circumvent the need to handle the structural changes but also fail to create long-term superpixel trajectories. In order to create long-term trajectories, while keeping the superpixel size homogeneous over time, approaches, such as [14] and [82], delete and create superpixels based on the objective function that is defined to minimize the segmentation error. But these functions lack any explicit awareness about the occlusion or disocclusion boundaries present in the video material. Thereby, the locations at which superpixels are terminated or created often do not coincide with the actual dis-/occlusion boundaries but turn out to be at rather random spots in the scene. It is therefore important to explicitly identify the locations where dis-/occlusion happens, in order to adapt the segmentation at the right spot.

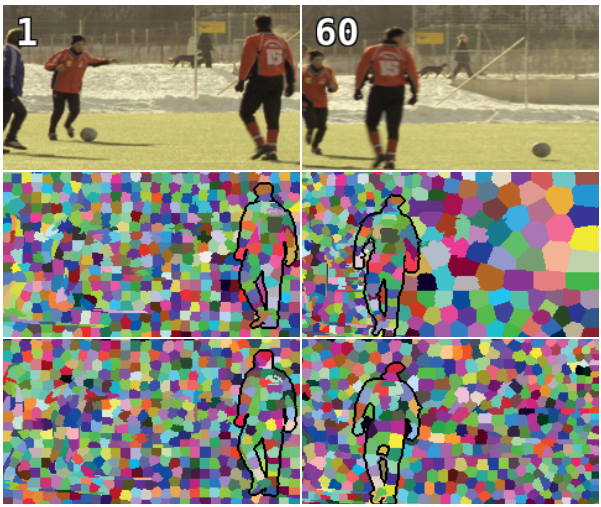


Figure 1.2: Top row: frame 1 and 60 of the soccer sequence. Middle row: label maps with temporal consistency but without a method to cope with structural changes in the video volume. The superpixels in the later part of the sequence are squeezed together on the left half of the frame while on the right half the size of the superpixels has to grow to fully occupy the newly uncovered image regions. Bottom row: label maps created with the proposed approach. The superpixels are temporally consistent and have equal size and shape over the whole sequence. (The silhouette of the player are manually marked for visualization purposes.)

Finally, as the segmentation into superpixels is a preprocessing step, to structure and organize the image data for subsequent processing stages, its execution should be as efficient and fast as possible. Thereby, ensuring that the overall processing time, needed to preprocess the data and to run the subsequent parts of the pipeline, does not surpass the time needed to run the original pipeline on pixel level.

1.4 Contributions

The goal of this work is to provide an efficient framework to create temporally consistent superpixel segmentations for arbitrary long video sequences. The main contributions of this work are as follows:

- A novel hybrid clustering scheme is introduced which transforms a well known superpixel approach for single images into a video approach for temporally consistent superpixels. The new approach enables streaming capabilities by utilizing a sliding window which is shifted over the video volume. The hybrid optimization scheme jointly comprises the pixel color information of multiple frames, while keeping the spatial position of the pixels localized to the frame level. By choosing this approach the segmentation error can be minimized while allowing for arbitrary movements of the superpixels along the image plane over time.

- In order to initialize new frames entering the sliding window, this work proposes two new approaches to propagate the superpixel labelings of previous frames. The first approach is trimmed for efficiency and exploits the robustness of a set of sparse feature points which are tracked between the frames. By spanning a triangle mesh over the feature points and warping the superpixel segmentation according to the transformed mesh, the consistency of the spatial superpixel constellation is ensured over time. A second approach utilizes dense optical flow to propagate each superpixel individually onto the next frame. Simultaneously, the approach detects and collects dis-/occlusion information about the propagated superpixels.

- Further, it is proposed to utilize the dis-/occlusion information to explicitly handle the structural changes in the video volume. This is achieved by identifying the dis-/occlusion boundaries and adapt the superpixel segmentation at these locations. It is further proposed to integrate the information on partially occluded superpixels into the optimization process, in order to improve the compliance of the superpixel flow with the underlying image flow.

- The proposed method is evaluated thoroughly by optimizing its parameters for multiple objectives simultaneously on independent training material. Subsequently, the method is compared to the state-of-the-art oversegmentation approaches for videos. The evaluation is performed utilizing well established benchmark metrics and a huge variety of input video data.

- Finally, the effectiveness of the novel approach for temporally consistent superpixels is shown, by applying it to the scenario of interactive video segmentation. Thereby, the reduced segmentation error as well as the improved run time performance are evaluated and compared to other state-of-the-art methods.

1.5 Structure of this Thesis

This thesis is structured as follows. The fundamental topics utilized throughout this thesis are briefly summarized in Chapter 2. Subsequently, the hybrid clustering approach for temporally consistent superpixels is described in Chapter 3. This includes the feature space selection, the hybrid optimization scheme, as well as the sliding window technique. The contributions of this chapter were previously published in [63] and [65]. Chapter 4 describes the mesh-based propagation technique and a superpixel size-based handling of the structural changes in the video volume previously published in [65] and [66]. Further, the chapter introduces the occlusion-aware propagation of individual superpixels and the integration of the collected dis-/occlusion information into the optimization procedure. Chapter 5 contains the selection of the parameters and the detailed evaluation of the proposed methods as well as a thorough comparison to other state-of-the-art oversegmentation approaches for video content. In addition, this chapter includes the demonstration of the new approach on the task of an interactive video segmentation, as it was in parts previously published in [64]. Finally, the thesis is concluded in Chapter 6.

Chapter 2

Fundamentals

This chapter revisits the fundamental concepts utilized in this thesis which can be separated into two main categories. The first part of the chapter will be focused on different techniques for image segmentation. This includes graph-based image segmentation utilizing Markov random fields (MRFs) or conditional random fields (CRFs) as well as clustering-based techniques. In the second part of this chapter, several approaches for the computation of optical flow between two consecutive frames will be described.

2.1 Image Segmentation

In this thesis, image segmentation is seen as the task of assigning a label to all pixels of an image from a given set of labels. By assigning different labels to pixels these pixels are divided into disjoint groups. To create a meaningful segmentation, those pixels which share a common feature like their color value get assigned the same label. In general, the segmentation can be done either interactive, semi-automatic or fully automatic. A fully automatic algorithm has no prior knowledge that is specific to the current input data but only generic knowledge such as it can be extracted from a training data set. In the semi-automatic case, prior knowledge can be given by a user such as a region of interest or a set of images containing the same object, like in the co-segmentation task [70]. For an interactive algorithm the user gives an initial input as well as further feedback at locations where the segmentation of the algorithm is incorrect. This feedback-loop can be repeated until the user is satisfied with the segmentation result. With the exception of Section 5.2 this thesis will be focused on the fully automatic case.

In addition to the grade of independence from user input, the task of image segmentation can be further categorized by the granularity of the final segmentation. The pixels of an image can either be separated into foreground and background (binary segmentation) or into multiple segments (multi-label segmentation). In the latter case, multiple objects or instances of an object class exist and each gets assigned its own label. Another scenario is that different parts of a complex object get assigned separate labels, e.g. the different parts of the human body like arms, legs, torso and head. How many segments are created depends on the final task, the

Figure 2.1: Top row: example for a binary segmentation with one foreground and one background label. Bottom row: a multi-label segmentation with several different foreground labels.

image content, as well as the selected scale. Figure 2.1 shows example results for the case of binary and multi-label segmentation. A further reason to create a fine-grained multi-label segmentation has emerged with the raise of image resolution in modern capturing devices. Due to the higher image resolution, many computer vision tasks are infeasible to be executed on pixel level, as they take minutes to process a single frame [71]. Therefore, it has become popular to group the pixels of an image on the basis of low-level features like texture or color to gain an image representation with far less image primitives than the pixel-based representation. Simultaneously, far more primitives are created than objects or object parts are visible in the image. The preprocessing task of pixel grouping is called oversegmentation or superpixel segmentation (a term introduced in [61]). The task can be seen as an extreme case of a multi-label segmentation. As subsequent algorithms only have to deal with the reduced set of image primitives, their execution time is in general reduced as well. To gain an overall time advantage over the original algorithm on pixel level, the combined execution times of the oversegmentation and the subsequent algorithm applied on superpixel level have to be lower than before. An important criteria for the selection of an oversegmentation algorithm is therefore a minimal execution time. Besides their property to facilitate the computational burden of downstream image processing algorithms, they provide the possibility to create content adaptive, region-based features like histograms or texture features. The usage of these features can boost the quality of the downstream algorithms. It is therefore equally important for an oversegmentation algorithm that the segments of the oversegmentation do not cross the object contours visible in the image.

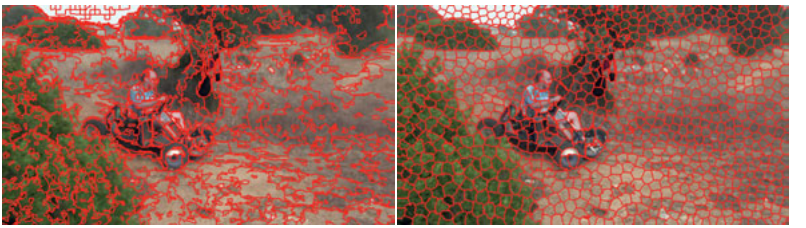


Figure 2.2: Comparison between a generic oversegmentation (left) and a superpixel segmentation (right) of an image. The segments of the superpixel segmentation are more compact and equally sized.

Although the term superpixel is often used as a synonym for oversegmentation, several authors have proposed criteria to distinguish between the two forms. The common properties of superpixel algorithms, as argued by [47] and [14], are the compact shape and homogeneous size of the superpixels. A comparison between a generic oversegmentation and a superpixel segmentation can be seen in Figure 2.2. In this work, the terminology of [47] and [14] will be adopted and the term oversegmentation will refer to the grouping of pixels or voxels based on low level features in general. The term superpixels will be used if the regions are created using some kind of compactness constraint.

While there exist many different approaches for image segmentation, this section will focus on two main categories which are important for the remainder of this work. First, a short introduction into graph-based image segmentation utilizing MRFs [9] or CRFs [42] will be given in Section 2.1.1. Second, the segmentation of images using clustering techniques like k-means [52], the simple linear iterative clustering (SLIC) approach [1], as well as the mean shift algorithm [18] will be discussed in the Sections 2.1.2 to 2.1.4.

2.1.1 MRF/CRF-based Image Segmentation

Since their introduction into the computer vision community by Geman and Geman in [32], Markov random fields have been used in many applications including image denoising [33], stereo-vision [40] and interactive image segmentation [8, 69]. A random field is a set of sites $\boldsymbol{n} = \{1,...,N\}$, where in the case of image segmentation each site represents a pixel. The neighborhood relations between the pixels are modeled by edges between the sites here denoted by the set of cliques $\boldsymbol{c}$. Each site has a corresponding random variable b_i which holds the label of the segment the pixel belongs to. Thus, the set of variables $\boldsymbol{b} = \{b_1, ..., b_N\}$ represents the labeling of the whole image. Additionally, the model contains a set of observed pixel values

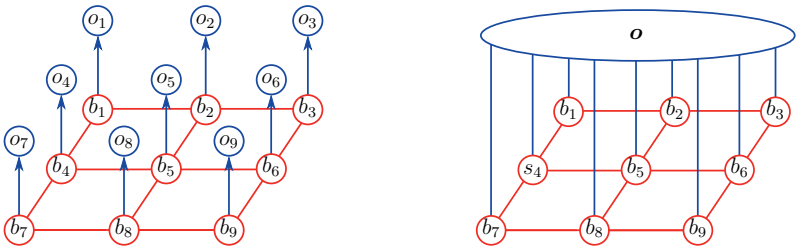


Figure 2.3: Visualization of the graphical models utilized for the two random field variants discussed in this section. (a) MRF: the value of each pixel depends on the underlying state but is independent from its neighbors. (b) CRF: the states globally depend on the image data.

$\boldsymbol{o} = \{o_1, ..., o_N\}$. The random variable of each site can take a value from the set of possible labels $\boldsymbol{k}$, i.e. $\boldsymbol{k} = \{FG, BG\}$ in the case of a binary segmentation.

If a generative model structure is assumed, where the observations directly depend on the underlying state configuration, the joint probability $P(\cdot)$ of the random field over the set of random variable $\boldsymbol{b}$ and the observed image data $\boldsymbol{o}$ can be written as

$$P(\boldsymbol{b},\boldsymbol{o}) = P(\boldsymbol{o}|\boldsymbol{b})P(\boldsymbol{b}) = \frac{1}{Z_1}\Phi_u(\boldsymbol{o}|\boldsymbol{b})\Phi_p(\boldsymbol{b}) \; . \tag{2.1}$$

Here, $\Phi(\cdot)$ are non-negative potential functions, where Φ_u reflects the likelihood of the pixels being generated from a specific labeling of $\boldsymbol{b}$ and Φ_p denotes the prior probability for a specific labeling. The partition function Z_1 is used to normalize the expression to form a proper probability function whose sum is equal to one.

By former assuming that the observations $\boldsymbol{o}$ are statistically independent from each other and thus the overall model has a structure as depicted in Figure 2.3(a), Equation (2.1) can be rewritten as

$$\frac{1}{Z_1}\Phi_u(\boldsymbol{o}|\boldsymbol{b})\Phi_p(\boldsymbol{b}) = \frac{1}{Z_1}\prod_{i\in\boldsymbol{n}}\Phi_u(o_i|b_i)\prod_{(j,i)\in\boldsymbol{c}}\Phi_p(b_i,b_j). \tag{2.2}$$

Here, the Markovian assumption is applied and thus the state of each random variable b_i depends only on the states of the variables included in its cliques. Further it is assumed that only pairwise potentials have a nonzero value.

To solve for the most probable labeling of the states given the observed image data, maximum a posteriori (MAP) inference which maximizes the posterior distribution

$P(\boldsymbol{b}|\boldsymbol{o})$ can be utilized. Using Bayes' rule the posterior probability for the labeling given the pixel data can be written as

$$P(\boldsymbol{b}|\boldsymbol{o}) = \frac{P(\boldsymbol{o}|\boldsymbol{b})P(\boldsymbol{b})}{P(\boldsymbol{o})} \propto P(\boldsymbol{o}|\boldsymbol{b})P(\boldsymbol{b}). \tag{2.3}$$

The last simplification is motivated by the fact that the prior for the image data is constant for all possible labelings. Therefore, the optimal labeling of the sites in the MAP sense $\hat{\boldsymbol{b}}$ is obtained by solving

$$\hat{\boldsymbol{b}} = \underset{\boldsymbol{b}}{\operatorname{argmax}}\, P(\boldsymbol{b}|\boldsymbol{o}) \tag{2.4}$$

$$\hat{\boldsymbol{b}} = \underset{\boldsymbol{b}}{\operatorname{argmax}}\, \frac{1}{Z_1} \prod_{i \in \boldsymbol{n}} \Phi_u(o_i|b_i) \prod_{(j,i) \in \boldsymbol{c}} \Phi_p(b_i, b_j) \tag{2.5}$$

$$\hat{\boldsymbol{b}} = \underset{\boldsymbol{b}}{\operatorname{argmax}}\, \frac{1}{Z_1} \exp - \left\{ \sum_{i \in \boldsymbol{n}} - \log \Phi_u(o_i|b_i) + \sum_{(j,i) \in \boldsymbol{c}} - \log \Phi_p(b_i, b_j) \right\}. \tag{2.6}$$

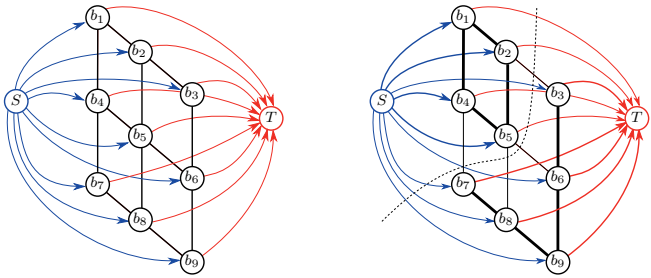
By defining the unary energy term $D_i(b) = -\log \Phi_u(o_i|b)$, the pairwise energy term $\lambda_{\mathcal{G}} V_{i,j}(b, d) = -\log \Phi_p(b,d)$, and considering only the exponent of Equation (2.6) the maximization problem can be converted into an energy minimization problem with the energy function

$$E_{\text{mrf}}(\boldsymbol{b}) = \sum_{i \in \boldsymbol{n}} D_i(b_i) + \lambda_{\mathcal{G}} \sum_{(j,i) \in \boldsymbol{c}} V_{i,j}(b_i, b_j). \tag{2.7}$$

Thereby, the function $V_{i,j}(b_i, b_j)$ is often referred to as the label compatibility function as it can be selected to prevent certain labels from occurring next to each other. The parameter $\lambda_{\mathcal{G}}$ is introduced to control the label smoothness, as with higher $\lambda_{\mathcal{G}}$ the costs for alternating labels become higher.
One of the basic assumptions made above is the statistically independence of the image data. But in general, this assumption does not hold, as the color of neighboring pixels is often similar and the change in color is smooth, suggesting a strong correlation. To relax the independence condition, it was proposed in [42] to use a CRF formulation, where the random variables of all sites globally depend on the observed image data. A possible structure is depicted in Figure 2.3(b). Assuming again that only pairwise clique potentials have nonzero contribution, the posterior distribution can be written as

$$P(\boldsymbol{b}|\boldsymbol{o}) = \frac{1}{Z_2} \prod_{i \in \boldsymbol{n}} \Phi_u(b_i,\boldsymbol{o}) \prod_{(j,i) \in \boldsymbol{c}} \Phi_p(b_i, b_j,\boldsymbol{o}) \tag{2.8}$$

$$P(\boldsymbol{b}|\boldsymbol{o}) = \frac{1}{Z_2} \exp - \left\{ \sum_{i \in \boldsymbol{n}} - \log \Phi_u(b_i,\boldsymbol{o}) + \sum_{(j,i) \in \boldsymbol{c}} - \log \Phi_p(b_i, b_j,\boldsymbol{o}) \right\}. \tag{2.9}$$



(a) (b)

Figure 2.4: (a) Graphical model proposed in [33] for the binary segmentation task. Each site is connected with a directed edge to the source and sink nodes, denoted with S and T, respectively. In (b), the line width of the arrows reflects the weights of the edges and the dotted line marks the minimal cut through the graph completely separating it into two partitions.

By defining a similar energy term as before which now depend on the observations, the MAP inference can again be rewritten as an energy minimization problem with the following energy

$$E_{\text{crf}}(\boldsymbol{b}) = \sum_{i \in \boldsymbol{n}} D_i(b_i, \boldsymbol{o}) + \lambda_{\mathcal{G}} \sum_{(j,i) \in \boldsymbol{c}} V_{i,j}(b_i, b_j, \boldsymbol{o}) \; . \tag{2.10}$$

For the binary segmentation case and constant pairwise terms the global optimal solution for (2.7) can be found using the min-cut/max-flow theorem proven by Ford and Fulkerson in [27] and first applied to the field of computer vision by Greig et al. [33]. The algorithm, proposed in [33], introduces two additional terminal nodes. The nodes are referred to as S- and T- nodes which represent the foreground and background label, respectively. Each node of the random field gets connected by a directed edge with both terminal nodes, as depicted in Figure 2.4(a). The weight of the edges is proportional to the unary energy term of the node reflecting the likelihood of the node belonging to the label, i.e. the segment represented by the terminal node. The pairwise edges are weighted according to the pairwise energy term. To derive the globally optimal labeling of the sites and thus the optimal segmentation in the sense of energy minimization, the cut has to be found which separates the S- and T-nodes and has the minimum total sum of weights. An example for a minimum cut is depicted in Figure 2.4(b), where bold lines indicate a high weight. For the case of a multi-label segmentation with an energy function

like (2.10) an approximate solution for the minimal cut can be found using the algorithm proposed in [8] which guarantees an optimal solution within a known factor of the global minimum.

Besides the graph-based segmentation algorithms, a second popular set of algorithms exists that perceives the segmentation problem as a clustering task. A subset of these algorithms will be described in the following sections.

2.1.2 Segmentation by Clustering

Clustering is the process of separating data points into groups whose features are similar in terms of some given measure. To segment a given image into visually distinct segments, the pixels i can be seen as data points, contained in a feature space, represented by its feature vector $\vec{x}_i$. As the pixels of an image segment should share a common feature, it is assumed that they form clusters in the feature space. Thus, each cluster should correspond to an image segment. Therefore, performing an image segmentation can be achieved by exploring the existing clusters in the feature space and assigning each pixel the label of the cluster it belongs to. Following the notation from the previous subsection, the label of a pixel i will be denoted by $b_i \in \boldsymbol{k}$.

An example for the arrangement of pixels in a feature space is depicted in Figure 2.5. The top row of the figure shows the original image and the corresponding multi-label segmentation map. For the second row 3000 pixels are randomly selected and plotted into the 3D feature space spanned by the dimensions of the RGB color space. Each data point is colored using the RGB values of the corresponding pixel. One can clearly distinguish the red, yellow, and green clusters of the petals, stamina, and background, respectively. In 2.5(d) a clustering of the data points is shown which corresponds to the segmentation shown in 2.5(b). The color of each data point corresponds to the assigned cluster and thus image segment.

One of the most popular clustering techniques is k-means clustering. Its basic concept was first described by Lloyd in 1957 and published later in [52]. In contrast to the graph-based segmentation techniques of Section 2.1.1, k-means includes no explicit model for pixel neighborhoods and therefore each pixel is seen as in independent data point in the feature space. In the original k-means algorithm the size B of the label set $\boldsymbol{k} = \{1,...,B\}$ has to be predefined by the user. To derive the optimal labeling $\hat{\boldsymbol{b}}$ for the data points, the k-means algorithms can be formulated as an energy minimization framework. Therefore, an energy function is defined which maps a given labeling $\boldsymbol{b} = \{b_i |\, i \,\forall\, \boldsymbol{n}\}$ of all pixels to an energy value.

$$E_{\text{kms}}(\boldsymbol{b}) = \sum_{i \in \boldsymbol{n}} E_{\text{asm}}^{\text{kms}}(i, b_i) \tag{2.11}$$

Here, $E_{\text{asm}}^{\text{kms}}(i,b)$ is the assignment energy cost which is needed to assign the data point i to the cluster b. In general, the energy cost is set to be proportional to

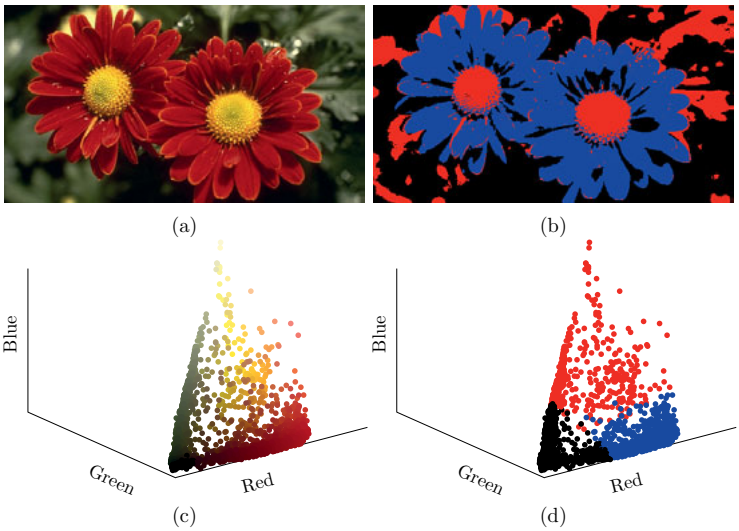


Figure 2.5: Real world example segmented using k-means clustering. In (a) and (b) the original image and the segmentation result with $\boldsymbol{k} = \{1,2,3\}$ are shown. In (c) and (d) 3000 randomly sampled pixels are plotted in the 3D RGB color space using the RGB color value and the corresponding cluster color of the pixels, respectively.

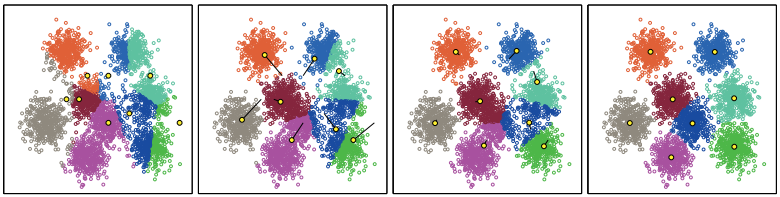
the Euclidean distance between the feature vector of the data point and the cluster center $\vec{\mu}_b$. The center is represented by the mean feature vector of the set of pixels assigned to the cluster $\boldsymbol{n}_b$. Here, the number of elements in a given set is denoted with $|\cdot|$.

$$E_{\text{asm}}^{\text{kms}}(i,b) \propto \sqrt{(\vec{x}_i - \vec{\mu}_b)^\top (\vec{x}_i - \vec{\mu}_b)} \tag{2.12}$$

$$\vec{\mu}_b = \frac{1}{|\boldsymbol{n}_b|} \sum_{i \in \boldsymbol{n}_b} \vec{x}_i \tag{2.13}$$

Finding the globally optimal assignment of pixels to the clusters is an NP-hard problem. But using an iterative scheme a locally optimal solution can be efficiently determined. The scheme is visualized in Figure 2.6 for a two-dimensional (2D) feature space with $|\boldsymbol{k}|=8$. The initial cluster centers are a randomly selected subset of feature vectors. Each subfigure shows the current cluster center position as black circles and the updated cluster assignments of each iteration step.

In each new iteration $z+1$ two steps are performed. First, given the mean feature vectors of the clustering of the previous iteration $\vec{\mu}_b^z$ each data point is assigned to



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

Figure 2.6: Example for an iterative cluster search in a 2D space using k-means ($|\boldsymbol{k}| = 8$). Only the first three iterations and the final clustering after convergence are shown. The cluster centers are denoted by black and yellow circles. Their position in the previous iteration is indicated by the black lines.

the cluster whose assignment energy cost is the lowest, i.e. whose Euclidean distance is the smallest

$$\hat{b}_i^{z+1} = \underset{b \in \boldsymbol{k}}{\operatorname{argmin}} \sqrt{(\vec{x}_i - \vec{\mu}_b^z)^\top (\vec{x}_i - \vec{\mu}_b^z)} \quad \forall i \in \{1,...,N\} \; . \tag{2.14}$$

In the following, this step will be denoted as the *assignment*-step. In the second step, the mean feature vectors of all clusters are updated using Equation (2.13) given the current assignment. These two steps are repeated until convergence occurs, i.e. the energy cost is not reducing any further or a maximum amount of iterations has been performed.

When the k-means algorithm is applied on the color space to generate an image segmentation, the created segments are in general not spatially coherent as it can e.g. be seen in Figure 2.5(b). This is due to the exclusive usage of the color space which does not provide any information about the pixels position and their spatial proximities. Another drawback of the approach is the huge search space for the minimal assignment energy, because for each pixel all cluster centers have to be checked for the minimal energy. To circumvent these disadvantages the, SLIC algorithm was proposed in [1]. The algorithm creates a superpixel segmentation with compact, spatially coherent segments and limits the search space for efficient processing. It will be described in more details in the next subsection.

2.1.3 SLIC Superpixels

The simple linear iterative clustering (SLIC) approach proposed by Achanta et al. [1] provides an efficient way to create spatially compact and coherent superpixels. Its basic principles were in parts already described in [96] and the authors of [73, 74] proposed a set of modifications which will be described in the following as well.

Like k-means, the approach conceives the pixels of an image as data points in a feature space and assumes that superpixels form clusters in this space. Instead of performing a cluster analysis in the RGB color space, the authors of [1] chose to represent every pixel i with a feature vector $\vec{x}_i = [l_i,a_i,b_i,x_i,y_i]^\top$. Thereby, l_i,a_i and b_i are the illumination and color difference components of the pixel i in the CIELAB color space and x_i,y_i its coordinates on the image plane. The CIELAB color space was chosen because of his perceptually uniform design [1] Although it was reported by [74] that the usage of the RGB color space can lead to slightly better results than using the perceptually uniform CIELAB color space.

To perform the clustering, SLIC optimizes the same objective function as k-means, stated in Equation (2.11). But in contrast to k-means the assignment energy for each pixel, here denoted by $\tilde{E}^{\text{slic}}_{\text{asm}}$, is defined in [1] as follows

$$\tilde{E}^{\text{slic}}_{\text{asm}}(i,b) = \sqrt{(E_{\text{col}}(i,b))^2 + (\tilde{\alpha}E_{\text{pos}}(i,b))^2}. \tag{2.15}$$

Where the energy functions $E_{\text{col}}(i,b)$ and $E_{\text{pos}}(i,b)$ are proportional to the Euclidean distances in the color space and in image coordinates, respectively:

$$E_{\text{col}}(i,b) \propto \sqrt{(\vec{x}^{\mathcal{C}}_i - \vec{\mu}^{\mathcal{C}}_b)^\top (\vec{x}^{\mathcal{C}}_i - \vec{\mu}^{\mathcal{C}}_b)} \tag{2.16}$$

$$E_{\text{pos}}(i,b) \propto \frac{1}{\bar{S}}\sqrt{(\vec{x}^{\mathcal{S}}_i - \vec{\mu}^{\mathcal{S}}_b)^\top (\vec{x}^{\mathcal{S}}_i - \vec{\mu}^{\mathcal{S}}_b)} \tag{2.17}$$

The superscripts $\mathcal{C}$ and $\mathcal{S}$ denote the subvectors of the (mean) feature vectors including only the color- and spatial-dependent components, respectively. To ensure that the result is independent from the image and superpixel resolution, the spatial distance is normalized by the average superpixel edge length $\bar{S} = \sqrt{N/|\boldsymbol{k}|}$. The user-defined parameter $\tilde{\alpha}$ addresses the different scaling of the color and spatial domain. It is selected from an interval of $\tilde{\alpha} \in [0,\infty)$. Selecting a higher $\tilde{\alpha}$ results in more compact superpixels, while a lower $\tilde{\alpha}$ increases the sensitivity to fine-grained image structures. As the selection of the value for $\tilde{\alpha}$ in Equation (2.15) is not intuitive, an alternative assignment energy term is defined in [74]

$$E^{\text{slic}}_{\text{asm}}(i,b) = (1-\alpha)E_{\text{col}}(i,b) + \alpha E_{\text{pos}}(i,b). \tag{2.18}$$

In contrast to the open interval of $\tilde{\alpha}$, the new parameter α has a closed interval of $\alpha \in [0,1]$. A visual comparison of different values of the spatial weight α can be found in Figure 2.7.

In order to minimize the objective function (2.11) with the new assignment energy term (2.18), the iterative optimization scheme of k-means described in Section 2.1.2 can be utilized. This involves again the assignment of each pixel to the cluster center with the lowest assignment energy and a subsequent update of the cluster centers represented by the mean feature vectors. But while for general k-means

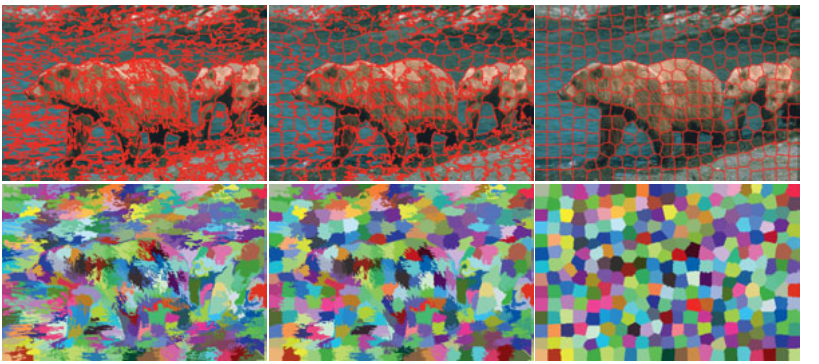


Figure 2.7: Comparison of label maps and boundary images with different values for α and approximately 300 superpixels. From left to right α takes the values 0.5, 0.9 and 0.99.

clustering the assignment energy for all pixel and cluster center combinations has to be evaluated, the effort can be drastically reduced in the case of a superpixel segmentation [1]. As the approximate spatial extent of a superpixel is known a priori, only a subset of possible cluster centers have to be checked for each pixel to find the minimal assignment energy. This is because for many pixels it is highly unlikely that they are assigned to a center which is far away on the image plane. For example, a pixel from the top-left corner of an image is highly unlikely to be part of a superpixel in the lower-right corner. Hence, the authors of [1] propose to limit the search space by a window spanned around the cluster centers in the spatial image domain and only evaluate the assignment energies for the pixels which are inside this window. This leads to a significant reduction of the computational complexity as only a fraction of the original distance calculations have to be performed. Simultaneously, the impact on the final superpixel segmentation is negligible. In [1] the edge length of the search window was selected to be two times the average superpixel edge length $\bar{S}$. The reduced search space in comparison to the search of general k-means is illustrated in Figure 2.8.

Although the spatial compactness of the segments is improved, due to the usage of the five dimensional feature space, the spatial coherence of the superpixels is not guaranteed yet. This is because again no neighborhood relations between the pixels are considered during the *assignment*-step of SLIC. To ensure the coherence of the final superpixel segmentation, Achanta et al. [1] propose a postprocessing step that is executed after the last iteration is performed. In the postprocessing, all single pixels

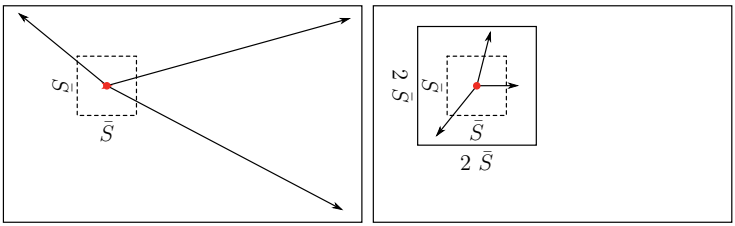


Figure 2.8: Principle of the search space limitation of the SLIC superpixel approach. The search space for each cluster center (red dot) in standard k-means (a) reaches over all pixels in the image plane (indicates by the arrows). In case of the SLIC superpixel algorithm (b), only a limited neighborhood around each superpixel center has to be searched (denoted by the square with edge length $2\bar{S}$). This is because pixels which have a large distance to a superpixel center on the image plane are unlikely to be part of this superpixel.

and pixel groups, which are not connected with the main-mass of the corresponding superpixel, are identified. In a subsequent step, these pixels are reassigned to the first directly neighboring superpixel discovered in a greedy search.

While this approach successfully ensures the spatial coherence of the superpixels, it does not fully obey the optimization target, as it does not minimize the objective function (2.11) during the postprocessing. For this purpose, Schick et al. [74] propose to exchange the *assignment*-step of SLIC with a more localized contour-based approach. In this approach, only pixels at the boundary between superpixels are considered to be flipped between the superpixels during the *assignment*-step of the iterative optimization. To ensure the spatial coherence of the superpixels, it has to be checked whether a flip breaks the spatial coherence of one of the involved superpixels. If only those flips are executed which conserve the spatial coherence a valid superpixel segmentation is created while obeying the objective target in every step of the algorithm. To efficiently check if a pixel breaks the spatial coherence, techniques emerged from the field of digital topology can be utilized. These were first formulated by Rosenfeld in [68] and will be described below.

The following description will first focus on the spatial coherence of a single superpixel. Later, the method will be extended to multiple superpixels. Initially, it will therefore be assumed that the currently not regarded superpixels are merged to a single background segment. Thereby, creating a binary segmentation where the currently regarded superpixel becomes the foreground and the surrounding superpixels

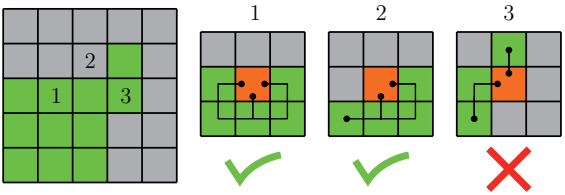


Figure 2.9: Left square: visualization of a pixel neighborhood with a simply connected foreground region colored in green. Right squares: close-ups of the three marked pixels with the currently regarded pixel colored in orange. The two left close-ups show valid simple points with respect to the green foreground region, as only a single 4-connected foreground region (marked by the lines with the dotted ends) is adjacent to the central pixel in the 4-connected sense. The example on the right is not a simple point, as here two separate 4-connected foreground components are adjacent to the regarded pixel.

make up the background. As the regarded superpixel is a 4-connected component without holes, it is called *simply connected* [68]. It should be noted that while the foreground is a 4-connected component the background can be 8-connected. A pixel whose label can be flipped (from background to foreground or vice versa), without breaking the topology of the foreground component, is called a *simple point* [68]. In order to keep the regarded superpixel spatially coherent while flipping the boundary pixels, it is therefore sufficient to ensure that only simple points are added or removed from its connected component. In the following, it is described how the simple points can be identified efficiently.

Flipping an interior pixel of the foreground to the background label would introduce a hole into the foreground component. As a result its topology would change. Therefore, no interior pixel can be a simple point and thus only boundary pixels have to be regarded. To efficiently decide if a boundary pixels is a simple point, it is sufficient to check the 3×3 neighborhood of the pixel for the following rule [68]: *A boundary pixel is a simple point if in its* 3×3 *neighborhood (excluding itself) there exists only a single connected foreground component that is adjacent to the regarded pixel.* Where adjacency and connectivity to the foreground component are understood in the 4-connected sense. An example with simple and non-simple points is visualized in Figure 2.9.

The validity of this simple check has been shown for the binary case in [68]. But it cannot be easily generalized to the multi-label case, as it was also noted in [14]. Instead it has to be checked sequentially if the currently regarded pixel is a simple

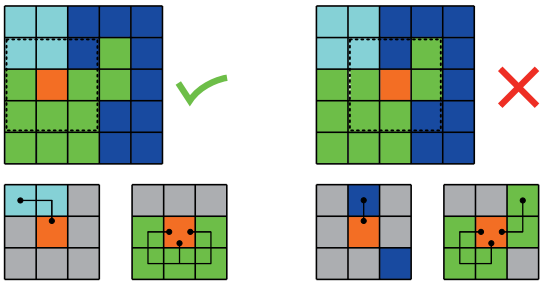


Figure 2.10: Two example cases for the determination of the simple point property of boundary pixels between superpixels. First row: the considered pixel (orange) and its 3×3 neighborhood (dotted box). Second row: the 4-connected components (indicated by the lines with the dotted ends) for the source and the target superpixel. The pixels of the currently not regarded superpixels are assigned a common background label (gray). In (a) the pixel is a simple point for both superpixels (cyan and green) and thus can be flipped between them. In (b) the pixel is a simple point for the blue superpixel but not for the green one and thus cannot be flipped.

point for all involved superpixels. As adding a pixel to a *target* superpixel always involves removing it from a *source* superpixel, the pixel has to be a simple point for the source as well as for the target superpixel. Otherwise, the spatial coherence of one of the involved superpixels could be broken up by the label flip. Only if the check is successful for both superpixels the pixel can be exchanged between them without breaking the spatial coherence. While checking for the simple point property the labels of all other superpixels are sequentially merged into a single background label, as it is shown in Figure 2.10.

By only exchanging boundary pixels with the simple point property between two superpixels, the approach proposed in [74] guarantees a spatially coherent superpixel segmentation. Simultaneously, the iterative optimization scheme obeys the objective function of Equation (2.11) at all times. As an initial condition the spatial coherence has to be guaranteed before performing the first iteration of the optimization scheme. To ensure this and to create a spatially homogeneous distribution of superpixels on the image plane, a suitable initialization is needed. In general, this can be achieved by choosing a grid-like structure with edge-length $\bar{S}$ or an equivalent honeycomb-structure. The number of superpixels to be created during the initialization has to be predefined by the user. In the following, an alternative oversegmentation method is described that derives the number of clusters automatically.

2.1.4 Mean Shift Superpixel Segmentation

The mean shift algorithm was originally proposed by Fukunaga and Hostetler in [29] and was later rediscovered by Cheng [17]. It describes a method to estimate the local gradient of a density function (mean shift vector) which can be used to seek the modes of the function. It is therefore used as a non-parametric technique for feature space analysis.

To estimate the gradient, a window $G(\vec{y})$ is defined in the feature space which is centered around the vector $\vec{y}$. The mean vector of the feature vectors $\boldsymbol{x} = \{\vec{x}_1,...,\vec{x}_N\}$ which fall into the range of the window is defined as

$$\vec{\mu}(\vec{y}) = \frac{\sum_{\vec{x}\in G(\vec{y})} K(\vec{x}-\vec{y})\vec{x}}{\sum_{\vec{x}\in G(\vec{y})} K(\vec{x}-\vec{y})}. \tag{2.19}$$

Where $K(\cdot)$ is a weighting kernel function of the window. The mean shift vector is then defined as the difference between the mean feature vector of the features inside the window and the window's center, i.e. $\vec{\mu}(\vec{y}) - \vec{y}$. By iteratively shifting the window by the mean shift vector and then reevaluating the vector, the mode of the density function of the samples is reached. A visualization of the principle of the mean shift algorithm can be found in Figure 2.11.

For a unimodal density distribution the algorithm is guaranteed to converge at the peak of the density distribution, where the gradient is zero [19]. The selected kernel function can either have a uniform weight or can be chosen to have decreasing weights (e.g. Gaussian) to reduce the influence of feature vectors closer to the window border. The window range as well as the form have an influence on the convergence properties of the algorithm. According to [19], the algorithm converges in a finite number of steps if a window with uniform weights is used. For a window with a weight that depends on the distance to the window center the algorithm is infinitely convergent. Hence, the algorithms should be terminated if the magnitude of the mean shift vector falls under a minimal threshold or a maximal number of iterations has been performed.

For the case of a multimodal distribution, the algorithm can be used to explore the amount of modes and their location by starting the algorithm at several different locations. As the center of each window follows the local gradient, all windows will end up in one of the multiple peaks. This property can be used to cluster the data by starting the algorithms at all data points and assigning each data point to the mode at which the window converges. In contrast to the previously discussed algorithms, the number of clusters cannot be predefined directly by the user. It can only be influenced by the selection of the window range and the shape of the kernel function. For the application of image segmentation this is first described by [18], where the CIELUV pixel values are used as feature vectors. All pixels for which the algorithms ends up in the same peak are assigned to the same image segment.

By keeping track of the data points traveled by each window the efficiency of the algorithm can be improved significantly, as the algorithm does not have to be started from the traveled pixel locations.

While in [18] solely the three dimensions of the the CIELUV color space were selected as feature space, the method was later refined in [19] by adding the two spatial dimensions of the image domain to the feature vectors. This leads to an oversegmentation with more compact segments, similar to the SLIC approach. To take into account the different scaling of the color and spatial domain, the single kernel function is exchanged against a composed kernel function $K_{\mathcal{R}_\mathcal{C},\mathcal{R}_\mathcal{S}}(\vec{x})$ which is defined as

$$K_{\mathcal{R}_\mathcal{C},\mathcal{R}_\mathcal{S}}(\vec{x}) = \frac{\mathcal{B}}{\mathcal{R}_\mathcal{C}\mathcal{R}_\mathcal{S}} K(\frac{\vec{x}^\mathcal{C}}{\mathcal{R}_\mathcal{C}}) K(\frac{\vec{x}^\mathcal{S}}{\mathcal{R}_\mathcal{S}}). \tag{2.20}$$

Here, $\mathcal{B}$ denotes a normalization constant. The two parameters $\mathcal{R}_\mathcal{C}$ and $\mathcal{R}_\mathcal{S}$ can be used to change the range in the different domains which results in different levels of segmentation granularity. Selecting a smaller range of the search window leads to higher resolution of the segmentation, i.e. more oversegmentation of the image. Similar to the SLIC approach, the mean shift algorithm does not guarantee that the produced segments are spatially coherent. Therefore, a postprocessing step is required, equal to the one described in Section 2.1.3. This step often additionally removes small segments which are below a user defined size. A visual comparison between the oversegmentations created by k-means, SLIC, and mean shift can be found in Figure 2.12. It can be seen that the size of the segments created by k-means as well as the mean shift algorithm are very inhomogeneous. Additionally, the segments produces by k-means are spatially incoherent. In contrast to that, the SLIC algorithm produces image segments of homogeneous size and shape which is preferred for many applications as it can be beneficial for subsequent processing steps [47, 60, 85]. The SLIC algorithm is therefore chosen as a foundation for this work and will be extended to the case of video oversegmentation in Chapter 3.

2.2 Optical Flow

An important tool used in many computer vision approaches is the optical flow. It can be described as the displacement vector of a point occurring in two different frames which depict the same scene at two different instances of time. The movements of the points can either be initiated by the movement of an object in the scene or by a movement of the camera. In general, it can be distinguished between two forms of optical flow which are both visualized in Figure 2.13.

First, there is the dense optical flow. Here, for each pixel of an image a displacement vector is available except for pixels where no corresponding pixel exists in the other frame. These situations can occur if a pixel gets occluded between the two frames or a pixel only appears in the second frame because it gets disoccluded. A

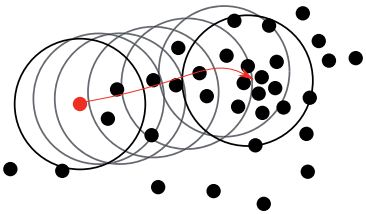


Figure 2.11: Principle of the mean shift algorithms for a 2D feature space with a unimodal distribution. The initial window is centered around the data point marked in red. By iteratively following the mean shift vector, the algorithm finds its way to the peak of the feature vector density function.

popular method to calculate dense optical flow is the approach proposed by Horn and Schunck in [38] which will be further explained in Section 2.2.1.

The second form is sparse optical flow. Here, only for a subset of pixels a displacement vectors is available. Often the pixels, where optical flow is available, coincide with visible corners in the image, as they can be tracked more reliably over successive frames. A method that is used frequently in the literature is the combination of selecting corners using [75] and tracking them to the next frame using [78]. This combination is often referred to as Kanade-Lucas-Tomasi feature tracker (KLT) and it will be further described in Section 2.2.2. It was experimentally shown in [5] that the KLT method tends to be more robust against image noise and quantization errors than the approach of Horn and Schunck. Therefore, an approach to interpolate a dense optical flow field from the sparse flow vectors of KLT will be described in 2.2.3. An alternative approach, proposed in [12], which combines the perks of KLT with the approach of Horn and Schunck will be summarized in 2.2.4.

2.2.1 Horn-Schunck-Method

The method proposed by Horn and Schunck in [38] is based on the brightness constancy assumption which can be expressed as follows

$$I(x,y,t) = I(x + dx, y + dy, t + dt). \tag{2.21}$$

Here, $I(x,y,t)$ is the image function at position $[x,y]^\top$ at time t. The change in time is denoted by dt while the displacement in x and y direction is denoted by dx and dy, respectively. Using a Taylor series approximation Equation (2.21) can be rewritten as follows

$$I(x,y,t) = I(x,y,t) + \frac{\partial}{\partial x} I(x,y,t)dx + \frac{\partial}{\partial y} I(x,y,t)dy + \frac{\partial}{\partial t} I(x,y,t)dt + e, \tag{2.22}$$

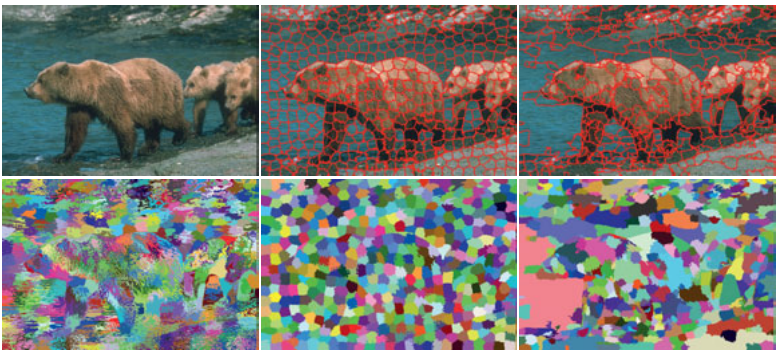


Figure 2.12: Segmentation examples for three different clustering-based algorithms (from left to right): k-means [52], SLIC [1], and mean shift [19]. Top row: segmentation boundaries. Bottom row: label maps. Instead of boundaries for the k-means algorithm, only the original image is shown, as the spatially not coherent segmentation would result in too many boundaries to display them meaningful. The number of segments for k-means and SLIC was selected to match the number of segments produced by the mean shift algorithm. It should be noted that the segments generated by the SLIC algorithm are much more compact than the results of the mean shift algorithm.

where e is the portion of higher order terms which will be neglected in the following. For clarity, the partial derivatives of the image function in x,y and t direction will be denoted as I_x, I_y and I_t, respectively. By further omitting the parameters of the derivatives, the following equation is obtained

$$I_x u + I_y v + I_t = 0. \tag{2.23}$$

Here, the displacement vector is denoted by $[u,v]^\top$, the time change between two frames is redefined to 1, and the position as well as the time indices were dropped for the purpose of clarity.

As this equation with two unknowns cannot be solved without further constraints, an additional constraint was introduced in [38] which enforces the optical flow field to be smooth over the image domain Ω. The constrained optimization problem can be solved by minimizing the following error functional

$$E_{\mathrm{hs}}(u,v) = \int_\Omega (I_x u + I_y v + I_t)^2 + \lambda_{\mathcal{F}}(|\nabla u|^2 + |\nabla v|^2) d\Omega. \tag{2.24}$$

This least square approach minimizes the sum of squared errors made in the brightness constancy assumption and the change in the flow field represented by the partial

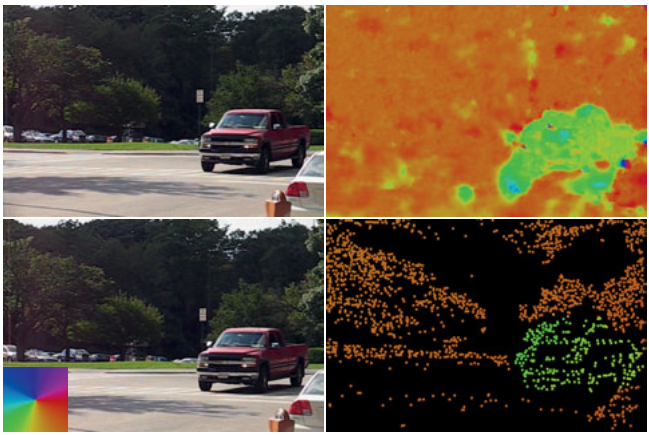


Figure 2.13: Left column: two consecutive frames of a video sequence. Right column: visualization of the different forms of optical flow. The upper image shows a dense optical flow field and the lower image shows a sparse flow field. The direction and amount of the displacement are encoded in the color and saturation, respectively. The utilized color coding is shown in the small square in the bottom left corner.

derivatives. The weighting factor $\lambda_{\mathcal{F}}$ regularizes the trade-off between smoothness of the flow field and compliance with the brightness constancy assumption. To minimize the error functional, the corresponding Euler-Lagrange equations can be utilized which are given by

$$(I_x u + I_y v + I_t) I_x + \lambda_{\mathcal{F}}(\Delta u) = 0 \tag{2.25}$$
$$(I_x u + I_y v + I_t) I_y + \lambda_{\mathcal{F}}(\Delta v) = 0. \tag{2.26}$$

Where Δv and Δu are the Laplacian of the flow field components. In the discrete case these can be approximated by $\Delta u \approx u - \bar{u}$, where $\bar{u}$ is the average of the u component of the displacement vector in a small neighborhood. By inserting the approximation and rearranging the terms the following equations are obtained

$$\begin{aligned} u &= \bar{u} - I_x \frac{\widetilde{P}}{\widetilde{Q}} \\ v &= \bar{v} - I_y \frac{\widetilde{P}}{\widetilde{Q}} \end{aligned} \quad \text{with} \quad \begin{aligned} \widetilde{P} &= I_x \bar{u} + I_y \bar{v} + I_t \\ \widetilde{Q} &= \lambda_{\mathcal{F}} + I_x^2 + I_y^2. \end{aligned} \tag{2.27}$$

As the displacement vector components still depend on the average of the small neighborhood, an iterative scheme starting from an initial guess can used to gradually approach a locally optimal solution. This can be seen as a form of the Jacobi

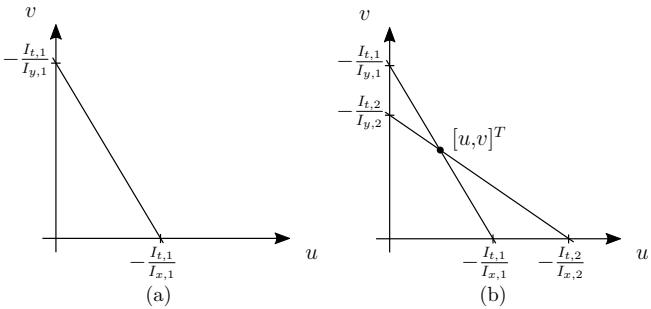


Figure 2.14: Visualization of the aperture problem. Left: a single line for a feature point is derived from Equation (2.23). Due to the aperture problem, it can only be said that the flow vector of this feature point lies somewhere on the drawn line. Right: Given a second pixel with assumed identical displacement vector, but different image gradients, a second line can be drawn. The intersection of the two lines is the optical flow vector shared by the two points.

method. In each iteration z a new optical flow vector $\vec{\omega}_z = [u_z,v_z]^\top$ is calculated using the average vector from the result of the last iteration $[\bar{u}_{z-1},\bar{v}_{z-1}]^\top$. The processing is stopped when convergence is reached, i.e. $||\vec{\omega}_z - \vec{\omega}_{z-1}||$ falls below a certain threshold for all pixels of the image.

2.2.2 Lucas-Kanade-Method

A second, very popular technique is to select feature points in an image using the method described in [75] and to compute the displacement vectors of the selected points between two frames using the method proposed by Lucas and Kanade in [53]. Together both methods are often referred to as KLT.

The basic principle of the method is depicted in Figure 2.14. For each feature point a line can be plotted into the 2D space spanned by the two variables of the displacement vector by successively setting the u and v variables in Equation (2.23) to zero. Due to the so-called aperture problem, it can only be said that the displacement vector $[u,v]^\top$ of the feature point lies somewhere on the line plotted in Figure 2.14(a). To solve this ambiguity, it is assumed in [53] that the displacement vectors of the pixels in a small neighborhood around the feature points are constant. Thus, an additional line from a neighboring pixel can be added to the uv-space which results in a cut of the two lines at the position of the shared $[u,v]^\top$ vector, as it can be seen in Figure 2.14(b).

But as image data is in general contaminated with some sort of noise, adding a

third and forth line of different neighboring pixels will not result in a cut at the same position. Hence, multiple neighbors are considered at once and the optimal solution in the least square sense is calculated. Following [12], the error function for the least-squares solution can be formulated as

$$E_{\mathrm{lk}}(u,v) = K_\rho * ((I_x u + I_y v + I_t)^2). \tag{2.28}$$

Where $*$ is the convolution operator and K_ρ is a Gaussian kernel with a standard deviation of ρ. By setting the partial derivatives with respect to u and v of Equation (2.28) to zero the following linear equation system is obtained

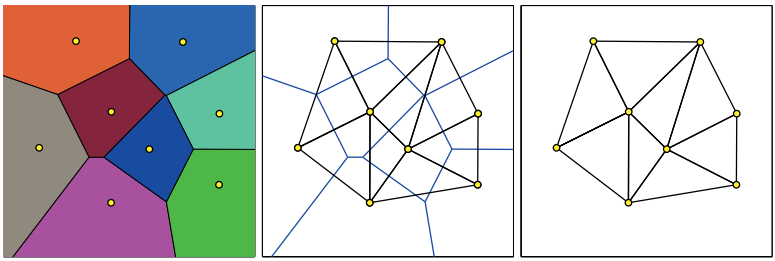
$$\begin{bmatrix} K_\rho * I_x^2 & K_\rho * (I_y I_x) \\ K_\rho * (I_y I_x) & K_\rho * I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -K_\rho * (I_x I_t) \\ -K_\rho * (I_y I_t) \end{bmatrix}. \tag{2.29}$$

The optical flow vector for an image patch can be determined by calculating the pseudo-inverse of the system matrix and multiplying it from the left side.

Due to the first order Taylor-approximation, the validity of Equation (2.22) only holds for small displacement vectors. To cope with image sequences with arbitrary motion a coarse-to-fine strategy can be used (cf. e.g. [2, 7]). Here, the origin and the target image are smoothed and subsampled multiple times before the displacement vectors are calculated for each scale starting from the coarsest level. Before the vectors for the next finer scale are calculated the origin image is warped according to the displacement vectors of the coarser scale. This strategy can be used to improve the results of [53] as well as [38].

2.2.3 Sparse-to-Dense Optical Flow Conversion

As the optical flow vectors created by the KLT method are often more reliable than the flow fields created by the Horn and Schunck approach (as e.g. shown in [5]), it can be desirable to convert the sparse optical flow vectors of KLT into a dense flow map. A standard method, as it is e.g. used by [62], is the creation of a triangle mesh between the tracked feature points and a linear interpolation of the flow vectors inside the triangles using Barycentric coordinates. The mesh is often created through a Delaunay triangulation [20] as it maximizes the minimal angle of all triangles and thus avoids sliver triangles. The triangulation can either be derived from the Voronoi diagram of the feature points, as it is shown in Figure 2.15, or by an approach such as the flip algorithm [43]. The former method connects all tracked points with an edge whose Voronoi cells share a common boundary. The latter first creates an arbitrary non-overlapping triangulation. Subsequently, for all triangles it is checked whether the circle that connects all three vertices contains any other vertex that is part of a neighboring triangle. If the condition is true for any triangle an edge flip is performed. In the flip the common edge is deleted and the two previously unconnected vertices are connected by a new edge, as it can be seen in Figure 2.16.



(a) (b) (c)

Figure 2.15: Duality between the Voronoi diagram and the Delaunay triangulation. In (a), the colored Voronoi cells of the clustering of Figure 2.6(d) are shown. In (b), the Voronoi diagram and the matching Delaunay triangulation are depicted as an overlay. In (c), only the triangulation is shown.

2.2.4 Lucas/Kanade meets Horn/Schunck

While the technique described in the previous section is a fast and efficient way to convert sparse flow vectors into a dense flow map, it e.g. does not obey motion boundaries. These can occur if different image parts move into different directions. At these occasions only a smooth transition between the directions is provided. To better handle these occasions, it is proposed by Bruhn et al. [12] to combine the approach of Horn and Schunck and KLT into a hybrid method. The new method combines the robustness of the local KLT method with the creation of dense flow maps using the global approach of Horn and Schunck. Simultaneously, it prevents the overly smoothed transitions at motion boundaries as they can be observed in interpolation techniques as well as the original Horn and Schunck method. To achieve this the Equations (2.28) and (2.24) are reformulated into

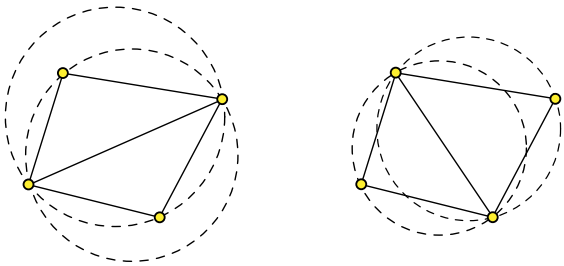
$$E_{\mathrm{lk}}(\vec{\omega}) = \underline{\vec{\omega}}^{\top} J_{\rho}(\nabla_3 I)\underline{\vec{\omega}} \tag{2.30}$$

$$E_{\mathrm{hs}}(\vec{\omega}) = \int_{\Omega} (\underline{\vec{\omega}}^{\top} J_0(\nabla_3 I)\underline{\vec{\omega}}) + \lambda_{\mathcal{F}}(|\nabla\underline{\vec{\omega}}|^2) d\Omega \tag{2.31}$$

using the flow vector in homogeneous coordinates $\underline{\vec{\omega}} = [u,v,1]^{\top}$ and the following notations:

$$|\nabla\underline{\vec{\omega}}|^2 := |\nabla u|^2 + |\nabla v|^2$$
$$\nabla_3 I := [I_x, I_y, I_t]^{\top}$$
$$J_{\rho}(\nabla_3 I) := K_{\rho} * (\nabla_3 I \nabla_3 I^{\top}).$$

Here, K_{ρ} again denotes the Gaussian kernel with standard deviation ρ. In this form it can be seen that the first part of Equation (2.31) is equivalent to the Lucas-Kanade



(a) (b)

Figure 2.16: Example of an edge flip between two triangles. In (a), the Delaunay condition is not fulfilled, as the circles connecting all three vertices of the triangles (dotted circles) do contain another vertex that is not part of the triangle. In (b), the edge flip has been performed by deleting the common edge of two triangles and connecting the two previously not connected vertices.

error function (2.30) with $\rho = 0$. Hence, it was proposed by [12] to plug in the original error function (with $\rho \neq 0$) into (2.31) to obtain the combined local-global method with the following error function

$$E_{\text{clg}}(\vec{\omega}) = \int_{\Omega} \vec{\omega}^{\top} J_{\rho}(\nabla_3 I)\vec{\omega} + \lambda_{\mathcal{F}}(|\nabla\vec{\omega}|^2)d\Omega. \tag{2.32}$$

Equivalent to Section 2.2.1, this error function can be minimized by solving the corresponding Euler-Lagrange equations which are given by

$$K_{\rho} * I_x^2 u + K_{\rho} * (I_x I_y)v + K_{\rho} * (I_x I_t) - \lambda_{\mathcal{F}}(\Delta u) = 0 \tag{2.33}$$
$$K_{\rho} * I_y^2 v + K_{\rho} * (I_x I_y)u + K_{\rho} * (I_y I_t) - \lambda_{\mathcal{F}}(\Delta v) = 0. \tag{2.34}$$

The solution can either be derived by using the Jacobi method, in a similar way as in 2.2.1, or using another gradient decent technique, like successive over-relaxation (SOR) as described in [12]. To improve the robustness of their approach, the authors propose several extension of their basic algorithm. First, they convert the quadratic optimization problem of (2.32) into a non-quadratic one using the the following error function

$$E_{\text{clg,nq}}(\vec{\omega}) = \int_{\Omega} \Psi_1(\vec{\omega}^{\top} J_{\rho}(\nabla_3 I)\vec{\omega}) + \lambda_{\mathcal{F}}\Psi_2(|\nabla\vec{\omega}|^2)d\Omega. \tag{2.35}$$

Where $\Psi_i(\cdot)$ with $i \in 1{,}2$ are non-quadratic penalty functions like the one proposed by Charbonnier et al. in [15]:

$$\Psi(x^2) = 2\gamma^2\sqrt{1 + \frac{x^2}{\gamma^2}}. \tag{2.36}$$

The parameter γ influences the scale and the growth of the penalty. The usage of a non-quadratic optimization tends to produce more robust results as outliers are penalized less, thus reducing their influence on the resulting optimum. Second, they incorporated the spatiotemporal Gaussian kernel $K_{\rho,\chi}$, proposed by [6]. Here, ρ and χ are the spatial and temporal standard deviation, respectively. For the corresponding Euler-Lagrange equations and their solution by using SOR the reader is referred to [12].

Chapter 3

Superpixels with Temporal Consistency

This chapter describes a new system for creating temporally consistent superpixels from video sequences. To give an overview of the proposed system, Section 3.1 outlines the basic principle and structure of the approach. A more detailed description of the subsystems can then be found in the following subsections as well as the next chapter.

3.1 System Overview

The general structure of the proposed framework is depicted in Figure 3.1. First, the incoming video frames are converted into the representation utilized by the framework and dense optical flow vectors are extracted between the current and the previous frame. The selected representation as well as the feature space setup are described in Section 3.2. Subsequently, a cluster analysis is performed on the feature vectors by using a contour evolving energy minimization framework that is described in Section 3.3.

As the length of video sequences can vary significantly – from seconds to hours or even days – the video can in general not be processed as a whole due to memory limitations of the computing system. Therefore, the extent of the cluster analysis is constrained by using a time limiting window whose structure is described in Section 3.4. The window acts similar to a buffer following the first-in, first-out (FIFO) principle. But in contrast to a general FIFO buffer, several frames have to be added to fill the window, before the first frame can be read out. This leads to a small temporal delay of F frames which is inherent in the proposed framework. But on the other hand, the window enables the processing of arbitrary long sequences and gives the framework streaming capabilities.

When a new frame is added to the system, its segmentation needs to be initialized. As image regions can move significantly from frame to frame, a simple copy of the previous segmentation, as proposed in [82], can be error-prone in many situations. Therefore, the segmentation needs to be warped (propagated) to roughly fit the con-

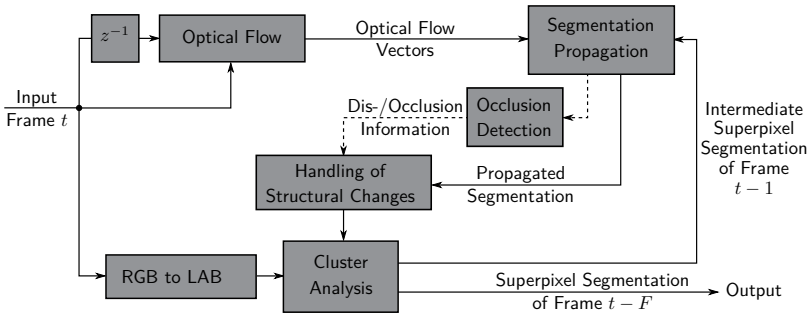
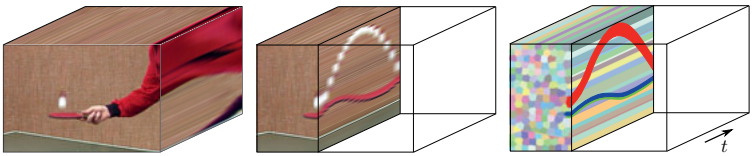


Figure 3.1: System overview: The video frames are converted to the CIELAB color space and optical flow is extracted. The flow vectors are utilized to propagate the previous segmentation onto the new incoming frames. Optionally, the propagation is simultaneously utilized to detect information about the occlusion and disocclusion of image regions that is used in the further cluster analysis to terminate and create new superpixels.

tent of the new frame. In this thesis, two separate approaches for the segmentation propagation are proposed which are described in the first part of Chapter 4. To ensure homogeneously sized superpixels, the superpixel segmentation needs to be adapted to structural changes in the video volume induced by occlusion and disocclusion of image regions. Therefore, this thesis proposes two novel approaches to explicitly handle these structural changes. Both methods decide on which superpixels are to be terminated and where new superpixels have to be created to cope with the structural changes. The first approach is purely superpixel size-based and can be combined with any propagation method. The second utilized dis-/occlusion information exclusively collected by only one of the propagation methods. The dis-/occlusion detection as well as the methods to handle of the structural changes are described in the second part of Chapter 4.

3.2 Feature Space Setup

As it has been shown in 2.1.3, the segmentation of an image into superpixels can be seen as a clustering problem, where feature vectors are divided into clusters and the cluster affiliation is equivalent with the membership to a superpixel. The single image superpixel approaches proposed in [1] and [74] treat each pixel as a five-dimensional (5D) feature vector that combines the position of the pixel in the color space with its spatial position. When moving to the video domain, the tem-



(a) (b) (c)

Figure 3.2: In (a), the video volume of the *tennis* sequence [85], showing a table tennis ball being played up and down, is depicted. In (b), a longitudinal section of the video volume at approximately the position of the table tennis ball is shown. (c) shows the ideal segmentation of the video volume. The red segment represents the ball and the blue and green segments the bat. Each segment follows its underlying image region over time throughout the volume. The segments representing the static background and the table ideally hold their constellation and shape over time.

poral position of the pixel is added to its feature vector to distinguish the pixels of different frames. Thus, here each pixel i is described by its feature vector $\vec{x}_i = [l_i,a_i,b_i,x_i,y_i,t_i]^\top$ where $\vec{x}_i^{\mathcal{C}} = [l_i,a_i,b_i]^\top$ are the color values of the pixel in CIELAB space, $\vec{x}_i^{\mathcal{S}}\,[x_i,y_i]^\top$ is its position in image coordinates and t_i is the frame index. The choice of the color space is made by following the argumentation of [19] and [1] to use a perceptually uniform color space such as the CIELAB or CIELUV space. As neither of the color spaces has a clear advantage in practice over the other, as stated in [19], the CIELAB space is chosen for all experiments because it was also utilized in the original superpixel approach of [1].

The requirements for temporally consistent superpixels that where already stated in Section 1.1 are tripartite. First, a temporally consistent superpixel should group pixels of similar color. Second, the groups should be spatially compact on a frame level. And third, the temporally consistent superpixel should follow their underlying image region through the video volume as it is illustrated in Figure 3.2.

The first and second requirement are equivalent with the requirements for a single image superpixel approach. As the pixels of a superpixel will have a similar color value and spatial position, their feature vectors will form clusters in the feature space consisting of color and image coordinate dimensions. Hence, for a single image approach each superpixel can be represented by its mean feature vector in the color dimension as well as in the spatial dimension as done by [1] and [74].

In addition to their single image superpixel approach, the authors of [1] also describe an extension in their paper which groups the voxels of a video volume into supervoxels. The approach extends the assumptions about the spatial position and distribution of the color values of an image region into the temporal domain. There-

fore, each supervoxel is represented by the mean vector of the six-dimensional (6D) feature vector. This essentially means the introduction of a single spatial center and a single temporal center for each supervoxel which has two major drawbacks with respect to the requirements of temporally consistent superpixel that will be further discussed in the following.

First, the approach prefers temporally compact supervoxels as it explicitly penalizes supervoxels with a long duration. This is due to the selected energy term in the optimization procedure. It follows the scheme described in Section 2.1.3 but adds a temporal energy term that is proportional to the temporal distance between a voxel and the temporal center of a supervoxel. Similar to the spatial energy, the temporal term is weighted to take into account the different scaling of the feature dimensions. Nevertheless, it enforces the optimization to prefer temporally close voxels over temporally distant voxels. Thereby, the creation of temporally compact supervoxels is implicitly encouraged. In particular, for applications like object tracking in surveillance videos this property is undesirable as each spatio-temporal segment should occupy the underlying region, as long as the region is visible.

A second drawback of the supervoxel approach, proposed in [1], is the usage of a single spatial center for the representation of each supervoxel. As can be seen in Figure 3.3, the underlying assumption that the pixels of a spatio-temporal segment are close to a common spatial center breaks if moving objects are considered as e.g. the table tennis ball in the depicted scene. The three figures in the top row show three example frames from the video sequence. In the schematic frames of the second and third row the position of the ball is indicated by the dotted circle. In the second row the spatial mean position of the ball in the y dimension is indicated by a dotted line. For the supervoxel approach of [1], this line would also represent the y position of the spatial cluster center of the supervoxel corresponding to the spatio-temporal region of the ball. As the voxels of the ball have a large distance to the spatial center, the spatial weighting term $\widetilde{\alpha}$ in Equation (2.15) has to be reduced. This lessens the influence of the spatial distance and efficiently increases the spatial range from which voxels can be assigned to the supervoxel. This is indicated by the area of the circle shaded in gray in Figure 3.3. Simultaneously, the reduction of the spatial weight also leads to spatially less compact supervoxels, as the spatial range is increased symmetrically in all spatial directions. Thus, it becomes more likely for spatially far away voxels to be assigned to a segment if their color is similar. An alternative would be to use several supervoxels to represent the trajectory of the ball. Both options contradict the requirements for temporally consistent superpixels and produce undesirable results, i.e. non-compact superpixels or short temporal superpixel trajectories. To overcome these drawbacks, it is proposed in this work to avoid the introduction of a cluster center on the temporal axis but only introduce centers in the color and the spatial domain.

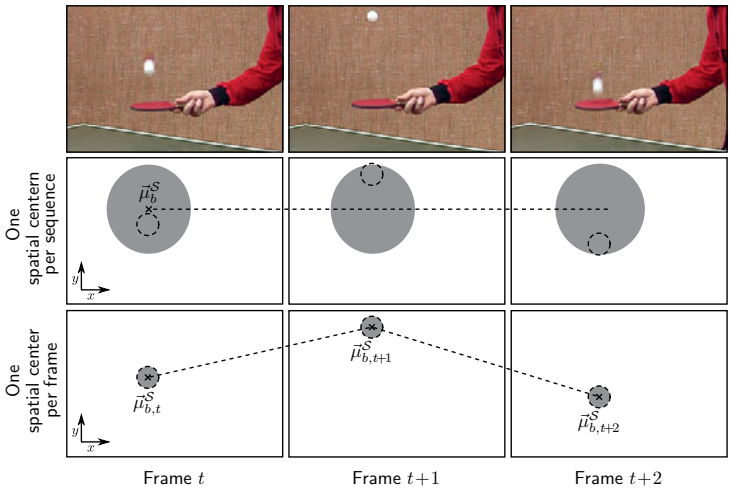


Figure 3.3: Top row: example frames from the table tennis sequence. Middle row: a segment (gray shaded) with one spatial center for all frames is created to cover the outline of the table tennis ball (dotted circle). Assuming a concentric extent of the segments around the spatial center, the segment has to grow in size to cover the ball. Bottom row: when for each frame a separate spatial center is used the segment can match the outlines of the many positions of the ball without changing its size.

The pixel color values of a uniformly colored region will form a dense cluster in the color space. In the following, it is assumed that the appearance of an object is – in a first approximation – almost constant over the course of a video. Hence, the pixel color values of the same region (visible in multiple frames) will be part of the same cluster in the color space. It should be noted that the color values of different frames are assumed to be located in the same instance of the CIELAB color space. Therefore, each temporally consistent superpixel can be represented by a single color centroid, globally valid for all frames. In contrast to that, the spatial distribution of the pixels of a temporally consistent superpixels may vary significantly from frame to frame. In particular, if the object is exposed to large displacements along the image plane as it is shown in Figure 3.3. Hence, to be able to capture arbitrarily moving objects, the spatial centroids are selected to be different on each frame and each of them is only locally valid on one frame. Thereby, assuming a different instance of the xy-space for each frame. This principle is depicted in the third row of Figure 3.3.

As a consequence, in a frame t each temporally consistent superpixel b is modeled by a vector $\mu_{b,t} = [\vec{\mu}_b^{\mathcal{C}}, \vec{\mu}_{b,t}^{\mathcal{S}}]^\top$ which includes the color mean value $\vec{\mu}_b^{\mathcal{C}}$ of all frames and a spatial position mean value $\vec{\mu}_{b,t}^{\mathcal{S}}$ for each of the frames. The latter preserves the spatial locality on frame level and the former ensures temporal consistency. The global color cluster center and the local spatial centers of a temporally consistent superpixel b can be determined as follows

$$\vec{\mu}_b^{\mathcal{C}} = \frac{1}{\sum_{t=1}^{T} |\boldsymbol{n}_{b,t}|} \sum_{t=1}^{T} \sum_{i \in \boldsymbol{n}_{b,t}} \vec{x}_i^{\mathcal{C}} \tag{3.1}$$

$$\vec{\mu}_{b,t}^{\mathcal{S}} = \frac{1}{|\boldsymbol{n}_{b,t}|} \sum_{i \in \boldsymbol{n}_{b,t}} \vec{x}_i^{\mathcal{S}}. \tag{3.2}$$

Here, $\boldsymbol{n}_{b,t}$ is the set of pixels assigned to the temporally consistent superpixel b in frame t. In general, the assignment of the pixels to the temporally consistent superpixels is unknown a priori and has to be determined following an optimization criteria which will be described in the following section.

3.3 Hybrid Optimization

The label set of a video volume contains the cluster assignments of all pixels of the volume and is denoted by $\boldsymbol{b} = \{b_i | \forall\, i \in \boldsymbol{n}_t, t \in \{1,...,T\}\}$. Here, $\boldsymbol{n}_t$ is the set of all pixel positions in frame t and the labels b_i come from the set of labels of all temporally consistent superpixels $\boldsymbol{k}$. From a given labeling of all pixels $\boldsymbol{b}$, the set of color cluster centers $\boldsymbol{\mu}^{\mathcal{C}} = \left\{\vec{\mu}_b^{\mathcal{C}} | \forall\, b \in \boldsymbol{k}\right\}$ and the corresponding spatial cluster centers $\boldsymbol{\mu}^{\mathcal{S}} = \{\vec{\mu}_{b,t}^{\mathcal{S}} | \forall\, b \in \boldsymbol{k}, t \in \{1,...,T\}\}$ can be easily derived using the Equations (3.1) and (3.2). The total assignment energy for the labeling $\boldsymbol{b}$ can then be calculated as follows

$$E_{\text{tcs}}(\boldsymbol{b}) = \sum_{t=1}^{T} \sum_{i \in \boldsymbol{n}_t} E_{\text{asm}}^{\text{tcs}}(i,t,b_i)\,. \tag{3.3}$$

To evaluate the assignment of a pixel i in frame t to a cluster b, the color and image coordinate dependent energies have to be added. Thereby, the same weighting scheme as defined by [74] is used, as it weights the color and spatial energy terms with the more intuitive factor $\alpha \in [0,1]$:

$$E_{\text{asm}}^{\text{tcs}}(i,t,b) = (1-\alpha) E_{\text{col}}^{\text{tcs}}(i,b) + \alpha E_{\text{pos}}^{\text{tcs}}(i,t,b). \tag{3.4}$$

Equal to the single image approaches described in Section 2.1.3, the energy terms are selected to be proportional to the Euclidean distance of the feature vector of the

pixel to the corresponding component of the mean feature vector of the superpixel

$$E_{\text{col}}^{\text{tcs}}(i,b) \propto \sqrt{(\vec{x}_i^{\mathcal{C}} - \vec{\mu}_b^{\mathcal{C}})^\top (\vec{x}_i^{\mathcal{C}} - \vec{\mu}_b^{\mathcal{C}})} \tag{3.5}$$

$$E_{\text{pos}}^{\text{tcs}}(i,t,b) \propto \frac{1}{S}\sqrt{(\vec{x}_i^{\mathcal{S}} - \vec{\mu}_{b,t}^{\mathcal{S}})^\top (\vec{x}_i^{\mathcal{S}} - \vec{\mu}_{b,t}^{\mathcal{S}})}\,. \tag{3.6}$$

Again, the Euclidean distance in the spatial domain is normalized by the edge length of the average sized superpixel to make the results independent from the image and superpixel resolution. It should be emphasized that the spatial energy $E_{\text{pos}}^{\text{tcs}}(i,t,b)$ is depending on the frame t in which the pixel i is located. Because the color energy for each temporally consistent superpixel is evaluated using its global color center and the spatial energy uses the local spatial centers, the optimization is referred to as a hybrid optimization scheme.

To create an optimal oversegmentation of a video volume, the labeling and the corresponding cluster centers have to be found which minimize Equation (3.3). In the general case, the problem of jointly finding the optimal cluster centers and the optimal labeling $\hat{\boldsymbol{b}}$ is NP-hard. But as for the single image approaches, described in Section 2.1.3, the optimization can be made computationally feasible by iteratively alternating between optimizing the labeling (*assignment*-step) and optimizing the cluster centers. Although this optimization scheme only reveals a locally optimal solution and heavily depends on the initial state of the labeling, the created temporally consistent superpixel segmentations are more than satisfying as will be shown in Chapter 5. To ensure the spatial coherence of the superpixels on frame level during the optimization, the same contour evolving optimization strategy as described in Section 2.1.3 is used when selecting the optimal label for each pixel. Thus, only label flips between neighboring superpixels are allowed that pass the test if a pixel is a simple point for the involved neighboring superpixels.

The starting point of the optimization is an initial labeling of the video volume $\boldsymbol{b}^{z=0}$ from which the joint cluster center set $\boldsymbol{\mu}^{z=0} = \{\boldsymbol{\mu}^{\mathcal{C},z=0}, \boldsymbol{\mu}^{\mathcal{S},z=0}\}$ can be derived. Again, the index z is utilized to denote the current iteration. In each new iteration $z+1$ the optimal labeling $\boldsymbol{b}^{z+1}$ is determined given the cluster centers of iteration z. The optimal labeling is derived by assigning each pixels to the neighboring superpixel – respecting the simple point test for all involved labels – for which the minimal assignment energy is created

$$\boldsymbol{b}^{z+1} = \underset{\boldsymbol{b}}{\operatorname{argmin}}\; E_{\text{tcs}}(\boldsymbol{b}|\boldsymbol{\mu}^z). \tag{3.7}$$

Subsequently, the cluster centers $\boldsymbol{\mu}^{z+1}$ are updated using the previously optimized labeling. By alternating these two steps, the energy is decreased with every iteration until a local optimum or another convergence goal is reached.

While the described optimization scheme could be applied on the entire video sequence at once, it is often not desirable and in most cases not reasonable to apply the framework on the whole video volume at once. First, the memory consumption will rise as more frames of the video are considered which in case of a full-length feature film would easily exceed the memory of a current desktop computer. Second, parts of the video could not be accessible at the start of the processing, e.g. in the case of live surveillance videos or video streaming services like Netflix. Third, as the optimization framework is prone to local optima, the initial constellation of the temporally consistent superpixels needs to be already roughly adapted to the video content to allow the superpixels to follow their underlying regions over time. This is not efficiently achievable in advance when significant object or camera movement as well as scene cuts are involved. And finally, it was assumed in Section 3.2 that the color of an image region is constant over time. Thereby, the color center of a temporally consistent superpixel would have to be globally valid over all frames of the video. In practical applications, this assumption might not hold in all situations especially when videos with natural lighting condition are regarded. Thereby, the selected model for the temporally consistent superpixels would not hold which could lead to failures in the segmentation. To circumvent the above mentioned cases, a windowing technique will be introduced in the next section.

3.4 Sliding Window

In order to restrict the number of video frames that have to be considered simultaneously during the optimization, a window is introduced which comprises W consecutive frames of the video volume. To optimize the complete video sequence, the window is shifted along the video volume frame by frame. Following each shift, a number of Z iterations of a slightly modified version of the hybrid optimization algorithm described in Section 3.3 is performed. As the window slides along the time axis during the optimization, it is called a sliding window. Limiting the number of simultaneously regarded frames drastically reduces the memory footprint and allows for a certain degree of scene changes, e.g. through gradual changes in illumination or color due to natural lighting conditions. Additionally, it gives the framework streaming capabilities as well as the possibility to dynamically adapt the constellation of the temporally consistent superpixels to structural changes inherent in the video volume. The first part of this section will discuss the adaption of the hybrid optimization to the sliding window technique. In the second part, the initialization of the sliding window will be described.

The modifications of the hybrid optimization affect the treatment of the frames inside the sliding window during the *assignment*-step. The sliding window contains P so called *past* frames, F so called *future* frames as well as a single *current* frame.

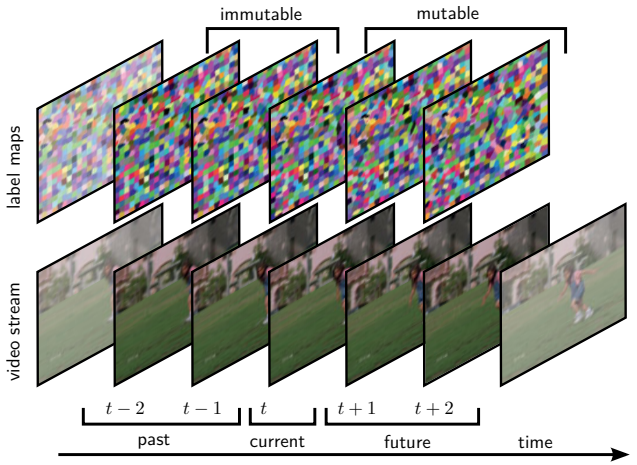


Figure 3.4: Sliding window approach. Bottom row: the video frames inside the sliding window (non-transparent) are divided into three groups (*past*, *current* and *future* frames). Top row: corresponding label maps. Only the *current* and *future* label maps are mutable and thus are altered during the optimization. Image taken from [63].

The total number of frames W is thereby given by $F+P+1$. An example with $W=5$ and $P=F=2$ is depicted in Figure 3.4. In this example, the frame t is the *current* frame and it is in the center of the sliding window. The sliding window is divided into mutable frames, i.e. the *current* and the *future* frames, and immutable frames, i.e. the *past* frames. During the *assignment*-step, only the contour pixels of the mutable frames are reassigned to the best matching neighboring superpixel to minimize the term (3.4). It should be noted that the sum of the total energy terms in Equation (3.3) now only comprises the frames inside the sliding window. When the cluster centers are updates, only the pixel values of the frames inside the sliding window are regarded. As the labeling of the *past* frames does not change anymore, the update of their spatial centers can be skipped. A summary of the hybrid optimization involving the sliding window can be found in Algorithm 1.

The position of the *current* frame is the last mutable position. When the sliding window is shifted and a frame leaves the position of the *current* frame to move into the range of the *past* frames, its segmentation becomes immutable and thus will not be altered anymore. Therefore, it can be said that the resulting final superpixel

input : W frames in sliding window around t; initial labeling $\boldsymbol{b}_t^{W,0}$
output: updated labeling $\boldsymbol{b}_t^{W,Z}$

determine parameters of color and spatial centers for given $\boldsymbol{b}_t^{W,0}$;
for $z \in [1, Z]$ **do**
  foreach *mutable frame* t' *in sliding window* **do**
    foreach *superpixel boundary pixel* i **do**
      if *pixel* i *is a simple point* **then**
        reassign pixel to superpixel which minimizes (3.4)
        given the cluster centers of $z - 1$;
      end
    end
  end
  forall *frames* t' *in sliding window* **do**
    if t' *is mutable frame* **then**
      update local spatial models in t';
    end
    accumulate global color information;
  end
  update global color centers from accumulated information;
end

Algorithm 1: Hybrid optimization of the segmentation inside a sliding window positioned around the current frame with index t. $\boldsymbol{b}_t^{W,z}$ denote the labeling of all pixels currently inside the sliding window at iteration z.

segmentation for this frame is generated at this point. But through the globally valid color centers the *past* frames still influence the segmentation in the *current* and *future* frames, as these are still mutable and thus their segmentation can change during the further optimization. The *future* frames help to adapt to changes in the scene, whereas the *past* frames are conservative and try to preserve the color value of the superpixels over time. If more *past* than *future* frames are used, the update of the color centers is more conservative. If more *future* than *past* frames are used, the update is more adaptive.

During the optimization, the boundaries of the temporally consistent superpixels in the mutable frames are aligned to the visible contours in the video frames. Nevertheless, the optimization is not able to comprehensively adapt the superpixel constellation in the case of large object movements or structural changes induced e.g. by the emergence of new objects. The former requires that the superpixel positions may have to differ significantly in consecutive frames. In some occasions, this can be too much to be handled successfully by the contour evolution. The latter

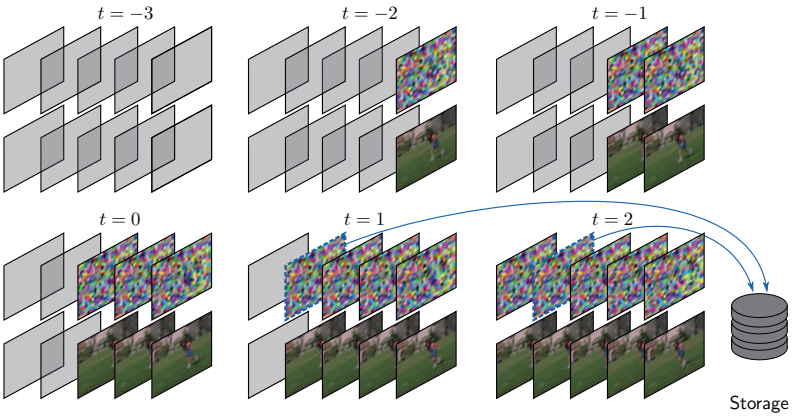


Figure 3.5: The sliding window is initialized from the front to the back positions. After the positions of the *future* and *current* frames (cf. Figure 3.4) are occupied, the segmentation of the first *past* frame (here marked in blue) can be stored as its segmentation does not change anymore.

implies that some temporally consistent superpixels will not be present in all frames as their underlying image region is not visible all the time. It is therefore crucial that the initial state of the labeling in these frames is chosen in such a way that the propagated constellation already resembles – up to some extent – the movement and structural changes inherent in the video volume. Thus, superpixels have to be moved to their new location according to the underlying image movement, and superpixels have to be terminated or created in the case of occlusion or disocclusion, respectively. Depending on the video content a concurrent initialization of all frames of the sliding window is not practicable in many situations. Therefore, a successive filling of the sliding window is proposed in this work.

To be able to better distinguish between the process of initializing the sliding window and initializing/adapting the segmentation of a new frame that enters into the sliding window the former will be referred to as initialization and the latter will be referred to as segmentation propagation. While the complete initialization process of the sliding window is depicted in Figure 3.5 and will be described in the following, the detailed description of the segmentation propagation can be found in Chapter 4.

The initial position of the *current* frame of the sliding window is at the time index $t = -(F + 1)$. Hence, the sliding window is empty at the start. When the sliding window is shifted and the first frame enters the window, its segmentation cannot

be propagated from former frames. Therefore, it is initially filled by a regular grid or a honey-comb-like arrangement of superpixels as proposed in [74]. This frame is positioned at the frontmost location of the sliding window. As a *future* frame its segmentation is mutable and thus the energy-minimization with respect to Equation (3.3) is performed. Then, the sliding window is shifted. Thereby, a new frame enters the window at the frontmost position. The old frame moves to the next position and its segmentation gets propagated onto the new frame. Afterwards several optimization iterations are performed to fit the superpixel boundaries to the frame content. This procedure is repeated until all positions in the sliding window are occupied. Then, the generation of temporally consistent superpixel can further proceed by repeatedly shifting the sliding window by one frame until the video sequence is completely processed. After all positions of the sliding window are occupied, a frame will leave the window at each shift and therefore will no longer be regarded during the optimization. But as the segmentation of the first *past* frame is already immutable the final superpixel labeling of a frame can be stored when arriving at this position.

Chapter 4

Superpixel Propagation and Handling of Structural Changes

The contribution of this chapter is twofold. The first part introduces two approaches for propagating the superpixel segmentation of a video frame onto its consecutive frame. In the second part of this chapter, methods are introduced to handle structural changes inherent in the video volume which are introduced by object- or self-occlusion and disocclusion.

4.1 Segmentation Propagation

After performing Z iterations of the hybrid optimization, the sliding window is shifted by one frame along the temporal axis of the video volume. Thereby, a frame leaves the sliding window and another frame enters into its range. Because the label assignments of the pixels in the new frame are at first glance completely unknown, the frame needs to be initialized. In the frame-by-frame processing approach of [82], the initialization is achieved by copying over the segmentation of the previous frame. But as image regions can move significantly from one frame to the next, this kind of initialization can lead to heavy segmentation errors. For example, when a superpixel holds its position, while the corresponding image region moves more than the edge length of the superpixel. In this case, the superpixel could change from being part of the foreground object to being part of the background. This is especially the case in video sequences with large object motion or camera displacement.

To encourage more temporal consistency, the authors of [46] proposed to utilize optical flow to propagate the superpixels onto the next frame. For the case of a sliding window, the authors of [63] achieved a similar result by propagating the intermediate segmentation of the most future frame $t{+}F{-}1$ (cf. Figure 3.4) onto $t{+}F$. The propagation is performed in both approaches by shifting the spatial centers of the superpixels by a weighted average optical flow computed over the area of each superpixel. The superpixels on the new frame are then created by starting from these initial spatial centers. In [46], the propagated centers provide the seed points to grow the superpixels on the frame using level-set techniques. And in [63], they

serve as the initial cluster centers in the iterative clustering-based approach. Both approaches share the common property that all pixels of the frame are reassigned during the optimization. Therefore, it is sufficient in [63] and [46] to only propagate the spatial centers. As the contour-based optimization procedure introduced in Section 3.3 requires a fully populated superpixel label map, solely propagating the spatial centers is not applicable in this work. Hence, this chapter will discuss different approaches to propagate an entire superpixel segmentation onto a new frame while roughly fitting it to the new image content. This substantially increases the complexity of the segmentation algorithm. But it was shown previously by the author of this thesis that the combination of propagating an entire label map and subsequently performing a contour-based optimization significantly improves the stability of the spatial constellation of the superpixels over time [65]. An example for the increased stability taken from [65] can be seen in Figure 4.1.

To warp a fully populated superpixel segmentation onto a new frame, there exist several possible approaches. In Figure 4.2, a visualization of a subset of these possibilities is shown that will be further discussed in the remainder of this section. The two variants depicted in Figure 4.2(a) and (b) show a per pixel propagation. Thereby, a dense per pixel optical flow is utilized to propagate each pixel independently. This can be done in a forward- or backward-directed manner. Figure 4.2(c) shows superpixel-wise propagation where each superpixel is shifted by a single optical flow vector. All three approaches are based on dense optical flow and will be described in Section 4.1.1. As the computation of a dense optical flow field is highly demanding, a computationally more efficient option is proposed in Section 4.1.2 which is depicted in Figure 4.2(d). Here, sparse optical flow vectors are utilized which are gained using the KLT approach described in Section 2.2.2. By spanning a triangle mesh between the feature points for which optical flow vectors are available, the labels inside each triangle can be warped by using the respective triangle deformation. In the subsequent Section 4.2, two approaches will be presented that can adapt the superpixel segmentation to the structural changes in the video volume induced by occlusion.

4.1.1 Propagation using Dense Optical Flow

The arguably most intuitive way to propagate an entire superpixel label map using dense optical flow is the forward-directed method depicted in Figure 4.2(a). Here, the optical flow vector of each pixel is used to shift the label of the pixel towards the position the vector is pointing to. While this procedure is simple and quickly implemented, its major drawback is its sensitivity to noisy and incorrect optical flow vectors. These lead to ambiguities where several vectors point onto the same pixel position. Thereby, the labels of several pixels are mapped onto the same spot while many other pixels are left uninitialized during the propagation. If the number of

Figure 4.1: Example taken from [65] for a challenging segmentation task with low contrast and large object motion. Top row: the original sequence with a marked area magnified in the rows below for which a full segmentation was performed. Middle row: spatial center propagation approach of [63]. Bottom row: combination of propagating the entire superpixel map and performing contour-based optimization. Only a subset of superpixels is shown (manually selected and colored). Same color means temporal connectedness. In the middle row, the superpixels are torn away by the motion introduced by the camera panning and motion of the bus, while they keep their position and constellation in the bottom row.

uninitialized pixels is high, this can lead to a disturbance of the spatial superpixel constellation similar to the one shown in the middle row of Figure 4.1. To encounter the problem of uninitialized pixels due to duplicate assignments, the optical flow direction can be reverted as depicted in Figure 4.2(b). Here, the optical flow field is computed from frame $t+F$ to frame $t+F-1$. This results in a flow vector from each pixel of the uninitialized frame to a position in the previous frame where the new label can be looked up. If two vectors point onto the same spot, all pixels are assigned the same label thereby no pixel is left uninitialized. While this approach ensures the exhaustive initialization of all pixels, it is still sensitive to noisy flow vectors.

To gain robustness against noisy vectors, the method proposed by Levinshtein et al. [46] tries to utilize the inherent properties of the superpixels. By the defi-

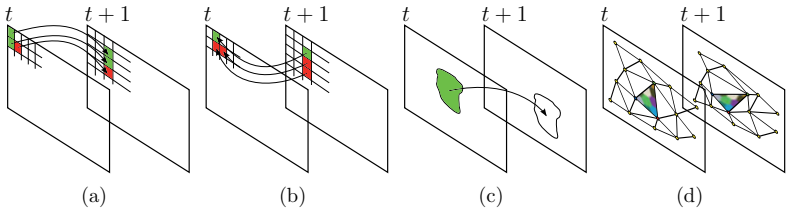


Figure 4.2: Four possible methods to transfer a superpixel segmentation onto a new frame. (a) and (b) depict pixel-wise propagation methods. In (c), an entire superpixel shape is transferred using a single optical flow vector. The triangle mesh-based method depicted in (d) is in general more efficient as it is based on tracked feature points instead of dense optical flow. Images partially taken from [65].

nition given in [47], superpixels should be limited in their spatial extent and their boundaries should coincide with the object boundaries present in the image. Due to their limited spatial extent, the optical flow inside each superpixel is assumed to be almost constant. Hence, errors introduced to the flow field by general image noise can be canceled out by taking the average of the flow vectors inside each superpixel. While this helps against noisy flow vectors and single outliers, this kind of filtering does not yield any protection against systematic errors in the calculation of the flow field. One type of systematic error is the oversmoothing of motion discontinuities. These emerge for instance at occasions in which the motion direction of an object is different from the direction of the background. At these locations, the smoothness prior inherent in the algorithms described in Section 2.2.1 and 2.2.4 becomes invalid. Thus, the smoothness terms in the Equations (2.24) and (2.35) lead to a contamination of the flow vectors of one side of the motion boundary with the flow direction from the other side. These erroneous vectors can induce a significant drift to the average flow vectors which as a result produces an incorrect propagation. To reduce the malicious effects of these vectors, [46] utilizes the property of the superpixels. Because the superpixel boundaries coincide with the object boundaries, they are also most likely to coincide with the motion boundaries. By applying a weighting function to the flow vectors, which decreases when approaching the superpixel rim, the influence of the potentially inaccurate vectors can be minimized. Thereby, improving the accuracy of the averaged optical flow vectors utilized during the propagation.

The approaches for superpixel for video content described in [46] as well as [63] utilize the averaged vectors to propagate the spatial centroids of the superpixels. While in [46], the propagated centroids are used as seed points to grow the superpixels in the new frame using level-set techniques, the centroid serve as initial spatial cluster centers in the clustering based approach of [63]. As the contour based opti-

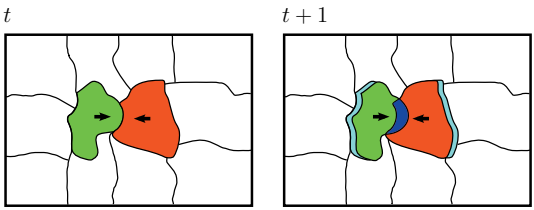


Figure 4.3: Propagating superpixels in a forward-directed manner can lead to situations where two adjacent superpixels are propagated towards each other (left image). Thereby, an overlap can occur as depicted in the right image and indicated in dark blue. Similarly, a gap (light blue) can occur where two superpixels are shifted in opposite directions. Image taken from [65].

mization requires an entirely populated superpixel label map, the solely propagation of the spatial centroids is not sufficient for the proposed framework. It is therefore proposed in this thesis to propagate the entire shape of each superpixel in a forward-directed manner by shifting each superpixel in the direction of the weighted average flow vectors. Thereby, the noise robustness of the weighted mean optical flow is leveraged and it is combined with the constellation and shape preserving properties of the propagation of a fully populated superpixel label map. To weight the optical flow vectors, a symmetric, two-dimensional Gaussian function is utilized (similar to [63]). The peak of the monotonically decreasing weighting function is chosen to coincide with the spatial center of the superpixel. Hence, the weighted average can be calculated as

$$\vec{\mu}_{b,t}^{\mathcal{F}} = \frac{1}{\sum_{i \in \boldsymbol{n}_{b,t}} K_{\sigma_{\mathcal{F}}^2}(\vec{x}_i^{\mathcal{S}} - \vec{\mu}_{b,t}^{\mathcal{S}})} \sum_{i \in \boldsymbol{n}_{b,t}} \vec{\omega}_{i,t} K_{\sigma_{\mathcal{F}}^2}(\vec{x}_i^{\mathcal{S}} - \vec{\mu}_{b,t}^{\mathcal{S}}) \tag{4.1}$$

where $\sigma_{\mathcal{F}}^2$ denotes the variance of the Gaussian kernel function $K_{\sigma_{\mathcal{F}}^2}$. The variance is selected to be $\sqrt{|\boldsymbol{n}_I|/(4 \cdot |\boldsymbol{k}|)}$ which corresponds to the radius of the average-sized superpixel in the frame. To avoid a shift onto invalid grid coordinates, each vector element is rounded to a valid integer value.

In the case that the averaged flow vectors of two neighboring superpixels point towards each other as depicted in Figure 4.3, the forward-directed propagation of the superpixel shapes can lead to overlapping superpixel parts. These overlaps can yield essential information about the structural changes inherent in the video volume because they are an indicator for the presence of object occlusion or self-occlusion. Similarly in other places, superpixels are shifted away from each other. Thereby, gaps in the propagated superpixel segmentation are produced which can be seen as indicators for disocclusion in the depicted scene. The information about these locations, gained in the forward-directed propagation, can be utilized to adapt the

segmentation to the structural changes induced through occlusion and disocclusion. Approaches to adapt the segmentation will be further described in Section 4.2. As the computational burden of dense optical flow methods is high when compared to sparse methods, the next section will describe an approach to efficiently propagate a fully populated superpixel segmentation through sparse optical flow.

4.1.2 Efficiency Improvement through Sparse Optical Flow

The following approach for a fast label propagation is inspired by the work presented in [55] and is visualized in Figure 4.4 for two sample video frames. Instead of calculating a dense optical flow as described in the previous section, only a set of sparse feature points is tracked between the current frame t and the next frame $t+1$ whose superpixel label map needs to be initialized. The feature points to track are detected on frame $t+1$ using a Harris corner detector [36]. The method of [75] is utilized to select "good" features to track. Finally, the tracking of the features is performed by the KLT feature tracker described in Section 2.2.2 (see Figure 4.4 second row). Outliers are removed by the cluster filter proposed in [55]. By applying a Delaunay triangulation as described in Section 2.2.3, a mesh is generated connecting the feature points of frame $t+1$ (Figure 4.4 third row, right). Subsequently, the mesh is warped (backward) onto the superpixel label map of frame t (Figure 4.4 third row, left) using the movement of the tracked feature points. Under the assumption of a piece-wise planar surface contained in each of the triangles, an affine transformation can be utilized to warp the content of each triangle (forward) from frame t onto frame $t+1$ as illustrated in Figure 4.5. Therefore, the transformation matrix $\mathbf{T}_{l,t+1}$ in homogeneous coordinates for each triangle l between frame $t+1$ and t is determined using the three tracked feature points of the triangle.

$$\mathbf{T}_{l,t+1} = \begin{bmatrix} a_{l,1} & a_{l,3} & a_{l,5} \\ a_{l,2} & a_{l,4} & a_{l,6} \\ 0 & 0 & 1 \end{bmatrix}_{t+1} \tag{4.2}$$

The matrix elements $a_{l,1}$ to $a_{l,4}$ determine the rotation, shearing, and scaling, whereas the elements $a_{l,5}$ and $a_{l,6}$ denote the translation. Using this transformation matrix of the triangle, the homogeneous coordinates of each pixel $(x,y,1)^\top_{t+1}$ in frame $t+1$ can be transformed into coordinates $(\tilde{x},\tilde{y},1)^\top_t$ of frame t.

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ 1 \end{bmatrix}_t = \mathbf{T}_{l,t+1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}_{t+1} \tag{4.3}$$

The transformed coordinates are clipped at the image borders and rounded to the nearest valid pixel position. Subsequently, they are used to look up the label in the superpixel label map of frame t. By assigning each pixel the looked up label, a fully

Figure 4.4: From top to bottom row: Original frames from [79] (cropped). Sparse feature points detected in frame $t+1$ are tracked backwards onto frame t. A mesh is obtained from a triangulation of the feature points in $t+1$. Through the movement of the tracked features, the mesh is deformed onto frame t. The superpixel label map of frame t is then warped by an affine transformation according to the inverse deformation of the triangles. The deformed label map is utilized as initialization for the optimization of frame $t+1$. Figure taken from [66].

populated initial segmentation is generated for frame $t+1$ (Figure 4.4 bottom row). To ensure that each pixel of the frame $t+1$ is covered by the mesh, four feature points are inserted at the corners of the frame and four at the middle of the frame borders.

Occasionally, some pixels are split-off from the main mass of a superpixel due to the warping of the transformation. Because the spatial coherency of the superpixels has to be ensured, these fractions are identified and an invalid label is assigned to them. Arising thereby, a label of a directly neighbored superpixel will be assigned to them during the next optimization step. Since this step is also required in a dense optical flow-based propagation approach as described in Section 4.1.1, it does not introduce any additional computational overhead.

As can be seen in the bottom right image of Figure 4.4, the sparse flow-based method performs a reliable label propagation on the inside as well as on the outside of the depicted objects. Solely for triangles that contain feature points attached to the background as well as to the object, an incorrect transformation is created. This is also the case for triangles which are attached to multiple objects moving in different directions. The effect can especially be seen in the area in front and on top of the car in Figure 4.4. In these locations, the movement of the car leads to a squeeze and shearing of the superpixels. In addition to this undesirable effect, the approach also lacks the capabilities to detect occlusions and disocclusions as provided by the approach proposed in the last part of the previous section. Nevertheless, the propagation by the sparse optical flow will be included in the experimental section as a baseline because it provides a universal and efficient way to propagate the superpixel segmentation.

In the next section, two approaches for the handling of structural changes in the video volume will be described. While the first is compatible with all propagation methods described in this section, the second can only be applied with the forward-directed propagation described in the second part of Section 4.1.1. This is because it relies on the dis-/occlusion detection only provided by the forward-directed propagation.

4.2 Handling of Structural Changes

Through the color conserving effect of the past frames of the sliding window (described in Section 3.4), the superpixels stick to equally colored regions over time. If these regions are part of a moving object and a propagation of the superpixels is performed, the superpixels follow their underlying region through the scene. Hereby, the object's movement through the scene causes occlusions and disocclusions. These lead to structural changes in the video volume as can for instance be seen in the volume of the table tennis sequence depicted in Figure 3.2(b). Here, the table tennis ball dynamically moves while the background is static. Thus, foreground superpix-

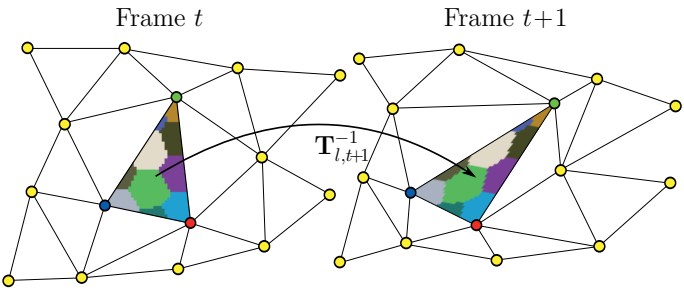


Figure 4.5: Warping of a superpixel label map covered by a triangle mesh. The parameters of the transformation matrix $\mathbf{T}_{l,t+1}$ are determined from the movement of the three vertices of the triangle. Image taken from [66].

els occupying the image region of the ball will in subsequent frames occupy regions that were previously occupied by background superpixels.

If no adaption of the segmentation is performed, the undesirable squeezing and expanding effects occur. An example of these effect is depicted in Figure 1.2. By adapting the segmentation to the structural changes, the homogeneous size constraint can be satisfied and the consistency of the superpixel flow with the underlying image movement can be improved. Thereby resulting for instance in a correct static background segmentation in scenes like the table tennis sequence of Figure 3.2. To adapt the segmentation to the structural changes, two operations need to be performed. First, superpixels whose corresponding image region gets occluded or moves out of the field of view of the camera need to be terminated. And second, new superpixels have to be created where new image regions appear either by camera movement of disocclusion.

The process to terminate a superpixel is similar to the treatment of split-off fractions of superpixels described above. By assigning an invalid label to all remaining pixels of a superpixel, they are automatically assigned to the best matching neighboring superpixel during subsequent optimization steps. Frames which become *past* frames of the sliding window will not be further optimized. Due to this, it has to be made sure that a valid label is assigned before a frame enters the area of the *past* frames. Therefore, superpixels are only deleted in the newest *future* frame.

Creating a superpixel can either be done by utilizing a gap in-between superpixels or by splitting up an existing superpixel into two. Additionally, it is possible to utilize a gap between a superpixel and the frame border. The steps to split a superpixel are depicted in Figure 4.6. For similar reasons as mentioned above and to ensure an optimization of the boundary between the new superpixels, the splitting

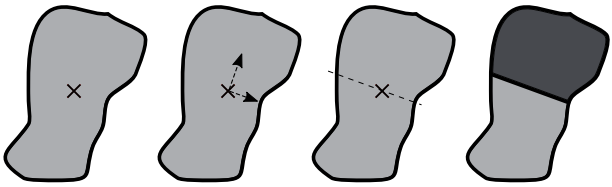
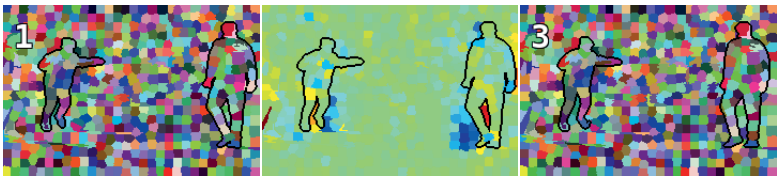


Figure 4.6: Steps performed to split a superpixel into two parts. In (a) the spatial center is computed. In (b) the eigenvectors of the spatial distribution of the pixels assigned to the superpixel are shown. (c) and (d) show the line through the spatial center along the eigenvector corresponding to the smaller eigenvalue which is used as demarcation line to partition the superpixel.

is performed only in the newest *future* frame. The process basically follows the principle proposed in [90]. First, the spatial center of the superpixel is calculated. Second, the eigenvectors and eigenvalues of the spatial distribution of the pixels assigned to the superpixel are determined. Third, a demarcation line is defined which passes through the spatial center of the superpixel and follows the eigenvector corresponding to the smallest eigenvalue computed in the previous step. The pixels on one side of the line keep their label and the pixels on the other side are assigned a new, unused superpixel label. The side which keeps the label is determined by comparing the mean color vectors of the split regions with the former mean color of the superpixel. The region with the lower Euclidean distance is selected to keep the label.

In general, the number of superpixels per frame is user-defined and hence the actual number of created superpixels should follow the specification given by the user. A discrepancy between the specified and the actual number of superpixels can occur if at a given point in time either more superpixels were created than terminated or vice versa. To enforce a fixed number of superpixels per frame, the number of creations/splits and terminations can be balanced. In order to decide at which location the superpixel segmentation should be altered, i.e. which superpixel to terminate or where to create a new one, the next subsections will introduce two methods for handling structural changes. The first will be purely based on the pixel count of each superpixel. This approach enables an efficient handling of the structural changes and can be applied in combination with any superpixel propagation method. The second proposed variant will be based on the dis-/occlusion information, whose collection was described in the second part of Section 4.1.1. Due to its nature, this method can only be applied if the dis-/occlusion information is provided by the propagation method.



(a) (b) (c)

Figure 4.7: A visualization of the temporal gradient of the superpixel size. (a) and (c) show the label maps for the first and third frame of the sequence. The silhouettes of the soccer player were manually marked for illustration purposes. In (b), the gradient of the superpixel size between the two frames is visualizes as a heat map (blue means shrinkage and red means growth). Especially in front of the moving legs of the players the superpixels shrink, while they expand between the legs.

4.2.1 Size-based Handling

The squeezing and expanding effects depicted in Figure 1.2 and 4.4 are the results of moving objects occurring in the scene. It can be seen that the superpixels shrink in front of the object where the background gets occluded by its movement. Simultaneously, their size increases behind the object to fill the newly disclosed space. Figure 4.7 visualizes this effect by color coding the variation of the superpixel size in time. It shows an example using the soccer player sequence whose original frames are depicted in the Figures 1.1 and 1.2. The left and right images show the label maps of the first and the third frame, respectively. In the middle image a heat map is drawn where each superpixel is filled with a color proportional to the gradient of the superpixel size. A blue color denotes a negative gradient (shrinkage) and a red color denoted a positive gradient (growth). It can be seen that the gradient has a strong negative amplitude (blue) in front of the moving legs of the two players and a strong positive amplitude (red) behind them. Simultaneously, the gradients in most of the remaining image regions are virtually zero (green). This observation strongly suggests that the structural changes can be handled up to some extent by observing the superpixel size over time. The simplest solution that was applied for example in [65] is to define a minimum and a maximum threshold for the superpixel size. After each shift of the sliding window, the size of each superpixel is measure in the current frame. Subsequently, superpixels whose size falls under the predefined minimum size are terminated and superpixels that grow bigger than the maximum size are split. The valid size corridor which is spanned through the two thresholds is illustrated in Figure 4.8. Here, the minimum and maximum thresholds are denoted by $|\boldsymbol{n}|_{\min}$ and $|\boldsymbol{n}|_{\max}$, respectively.

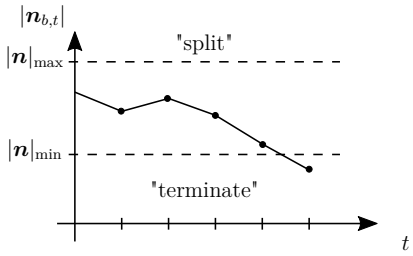


Figure 4.8: A visualization of the valid superpixel size corridor in the current frame of the sliding window which is spanned between a minimum and a maximum value.

While the size-based approach handles the structural changes in most situations successfully, it has two major drawbacks. First, it can introduce superpixel motion where no underlying image flow is present. The reasons for this unintended superpixel flow are distinct image regions which are smaller than $|\boldsymbol{n}|_{\min}$. If a superpixel which corresponds to such a region gets deleted because it is smaller than $|\boldsymbol{n}|_{\min}$, its pixels will be taken over by a neighboring superpixel. If the region is distinct enough from its surrounding, it can happen that the neighboring superpixel reduces to the pixel of the distinct region and therefore gets deleted as well. Even in static scenes, this leads to a superpixel movement toward the distinct region which causes the superpixel flow to differ significantly from the underlying image flow. A second issue are the rather random locations of the termination and splitting operations. As can be seen from Figure 4.7, the shrinking and expanding superpixels are often found in front or behind a moving object but in many cases do not exactly coincide with the actual occlusion or disocclusion boundaries present in the video material (i.e. the black silhouettes in Figure 4.7). This comes from the fact that none of the used conditions are specific to the area of the occlusion and disocclusion boundaries.

A solution to the first issue can be the method proposed by the author of this thesis in [63]. Here, the superpixel size gradient is utilized to predict the size of a superpixel in the future. If a superpixel is small, but its size does not further decrease, it will not be deleted. While this eliminates the unintended superpixel motion pattern, the locations of the adaptations are partially still at rather random spots. This by itself can have a negative influence on the correctness of the superpixel flow as well. Therefore, a new approach will be proposed in the following section which explicitly utilizes occlusion and disocclusion information gained during the superpixel label map propagation.

4.2.2 Handling by Occlusion and Disocclusion Detection

This section proposes a novel approach to handle structural changes which explicitly utilizes the occlusion information gained during the forward-directed propagation described in the second part of Section 4.1.1. The information consists of overlapping parts of superpixels on the one hand and gaps in-between superpixels on the other hand. By classifying the overlapping pixels into the categories of occluded and occluding candidates, knowledge is gained of where the course of the actual occlusion boundaries lies. This knowledge is then utilized to decide on the termination of superpixels. Similarly, gaps between the propagated superpixels are used as indicators for disocclusion where new superpixels should be created.

Before performing the forward-directed propagation, as described in Section 4.1.1, no statement can be made on any depth ordering of the superpixels. Therefore, they are at first propagated successively in an arbitrary order onto the new frame. The process is illustrated in Figure 4.9 on the example of the parachute sequence from the data set published by [80]. It can be seen that after the propagation through the shift by the mean optical flow vector the superpixels do overlap. The overlapping areas can be seen as a binary mask in the lower right image of Figure 4.9. For the segmentation to be consistent with the underlying image content, the pixels in the overlapping region need to be assigned to the superpixel which corresponds to the occluding scene content. But without further information about the objects of the scene and their movement, it is unclear which superpixel is located on top of which superpixel. Therefore, a mechanism is needed to determine the order of the superpixels and to assign the pixels of the overlapping region to the superpixel whose appearance model fits best to the image content.

Based on the local appearance information, this assignment problem could be solved by the contour-based optimization described in Section 2.1.3. But the compactness and homogeneous size constraints of the optimization would be likely to have a big influence on the final labeling in many cases and could override the appearance-based assignment. To lower the influence of these, constraints the weight α in the energy term (2.18) could be decreased. But this approach could result in superpixels with extremely rugged boundaries since no influence can be wielded on the homogeneity of the labeling. Hence, this work proposes to employ a graph cut-based optimization on the overlapping areas as described in Section 2.1.1 in order to be able to enforce a homogeneous labeling in these areas. In this way, the label homogeneity is influenced by the pairwise energy of the labeling assessment energy term (2.10). The proposed approach was inspired by [85] where superpixels are created by laying out overlapping, rectangular patches on the image plane. By spanning a graph over the pixels, the optimal superpixel affiliation of each pixel can be found by applying a multi-label graph cut optimization [8]. While the approach

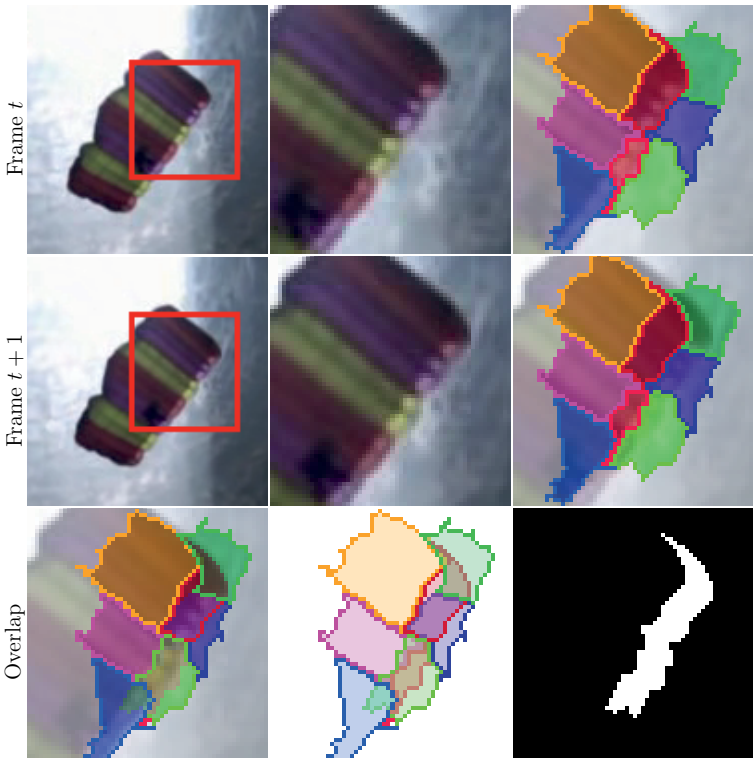


Figure 4.9: Visualization of the overlap detection. Top row (left to right): Original frame t. Magnification of the marked area. A manually selected subset of superpixels is displayed as an overlay. Middle row: Subsequent frame $t+1$. The subset of superpixels taken from frame t blended over the image content of frame t+1. Bottom row: Each superpixel of the subset has been shifted by the weighted mean optical flow vector computed over its respective area (left). For clarity and to emphasize the overlapping parts of the superpixels (represented through the color blending), the image data is removed (middle). Thereby, the ordering in still unknown. The area of the overlapping parts of the superpixels (marked in white in the right image) are further processed to decide on the ordering of the superpixels.

of [85] when applied to whole images is computationally highly demanding and can take more than 10 seconds per frame, it can be performed quickly on small image portions like the overlapping regions. This work therefore proposes to instead of laying out overlapping patches as in [85] to apply the multi-label optimization on the parts of the superpixels which overlap after the propagation. Thereby, the assignment problem of the overlapping regions is solved without loosing the ability to enforce a homogeneous labeling during the optimization.

To prepare the graph cut-based optimization, the involved superpixel labels for any overlapping region are recorded during the propagation. The set of possible labels will be denoted with $\boldsymbol{k}_{\text{ovr}}$. As the assignment problem is solved for every propagated segmentation independently, the frame index will be skipped in this passage where an ambiguity can be excluded. The graph is spanned only over the pixels of the overlapping regions $\boldsymbol{n}_{\text{ovr}}$. Each node of the graph represents a pixel to which one of the labels contained in $\boldsymbol{k}_{\text{ovr}}$ can be assigned. The edges between nodes indicate neighboring pixels in a 4-connected neighborhood. The quality of every possible labeling of the set of nodes $\boldsymbol{b}_{\text{ovr}} = \{b_i | b_i \in \boldsymbol{k}_{\text{ovr}}, i \in \boldsymbol{n}_{\text{ovr}}\}$ can be assessed by an energy function that is defined as follows

$$E_{\text{ovr}}(\boldsymbol{b}_{\text{ovr}}) = \sum_{i \in \boldsymbol{n}_{\text{ovr}}} D(i, b_i) + \lambda_{\mathcal{G}} \sum_{(i,j) \in \boldsymbol{c}_{\text{ovr}}} V_{i,j}(b_i, b_j)\,. \tag{4.4}$$

Here, $\boldsymbol{c}_{\text{ovr}}$ is the set of cliques in the overlapping area where only the 4-connected neighbors are regarded. As the decision about the topmost superpixel should depend on the similarity of the underlying image data to the possible superpixels, the unary term is defined as follows

$$D(i,b) = \begin{cases} E_{\text{col}}^{\text{tcs}}(i,b), & \text{if } b \in \boldsymbol{k}_{\text{ovr},i} \\ \infty, & \text{else} \end{cases} \tag{4.5}$$

with $\boldsymbol{k}_{\text{ovr},i}$ denoting the set of superpixel labels which overlap at the pixel i. This selection of the unary term guarantees that only those labelings produce a finite total energy which have a label assigned at each location that was actually involved in the overlap at this position. As the unary term only includes the color depending energy of Equation (2.18), the superpixel compactness constraint does not interfere in this case. To still favor equally labeled neighbors and thus a homogeneous and a spatially coherent labeling, the pairwise term is selected to be

$$V_{i,j}(b_i, b_j) = \Gamma(b_i, b_j) \cdot \exp\left(-\beta |\vec{x}_i^{\mathcal{C}} - \vec{x}_j^{\mathcal{C}}|^2\right). \tag{4.6}$$

Where $\Gamma(\cdot,\cdot)$ is the indicator function which is defined to be zero if both input arguments are equal and one otherwise. The weighing factor β serves as a normalization constant and includes the variance of all color differences in the overlapping regions and is chosen to be as defined in [69]

$$\beta = \left(2E\left[(\vec{x}_i^{\mathcal{C}} - \vec{x}_j^{\mathcal{C}})^2\right]\right)^{-1} \tag{4.7}$$

where $E[\cdot]$ denotes the expectation value. Choosing the pairwise term in such a way prefers equally labeled pixels which encourages a spatially coherent labeling. If different labels are assigned to the pixels and their color values are similar, higher costs are produced than if their colors are different. Thereby, encouraging a homogeneous labeling of the segments. The optimal labeling of the graph, which is finally used to decide on which superpixel overlaps all others, is gained by performing two alpha expansion iterations of the optimization framework proposed in [41].

Although the pairwise term of Equation (4.4) favors a homogeneous labeling, it is not guaranteed that the resulting labeling is spatially coherent. For those rare cases where pixels are split-off from the main mass of a superpixel, the same treatment as described above is applied. In general, the amount of overlapping superpixels increases with the degree of motion that is present in the scene. Therefore, the size of the graph increases in conjunction with the dynamic of the scene. But as often not all pixels are part of the same 4-connected component, the graph consists of multiple disjoint subgraphs. For these, the assignment problem can be solved independently which in general keeps the extent of the graphs at a reasonable level.

After solving the assignment problem in the overlapping regions, the final labeling is compared with the initially overlapping superpixels. For each pixel that got occluded, a record is created. This results in a set of occluded pixels per superpixel b and time index t denoted by $\boldsymbol{n}_{b,t}^{\mathcal{H}}$. If the underlying image region of a superpixel is getting more and more occluded by an object, the sum of occluded portions will increase over time. Therefore, it is important to keep track of the number of pixels that have already been occluded. For the case of this framework, the extent of the sliding window limits the time of recording. To decide whether a superpixel b should be terminated, the accumulated sum of the occluded pixels is checked against the previous size of the superpixel with the following condition.

$$\sum_{\tau=t-P}^{t+F} |\boldsymbol{n}_{b,t}^{\mathcal{H}}| \geq |\boldsymbol{n}_{b,t-P}| \tag{4.8}$$

Thereby, $|\boldsymbol{n}_{b,t-P}|$ denotes the number of pixels assigned to superpixel b in the oldest frame of the sliding window. The condition (4.8) is checked each time a superpixel label map is propagated onto a new frame. If it is fulfilled the superpixel is terminated by the procedure described above.

An additional reason to keep track of the occluded pixels and their position is illustrated in the left column of Figure 4.10. Here, the superpixels are schematically depicted by circles and each row depicts a time step. In each time step, a propagation and a number of Z iterations of the hybrid optimization as described in Section 3.3 are performed. It can be seen that the red superpixel is shifted in each time step to the right by the mean flow vector inside the superpixel (indicated by the arrow). The two superpixels to its right are not shifted as their mean flow

is zero. Therefore, they should not move at all when the hybrid optimization is performed. But as a small fraction of the yellow superpixel gets occluded by the red one, the recalculated spatial center (indicated by the crosses) will be far more right than the initially propagated one. During the hybrid optimization, the compactness constraint enforced by $E_{\text{pos}}^{\text{tcs}}$ in Equation (3.4) will therefore lead to a gradual right shift of the yellow superpixel. Thereby, it will regain in size as the spatial distance term in Equation (3.4) favors equally sized superpixels (indicated by the dotted outline). This in return will lead to a shift of the right superpixel unless it is stopped by an object boundary which would result in both right superpixels being squeezed together. This observation is similar to observations made in [14] about the superpixel flow at image boundaries.

To encounter the problem of the pushed superpixels and to improve the consistency of the superpixel flow, this work proposes to utilize the knowledge gained during the propagation once more to compensate for the hidden fractions of the superpixels. In order to integrate the hidden fraction into the hybrid optimization, they have to be considered when the new spatial centers of the superpixels are recalculated. But in general, the sets of hidden pixels of two different frames might not be aligned to each other. This is because not all half-occluded superpixels are static as suggested by Figure 4.10 but will also move during the process of occlusion. Therefore, the position of their occluded fractions need to be updated over time. This enables the utilization of the hidden pixels of different frames to update the spatial center of subsequent frames. As the hidden fractions of a superpixel are still a logical part of its visible fraction, the hidden part should move in the same way as the visible part. This can be achieved by applying the same weighted average flow vectors to the hidden as to the visible fractions. Hence, the spatial center of a superpixel b in frame t' of the sliding window that is centered around frame t can be recalculated as follows

$$\vec{\mu}_{b,t'}^{\,\mathcal{S}} = \frac{1}{|\boldsymbol{n}_{t'}^{A}|}\left(\sum_{i\in\boldsymbol{n}_{b,t'}}\vec{x}_i^{\,\mathcal{S}} + \sum_{j\in\boldsymbol{n}_{b,t'}^{\mathcal{H}}}\vec{x}_j^{\,\mathcal{S}} + \sum_{\tau_1=t-P}^{t'-1}\left(\sum_{k\in\boldsymbol{n}_{b,\tau_1}^{\mathcal{H}}}\vec{x}_k^{\,\mathcal{S}} + \sum_{\tau_2=\tau_1}^{t'-1}\vec{\mu}_{b,\tau_2}^{\,\mathcal{F}}\right)\right). \tag{4.9}$$

Here, $|\boldsymbol{n}_{t'}^{A}|$ is a substitute for $|\boldsymbol{n}_{b,t'}| + |\sum_{\tau_1=t-P}^{t'}\boldsymbol{n}_{b,\tau_1}^{\mathcal{H}}|$. By incorporating the hidden fractions, the shift of the spatial centers and the resulting expansion is suppressed which is illustrated in the right column of Figure 4.10. It shows that the newly calculated spatial center now coincides with the initially propagated center as also the occluded part is regarded (shaded area). Therefore, the two superpixels on the right are not shifted in the following iterations of the optimization. From the third to the fourth row, the yellow superpixel gets occluded completely and is therefore deleted as Equation (4.8) is fulfilled. Thereby, it was assumed that the size of the superpixel in the first row is equal to the size in the oldest frame of the sliding window.

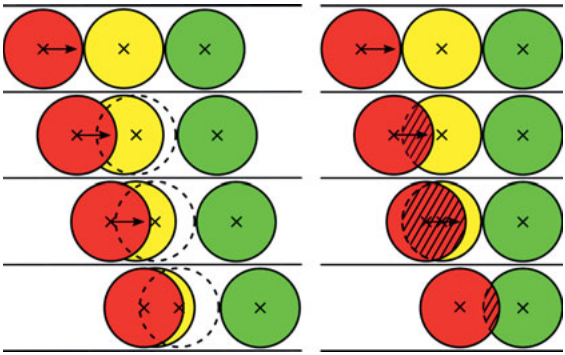


Figure 4.10: Visualization of the unintentionally shifted superpixels. The left superpixel (red) is propagated (indicated by the arrow) towards two stationary superpixels (yellow, green). Without knowledge of the yellow superpixel getting occluded by the red superpixel (left column) its newly calculated spatial center will have an offset compared to the propagated center. The offset of the spatial center will lead to a shifting of the yellow superpixel as the spatial term of Equation (3.4) pursues superpixel size equalization (indicated by the dotted circle). This effect is further propagated which results in a shift of the neighboring (green) superpixel. By utilizing the occlusion information to compute the "true" spatial center (right column) the drift can be avoided resulting in a more accurate superpixel movement.

Chapter 5

Experimental Results

This chapter contains a thorough evaluation of the proposed framework for temporally consistent superpixels. Therefore, Section 5.1 will describe the experimental setup and subsequently compare the framework to other state-of-the-art methods for supervoxels and video-based superpixels. Finally in Section 5.2, the effectiveness of the proposed framework is demonstrated on the example of interactive video segmentation.

5.1 Evaluation of Temporally Consistent Superpixels

The evaluation of the segmentation quality is a complex problem with many aspects and an additional dependency on the indented application of the segmentation. In particular, when dealing with video segmentation the evaluation has two sides. On the one hand, the per frame quality is of interest. This includes how well the segmentation boundaries comply with the visual outlines of objects in the scene. On the other hand, the spatio-temporal quality of a segmentation is of great importance. Thereby, it is of interest how exhaustively the temporal connections between the image regions inherent in the video volume have been extracted and if the temporal connections between superpixels are correct. Additionally, the superpixel flow should be consistent with the underlying image movement. To assess these segmentation properties, Section 5.1.1 introduces the established benchmark metrics from the literature. In Section 5.1.2, the utilized data sets as well as the experimental setup are described. This also includes the optimal framework-parameter selection. Which is achieved via grid search on an independent training set of video sequences. Subsequently, the segmentation quality is evaluated and compared to the state of the art on a test set for which the results are shown in Sections 5.1.3 and 5.1.4. Finally, the computational complexity of the proposed approach is evaluated in Section 5.1.6.

5.1.1 Benchmark Metrics

The standard metric used to evaluate a binary image or video segmentation is the **Misclassification Rate** ϵ. It measures the ratio of incorrectly labeled pixels, i.e. foreground pixels which were labeled as background pixels and vice versa, to the total amount of pixels. Given a segmentation into two pixel sets $\boldsymbol{n}_{fg}$ and $\boldsymbol{n}_{bg}$ (foreground and background) as well as a ground truth segmentation denoted by $\boldsymbol{g}_{fg}$ and $\boldsymbol{g}_{bg}$ the misclassification rate can be calculated as follows

$$\epsilon = \frac{|\boldsymbol{n}_{fg} \cap \boldsymbol{g}_{bg}| + |\boldsymbol{n}_{bg} \cap \boldsymbol{g}_{fg}|}{|\boldsymbol{g}_{fg} \cup \boldsymbol{g}_{bg}|}. \tag{5.1}$$

In the case of an interactive segmentation, where a user manually marks some pixels as foreground or background, the manually marked pixels are entirely excluded from the calculation of the metric.

A more complex case is the evaluation of a multi-label segmentation with B segments $\boldsymbol{n}_1,...,\boldsymbol{n}_B$ and M ground truth segments $\boldsymbol{g}_1,...,\boldsymbol{g}_M$ where each segment contains e.g. an object of the scene. To be able to easily assess the correct classification of a pixel in such a scenario, two conditions have to be fulfilled. First, the granularity of the generated segmentation has to be equal to the granularity of the ground truth segmentation. A group of people, for example, has to be segmented as a whole or as individual entities in both partitionings so that $B = M$. Second, the correspondences between the individual parts of the segmentation have to be known, i.e. the knowledge which segment $\boldsymbol{n}_b$ corresponds to which ground truth segment $\boldsymbol{g}_b$. As a superpixel segmentation is an explicit oversegmentation of an image many objects will most likely be segmented into several parts. The location of these interior segmentation boundaries is often unpredictable and depends on different factors including the initialization and compactness constraints. Hence, it is unclear which of the boundaries are the relevant ones. Additionally, no exclusive correspondences between the segments and a ground truth segmentation can be given a priori. The same holds true for the case of video oversegmentation and temporally consistent superpixels. Consequently, a set of specialized evaluation metrics is necessary to assess the segmentation quality. These metrics will be described in the remainder of this section. While some of the metrics where initially proposed for the image segmentation case and were extended later to the video domain, others have been explicitly tailored for the video case. Further details about the metrics can be found in [57, 58, 74, 92, 93].

Since it is initially unclear which superpixel can be assigned to which ground truth segment, Levinshtein et al. [47] propose to measure the number of pixels *bleeding out* of the ground truth segments. Thereby, penalizing superpixels which cross a boundary of the ground truth segmentation. The metric was extended to the video

domain by [92] as **3D Undersegmentation Error** and can be calculated for a ground truth segmentation with $\boldsymbol{g}_1,...,\boldsymbol{g}_M$ as follows

$$\mathrm{UE} = \frac{1}{M}\sum_{m=1}^{M}\frac{\left(\sum_{(\boldsymbol{n}_b|\boldsymbol{n}_b\cap\boldsymbol{g}_m\neq\emptyset)}|\boldsymbol{n}_b|\right)-|\boldsymbol{g}_m|}{|\boldsymbol{g}_m|}. \tag{5.2}$$

A second metric for the evaluation of a video-based oversegmentation that is proposed in [92] is the **3D Segmentation Accuracy**. Prior to [92], a similar metric was already utilized to assess the segmentation quality on single images, e.g. in [51]. It measures the fraction of the ground truth segmentation that can be correctly reproduced by the superpixel segmentation. Thereby, it is again assumed that in a high quality segmentation each superpixel should only overlap with a single ground truth segment. A superpixel is assigned to a ground truth segment if the majority of the pixels of the superpixel resides inside the ground truth segment. Given M ground truth segments and the set of superpixel labels $\boldsymbol{k}_m$, which are assigned to the ground truth segment $\boldsymbol{g}_m$, the accuracy can be calculated as

$$\mathrm{SA} = \frac{1}{M}\sum_{m=1}^{M}\frac{\sum_{b\in\boldsymbol{k}_m}(|\boldsymbol{n}_b\cap\boldsymbol{g}_m|)}{|\boldsymbol{g}_m|}. \tag{5.3}$$

The above metrics presume the availability of a ground truth labeling of the input data. As the creation of such a ground truth segmentation is a labor intensive task, several authors proposed metrics to assess the quality of a superpixel segmentation without this information. A widely used metric is the **Explained Variation** proposed in [57]. It indicates how well the original image information can be represented with a given superpixel segmentation as a representation of lowest detail. Its extension to the video domain, proposed in [92], can be calculated as follows

$$\mathrm{EV} = \frac{\sum_i(\vec{\mu}_i^{\mathcal{C}}-\vec{\mu}^{\mathcal{C}})^\top(\vec{\mu}_i^{\mathcal{C}}-\vec{\mu}^{\mathcal{C}})}{\sum_i(\vec{x}_i^{\mathcal{C}}-\vec{\mu}^{\mathcal{C}})^\top(\vec{x}_i^{\mathcal{C}}-\vec{\mu}^{\mathcal{C}})}. \tag{5.4}$$

Here, $\vec{\mu}^{\mathcal{C}}$ denotes the global mean color vector and $\vec{x}_i^{\mathcal{C}}$ is the color vector at the voxel position i. The vector $\vec{\mu}_i^{\mathcal{C}}$ is the mean color vector inside the segment that the voxel i is assigned to.

A further metric, proposed in [92], to evaluate the spatio-temporal segmentation quality in the absence of ground truth information, is the average temporal length. It calculates the mean duration of the spatio-temporal segments to measure the ability to track image regions over time. As the maximal achievable mean duration depends on the number of frames of the segmented video sequence, the metric is hard to compare when calculated on videos of different length. It was therefore proposed in [14] to divide the average temporal length of the spatio-temporal segments by the total number of video frames to gain the final metric named **Temporal Extent**.

While this metric is a valid indicator for the temporal consistency of a segmentation, it has to be evaluated in conjunction with other metrics which quantify the segmentation error. This is necessary because a long temporal duration of spatio-temporal segments could also be created by a static segmentation. But a static segmentation would also result in a high segmentation error for scenes with object or camera motion which is not indicated in the temporal extent. Consequently, long temporal segments are only of high value with a low spatio-temporal segmentation error indicated by the 3D segmentation accuracy or the 3D undersegmentation error.

Another video domain metric to evaluate the quality of a spatio-temporal oversegmentation is the so called **Label Consistency**. It is proposed in [14] and measures the consistency of the superpixel flow with the underlying image movements. To measure the label consistency, it is assumed that the true movement of each pixel is known and thus the labeling of a frame can be propagated using this ground truth knowledge. Afterwards, the number of pixels which agree between the propagated labeling and the labeling generated by the algorithm for this frame is determined. The label consistency is then given as the ratio between the number of agreeing pixels and the total number of pixels per frame averaged over all frames. The importance of this metric highly depends on the use case of the spatio-temporal oversegmentation. If the final objective is solely to segment the objects of the scene the label consistency property may be of minor priority. This is because a correct segmentation of an object can be created even if the movement of the spatio-temporal segments inside and outside the objects differ from the image flow. Only if object boundaries are crossed this leads to an error in the final segmentation. For other applications such as object tracking the label consistency can be of high value as the motion of the segments might be used as a feature in the tracking pipeline.

The purpose of the above described metrics is to evaluate the spatio-temporal quality of a video oversegmentation. Since each spatio-temporal oversegmentation can be sliced at frame instances, to form a superpixel segmentation of the frame, this work will additionally utilize the following metrics to evaluate the per frame segmentation quality. To measure the compliance of the superpixel boundaries with the boundaries of the ground truth segmentation, the **Boundary Recall** will be used. It measures the fraction of the boundaries, annotated in the ground truth segmentation, which is covered by superpixel boundaries. A ground truth boundary pixel is thereby counted as covered if a superpixel boundary is located within a certain distance to this pixel. In the experiments conducted for this thesis, the distance is set to one pixel.

Additional to a high boundary recall, many authors argue that it is favorable for superpixels to have homogeneous size and compact shape. Therefore, the compactness of the created segments will be evaluated by using the **Superpixel Compactness** metric $\bar{C}$, proposed in [74]. It utilizes the **Iso-perimetric Quotient** to

quantify the circularity of a superpixel. The quotient is also used in [58] to assess the compactness of superpixels. For a superpixel b the iso-perimetric quotient is defined as the quotient of the superpixel area $|\boldsymbol{n}_b|$ and the area of a circle with a radius r_b which is equivalent to the radius of the superpixel. Thus, it can be calculated as

$$Q_b = \frac{|\boldsymbol{n}_b|}{\pi r_b^2} = \frac{4\pi|\boldsymbol{n}_b|}{L_b^2} \tag{5.5}$$

where L_b is the perimeter of the superpixel b. The superpixel compactness $\bar{C}$ for the segmentation of an image is then computed as follows

$$\bar{C} = \sum_{b\in\boldsymbol{k}} Q_b \frac{|\boldsymbol{n}_b|}{|\boldsymbol{n}|}. \tag{5.6}$$

The different superpixel sizes are taken into account by weighting each quotient with the ratio of the superpixel size $|\boldsymbol{n}_b|$ to the total number of pixels in the image $|\boldsymbol{n}|$. In order to assess the homogeneity of the superpixel sizes, the **Variance of Area**, proposed in [58], will be utilized. It is calculated for a frame t as

$$\mathrm{VoA}(t) = \frac{1}{\bar{S}^2} E\left[(|\boldsymbol{n}_{b,t}| - \bar{S}^2)^2\right] \tag{5.7}$$

where $E[\cdot]$ is the expectation value, $\boldsymbol{n}_{b,t}$ is the set of pixel of the superpixel b in frame t, and $\bar{S}^2$ is the average superpixel area.

In the original work of [92] the spatio-temporal benchmark metrics, such as the 3D undersegmentation error and the 3D segmentation accuracy, are plotted over the average number of supervoxels per video sequence. Opposed to this, it is proposed in [14] to plot the metrics over the average number of superpixels per frame. This representation is justified by arguing that videos of different length and content require in general a different number of spatio-temporal segments. While this argument is valid, such a representation also results in the loss of the spatio-temporal aspect of the metrics. This is because such a representation misses all indication of the temporal consistency of a segmentation. Thereby, an independent superpixel segmentation of the individual frames (without any connection between the superpixels of different frames) can achieve the same or a better segmentation quality (in terms of these metrics) as a temporal consistent segmentation. The lack in spatio-temporal segmentation quality, in this case, is only indicated by additional metrics like the temporal extent and the label consistency. Therefore, a reliable evaluation in this case can only be done by looking to all metrics simultaneously.

In order to avoid the loss of the spatio-temporal aspect of the metrics and to facilitate the evaluation, the results of the video-based metrics will be plotted over the number of supervoxels in this thesis. In contrast to that, the boundary recall, the superpixel compactness, as well as the variance of area are primarily 2D metrics

and will therefore be plotted over the average number of superpixels per frame. To take into account the different lengths and characteristics of video sequences, the metrics will be plotted separately for individual data sets whose videos share those properties to a certain degree. The benchmark metrics were partially computed using the code provided by [3] and [92].

5.1.2 Data Sets and Experimental Setup

To find the optimal parameter settings of the proposed framework for temporally consistent superpixels and to evaluate its segmentation quality, a diverse set of image and video data is required. This subsection introduces the utilized data sets and describes their features and limitations.

Four of the benchmark metrics described above require the availability of some kind of ground truth information. Hence, the video material needs to be annotated by human subjects or has to be produced synthetically. The material utilized in this thesis to evaluate the segmentation quality of the proposed framework in a qualitative and quantitative manner consists of three separate data sets containing different video material. The data sets vary in the characteristics of the scenes that are depicted as well as in the availability of ground truth data.

The first data set is provided by Chen and Corso [16] and consists of a collection of eight video clips with up to 85 frames per clip. While the number of sequences is low compared to other data sets, the provided ground truth information is of high value. For each frame of the eight sequences a multi-label ground truth segmentation is provided which is consistent over all frames. The temporally dense availability of the ground truth data enables an accurate evaluation of the spatio-temporal segmentation quality. In particular for approaches which provide short-range temporal segments.

The second data set is much larger and based on video sequences provided by Sundberg et al. [76]. It contains 100 video clips which are split into a training and a testing set containing 40 and 60 video clips, respectively. Each clip provides up to 121 frames and is provided in a high-definition (HD) resolution, i.e. 1920×1080 or 1280×720. Additionally, the video material is provided in a lower resolution version with only half the extent in both dimensions. For the experiments described in the remainder of this subsection the testing portion of the data set is utilized. To reduce the high computational burden of the experiments, the lower resolution version of the clips is used. In contrast to the first data set, not for every frame a ground truth segmentation is available. Instead, a set of four multi-label ground truth segmentations is provided by [30] for every 20th frame. The ground truth data for each frame was created by four different human subjects which were free to select

Figure 5.1: Comparison of the coarseness of different ground truth segmentations on the example of the *birds of paradise* sequence from [76]. The original frame in the left image was segmented by four different human subjects. Two of the four ground truth segmentations, provided by [30], are shown on the right hand side. While the right frame is segmented into details like leaves and sticks, the other partition only separates the two birds from the background.

the coarseness of the created segmentation. Therefore, the provided ground truth varies in coarseness from subject to subject. Figure 5.1 shows a frame of the *birds of paradise* sequence with the segmentation of two subjects. It can be seen that one of the subjects segmented the example frame into three segments. Simultaneously, the other subject chose to provide a finer segmentation with individual leaves and sticks separated from the background. When evaluating a video segmentation the latter type of ground truth segmentation will inevitably lead to a higher segmentation error if the number of ground truth segments is higher than the number of automatically generated ones. Since in this section only oversegmentation methods are considered in the evaluation, it can be assumed that the number of automatically generated segments is in general larger than the number of ground truth segments. Hence, the higher error, due to the higher granularity of the ground truth segmentation, has only a marginal impact on the resulting metrics. For the sake of convenience, the first set of data will be denoted as Chen's data set and the second one as Sundberg's data set.

To measure the label consistency metric, described in Section 5.1.1, ground truth optical flow information is required. Since for both former data sets no such information is provided, the MPI Sintel flow data set is utilized as an additional data source. It is provided by [13] and consists of 23 synthetic scenes taken from the *Sintel* movie. The rendered sequences are up to 53 frames long and contain a lot of object motion, very dynamic camera movements, as well as motion and defocus blur. In Figure 5.2 a pair of example frames and the corresponding ground truth optical flow are shown.

In order to evaluate the proposed framework, it is implemented in C++. It was already discussed, in Chapter 4, that the computation of the dense optical flow makes up a large portion of the computational burden of the framework. Therefore, besides a version with forward-directed, dense optical flow propagation, as described

Figure 5.2: Two example of the ground truth optical flow information provided by the MPI Sintel data set [13] (cropped). The color coding of the optical flow is the same as in Figure 2.13.

in Section 4.1.1, a second variant is implemented using the mesh-based propagation, described in Section 4.1.2. The former is combined with the occlusion-based handling of the structural changes, as described in Section 4.2.2, and the latter is combined with the min-max-size-based handling, described in Section 4.2.1. In the following, the first variant of the framework will be denoted by TCS (occlusion) and the second light-weight version by TCS (mesh). Both variants are implemented using the OpenCV library [11]. The dense optical flow, utilized in the TCS (occlusion) variant, is created using the library provided by [49]. To compute the minimal cut in the case of an overlap during the propagation of the superpixels, the implementation of [41] is utilized.

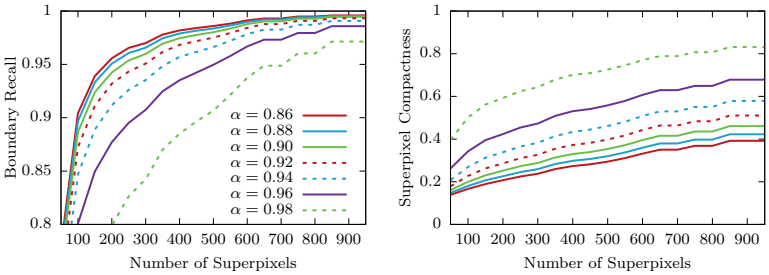
Besides the user selected number of superpixel per frame $|\boldsymbol{k}|$, both variants of the proposed framework contain a common set of parameters controlling certain aspects of the resulting segmentation. On the one hand, this is the configuration of the sliding window (F and P) . And on the other hand, the spatial weight α which in Equation (3.4) regulates the spatial compactness of the temporally consistent superpixels. Further, both variants have exclusive parameters which are namely the min-max-size thresholds ($|\boldsymbol{n}|_{\min}$ and $|\boldsymbol{n}|_{\max}$) for TCS (mesh) and the weighting term $\lambda_{\mathcal{G}}$ for TCS (occlusion). The weighting term $\lambda_{\mathcal{G}}$ of Equation (4.4) has only a modest influence on the consistency of the labeling in the overlapping areas. It is empirically set to a common value of 50 [69]. The size thresholds are set to $|\boldsymbol{n}|_{\min} = 0.5 \cdot \bar{S}^2$ and $|\boldsymbol{n}|_{\max} = 1.5 \cdot \bar{S}^2$. Thereby, $\bar{S}^2$ is the average superpixel size which is derived from the number of superpixels per frame. By selecting these values the number of actually generated superpixels per frame tends to settle around the user selected value. The selection of the remaining parameter values by performing a set of preliminary experiments will be described in the following passage. To avoid any overfitting to the video data, the preliminary experiments are performed on

separate training data which is not used in the final evaluation.

The spatial weighting term α is an inherited parameter from the single image superpixel approach of Schick et al. [74], revisited in Section 2.1.3. It controls the compactness of the created superpixels, as it has been qualitatively visualized in Figure 2.7. For a segmentation to be of high quality, the generated segmentation boundaries should be coherent to the object boundaries visible in the image. On the other hand, it has been argued by [60, 47, 74] that it is beneficial to have compact and equally sized superpixels. It e.g. allows for a better capturing of spatially coherent information. In addition, it simplifies the execution of subsequent processing steps, as e.g. equally sized superpixels tend to have a lower average number of neighbors which eases the evaluation of neighborhood relations. Additionally, further calculations such as feature extraction can be performed on almost equally sized segments. But as it has been stated by Schick et al. in [74], segmentations with a high compactness tend to have a lower boundary recall and vice versa. When the proposed framework is applied to single images instead of video material, it behaves equivalently with the original algorithm proposed in [74]. This is because the sliding window only contains a single frame on which the optimization is done independently, during the frame passes through the *future* and *current* frames. Therefore, the images of the Berkeley image segmentation data set [54] will be initially used to discuss the optimal spatial weight. The data set contains a diverse set of images and provides five ground truth segmentations per image created by different human subjects. For the following experiments the 100 images of the validation subset were utilized.

In Figure 5.3, the 2D boundary recall and the superpixel compactness for the subset are plotted over the number of superpixels per image. For the graphs the spatial weight α was set to different equidistant values from the range between 0.86 and 0.98. The graphs show that increasing the value of the spatial weighting leads to more compact superpixels. But simultaneously, it also decreases the recall of the boundaries, as with higher circularity it gets more difficult to trace the fine grained boundaries of the objects. For the sake of clarity, only the boundary recall and superpixel compactness are plotted, as additional metrics like the variance of area metric in this case basically provide the same information. For this case, the variance of area grows with decreased values of α. The high negative correlation between the boundary recall and superpixel compactness metrics, as it is shown in Figure 5.3, is also described in [74]. As a consequence, the authors argue that the selection of the optimal parameter for single images is highly depending on the intended segmentation application.

To further investigate the influence of the spatial weight on the spatio-temporal segmentation quality, additional experiments on video data are required. Thereby, the sliding window gets filled and additional parameters have to be considered. These are the number of *past* frames P, the number of *future* frames F, as well as



(a) (b)

Figure 5.3: Boundary recall and superpixel compactness for different values of the spatial weight α computed on the validation subset of the Berkeley image segmentation data set [54] plotted over the number of superpixels per image. Segmentations with a high boundary recall simultaneously have a low superpixel compactness.

the number of iterations after each shift Z, as described in Section 3.4. The hybrid optimization assigns the boundary pixels of the superpixels to the best fitting superpixel. During this process only the segmentations of the *future* and *current* frames are altered. Therefore, the selection of F influences the ability of the algorithm to adapt the segmentation to the image content. This is because at all *mutable* positions (i.e. *future* and *current*) the frames are optimized for Z iterations after each shift of the sliding window. Thus, selecting F too small may result in a higher segmentation error, while selecting a too high value unnecessarily increases the computational burden. In [1], the total number of iterations for the SLIC superpixels algorithm is given with 10. As the contour evolving optimization, utilized in the proposed framework, only allows label flips at the superpixel boundaries it converges slower than the original SLIC algorithm. To compensate for the slower convergence, the number of iterations will be set to $Z = 10$ and the number of *future* frames to $F = 2$. This results in a total of 30 iterations performed on each frame before it is shifted into the immutable area of the *past* frames. Although the number of iterations is increased in comparison to SLIC, the total processing time of the optimization is only marginally higher. This is because the contour-based optimization only has to consider the boundary pixels which reduces the number of computationally intensive operations that have to be performed in each iteration.

Fixing the number of *future* frames leaves two remaining parameters to be selected, namely α and P. Their value for the main experiments will be selected by optimizing for the three main benchmark metrics for spatio-temporal segmentation quality, namely the 3D segmentation accuracy (SA), 3D undersegmentation error

(UE) and temporal extent (TEX). To avoid any overfitting on the test data, the optimization is performed on a separate data set, provided by [80]. The Segtrack data set consists of six video clips with up to 73 frames and a binary ground truth segmentation for each frame. The parameter optimization is achieved by performing a grid search in the intervals $\alpha = [0.86, 0.98]$ and $P = [2, 20]$ of the parameter space. For each combination of parameters a segmentation of the whole Segtrack data set in seven levels of coarseness is performed. The levels were selected to be equidistantly distributed in the range of 50 to 950 superpixels per frame. The final benchmark value for each of the three metrics is yielded by averaging over all sequences and levels of coarseness. Thereby, gaining a single value $\bar{\zeta}_m$ for each parameter combination (α, P) and benchmark metric $m \in \{\mathrm{SA}, \mathrm{TEX}, \mathrm{UE}\}$. The sets of values for the different metrics are separately normalized to lie in the interval between zero and one. Further, the scale of the 3D undersegmentation error is inverted and thus higher denoted a better result for all $\bar{\zeta}_m$. The results for the selected range of parameter combinations are depicted as color coded maps in Figure 5.4.

The plot of the 3D segmentation accuracy in Figure 5.4(a) shows that there is a negligible influence of the number of *past* frames on the segmentation accuracy. If, on the other hand, the spatial weight is selected to be too high the segmentation accuracy is decreased severely. This is different for the temporal extent and the 3D undersegmentation error, as it can be seen in (b) and (c) of the Figure 5.4. Here, both metrics show an improvement with increasing compactness of the superpixels. In contrast to this behavior, the influence of the number of *past* frames has an inverse nature. While the temporal extent decreases with a raising number of *past* frames, the segmentation error simultaneously improves. Remember, that the scale of the undersegmentation error is reversed in this passage and thus red means better. This behavior is in line with the purpose of the *past* frames, as it was described in Section 3.4. Since the superpixel segmentations of the *past* frames are unaltered, they preserve the color of the superpixels and thus prevent them from overlapping object boundaries. Up to some extent the effect can be observed in Figure 5.4 (c), until it levels off in the right half of the plot.

In order to choose the optimal set of parameters, all three metrics have to be regarded simultaneously. It is therefore proposed in this thesis to combine the individual terms $\bar{\zeta}_m(\alpha, P)$ in a multiplicative way to form the combined metric $\bar{\zeta}_{\mathrm{total}}$ as follows

$$\bar{\zeta}_{\mathrm{total}}(\alpha, P) = \bar{\zeta}_{\mathrm{SA}}(\alpha, P) \cdot \bar{\zeta}_{\mathrm{TEX}}(\alpha, P) \cdot \bar{\zeta}_{\mathrm{UE}}(\alpha, P). \tag{5.8}$$

It should be noted that the scaling of the terms as well as their combination could have been chosen differently, e.g. a weighted sum where the weights reflect application specific preferences. But in the absence of any specific requirement the multiplicative approach was chosen, to avoid the need for an explicit weighting of

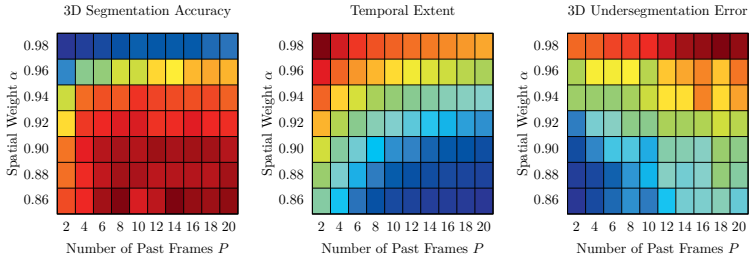


Figure 5.4: Color coded plots for the 3D segmentation accuracy $\bar{\zeta}_{\mathrm{SA}}(\alpha, P)$, temporal extent $\bar{\zeta}_{\mathrm{TEX}}(\alpha, P)$, and 3D undersegmentation error $\bar{\zeta}_{\mathrm{UE}}(\alpha, P)$. For each rectangle the metrics were averaged over all sequences of the Segtrack [80] data set and seven levels of coarseness. The color maps are adjusted to the minimum and maximum values of the normalized metrics. Here, the scale of the 3D undersegmentation error is inverted, so red means better in all plots.

the terms. Thereby, providing a rather general optimal choice of the parameters. The result of the combination is plotted in Figure 5.5. It can be seen that for the chosen scaling and combination the optimal band of parameters lies around $\alpha = 0.94$ to $\alpha = 0.96$ with a tendency to higher numbers of *past* frames. For the following evaluation, the parameters are selected to be $\alpha = 0.96$ and $P = 16$, as they form the maximum value in the chosen representation.

In the remainder of Section 5.1, the two variants of the proposed framework will be further evaluated and compared to three state-of-the-art spatio-temporal oversegmentation approaches from the literature. Namely, the streaming hierarchical video segmentation (sGBH) method of [93], temporal superpixels (TSP) [14], and online video seeds (OVS) [82]. The methods are chosen for the comparison, because all four approaches only process a subset of frames at once. Thereby, all methods are in principle capable of a streaming processing mode, where no simultaneous access to the whole video clip is required. For the evaluation, the implementations provided on the authors' websites were used. Whenever possible, the parameters were set as mentioned in the authors' publications or documentation. Just like the proposed framework, the implementations of TSP and OVS only allow to set the desired number of superpixels per frame to select the granularity of the segmentation. Hence, the resulting number of supervoxels per video sequence can only be influences indirectly. To allow for a fair comparison, the numbers of superpixels per frame were selected in such a way that the number of generated supervoxels is approximately identical for all approaches.

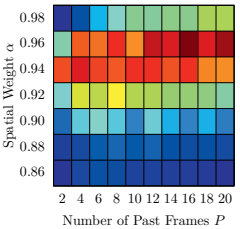


Figure 5.5: Plot of the multiplicative combination $\bar{\zeta}_{\text{total}}(\alpha, P)$ of the three metrics shown in Figure 5.4. The maximum of the combined metric lies in the band from $\alpha = 0.94$ to $\alpha = 0.96$ with a tendency to higher numbers of *past* frames P.

5.1.3 Per Frame Segmentation Quality

This subsection will be focused on the segmentation quality achieved on a per frame level. Therefore, the quality is first assessed in a qualitative and subsequently in a quantitative manner by using the benchmark metrics described in Section 5.1.1. Thereby, only those metrics are considered which do not regard the temporal consistency of the segmentation.

In Figure 5.6 color-coded label maps for the proposed framework and the three state-of-the-art approaches are shown. For each approach a level of detail was chosen showing approximately 1500 superpixels for the left column and about 400 superpixels for the two right columns. The figure only shows label maps for the TCS (occlusion) variant because on the frame level no difference is noticeable between the two variations of TCS.

The most noticeable difference in segmentation characteristics can be seen when comparing the segmentation of TSP and TCS with the segmentation of sGBH and OVS. While the superpixels of the former two algorithms are compact and approximately equally shaped. the two latter ones produce a wide variety of sizes and irregular shapes. The difference between the proposed approach and TSP is less obvious and can be noticed only in smaller details, such as the folds in the shirt of the salesman or the book shelf, where the shapes of TSP are less compact. It should also be noted that OVS as well as TSP produce elongated segments along the boundaries of many objects. This effect is in particular visible in the *ice skating* sequence of Figure 5.6. These elongated segments are as thin as a row of one pixel and will therefore cover a ground truth segmentation boundary on both sides of the segment. This can have huge impact on the boundary recall which is confirmed in the quantitative metrics. To better visualize the difference between the results of the TSP approach and the TCS variant, a magnification of two of the color-coded label maps can be found in Figure 5.7.

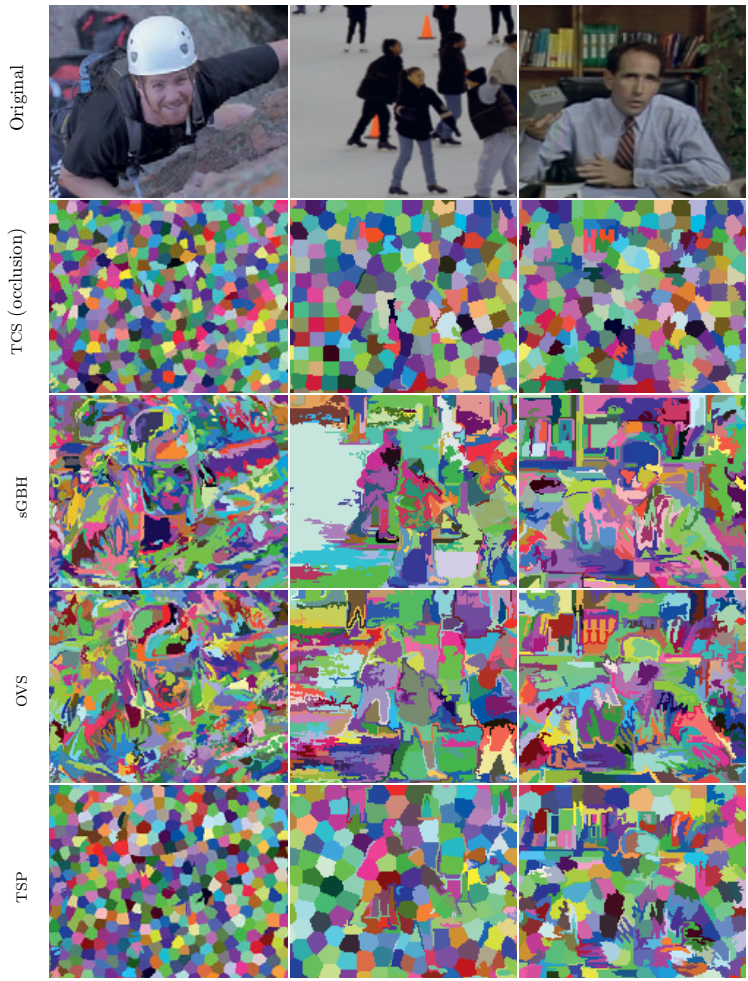


Figure 5.6: Qualitative comparison of color-coded label maps. All frames have approximately 1500 (left column) or 400 (middle and right columns) superpixels (frames are partially cropped for display purposes). Only TCS (occlusion) is shown, because the difference to TCS (mesh) is not noticeable. The label maps show that the proposed TCS method and TSP produce more compact superpixels than sGBH and OVS.

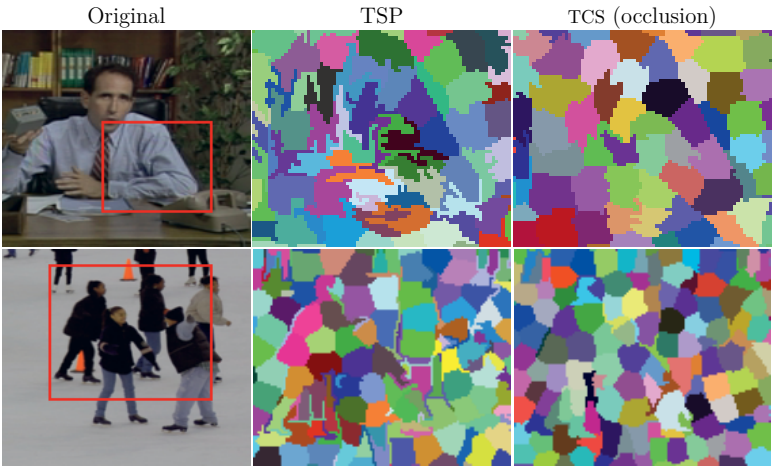


Figure 5.7: Detail view of the color-coded label maps of the *salesman* and the *ice skating* sequence. The label maps on the right show the magnified area of the red rectangle for the TSP and the TCS (occlusion) approach. In the top row the magnification shows that the TSP approach creates irregularly shaped segments in the area of the shirt, where the segments of the TCS variant are more compact. In the bottom row it can be seen that the TSP approach creates many elongated segments along object boundaries which have a misleading impact on the boundary recall metric. Despite the higher boundary recall, the thin segments do not result in a more meaningful segmentation, because it is unclear if the line of pixels belongs to the background or the foreground.

In Figure 5.8, the boundary recall and the superpixel compactness are plotted over the number of superpixels per frame. As it can be expected the boundary recall increases for all approaches with increasing number of superpixels. The plots show the same inverse correlation between the superpixel compactness and the boundary recall as it was already observed in Section 5.1.2. The approaches with less compact superpixel have a higher boundary recall than the approaches with more compact ones. Like it was mentioned earlier, TCS (occlusion) and TCS (mesh) produce segmentations with virtually no difference on the 2D benchmark metrics. The boundary recall for the approaches producing the elongated superpixels along the object boundaries (OVS and TSP) is especially high on Chen's data set. For Sundberg's data set the performance of TSP drops below the performance of TCS (occlusion) and TCS (mesh). The performance of the proposed approach is worst for Chen's data set.

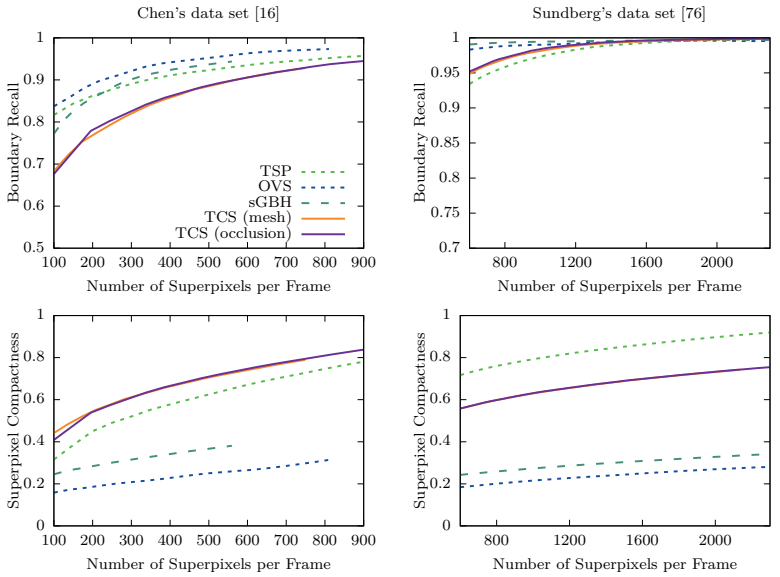


Figure 5.8: Boundary recall and superpixel compactness for different numbers of superpixels evaluated for Chen's [16] (left) and Sundberg's [76] (right) data set. It should be noted that the superpixel compactness as well as the boundary recall of the two TCS methods could be adjusted by changing the spatial weight α, as it was shown in Section 5.1.2.

As it has been shown in the previous subsection, the spatial weighting parameter α could be used to make the algorithm more sensitive to fine-grained details. Thereby, achieving a better boundary recall at the price of a lower compactness.

The visual impression of Figure 5.6, that the superpixels produced by sGBH and OVS vary more in size than the ones produced by TCS and TSP, is confirmed by the quantitative metrics. In Figure 5.9, the variance of area is plotted over the number of superpixel per frame. It can be seen that the superpixels produced by sGBH and OVS have a variance in size of over 2. For Sundberg's data set the variance of sGBH is particularly high and starts at about 6 and linearly increases up to 18. The lowest variance of area is produced by the proposed framework with a variance of about 0.1 on a wide range of superpixel numbers. Only for Sundberg's data set the TSP approach gains a similar variance of area.

Overall, both TCS variants perform equally well in terms of per frame segmen-

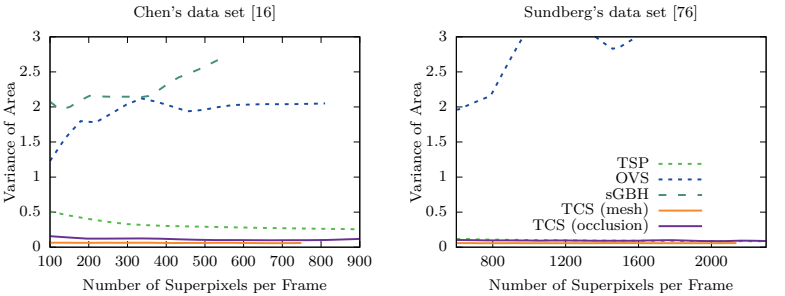


Figure 5.9: Variance of area for different numbers of superpixels evaluated for Chen's and Sundberg's data set. The variance of area curve of sGBH on Sundberg's data set starts at about 6 and linearly ramps up to 18. It therefore exceeds the scale of the graph.

tation quality. In comparison to the other approaches the compactness of the segments is higher in most cases which results in a slightly worse boundary recall. As discussed in the previous section this could be regulated by adjusting the spatial weighting term α. Reducing the value increases the boundary recall but also leads to an increased variance of area and lowered superpixel compactness. Ultimately, the decision on the value of the spatial weight is application specific, as in some applications compactness has higher priority than boundary recall and vice versa.

5.1.4 Spatio-Temporal Segmentation Quality

After assessing the per frame segmentation quality in the previous subsection, this subsection will focus on the spatio-temporal segmentation quality. Therefore, it is first evaluated how well the generated segmentations comply with the ground truth segmentation in all three dimensions of the video. Subsequently, it is assessed how well the segmentations represent the data of the video volume.

For the first part of the evaluation, the 3D segmentation accuracy as well as the 3D undersegmentation error are utilized. Both penalize the crossing of spatio-temporal segmentation boundaries. The metrics are plotted in Figure 5.10 for both data sets which have a ground truth segmentation available. As mentioned in Section 5.1.1 the metrics are plotted over the number of supervoxels. Thereby avoiding that a frame-wise executed superpixel segmentation produces equivalent benchmark results without connecting the superpixels across different frames. For both data sets the 3D segmentation accuracy increases with raising numbers of supervoxels. The only exception is the OVS approach which produces an almost flat accuracy

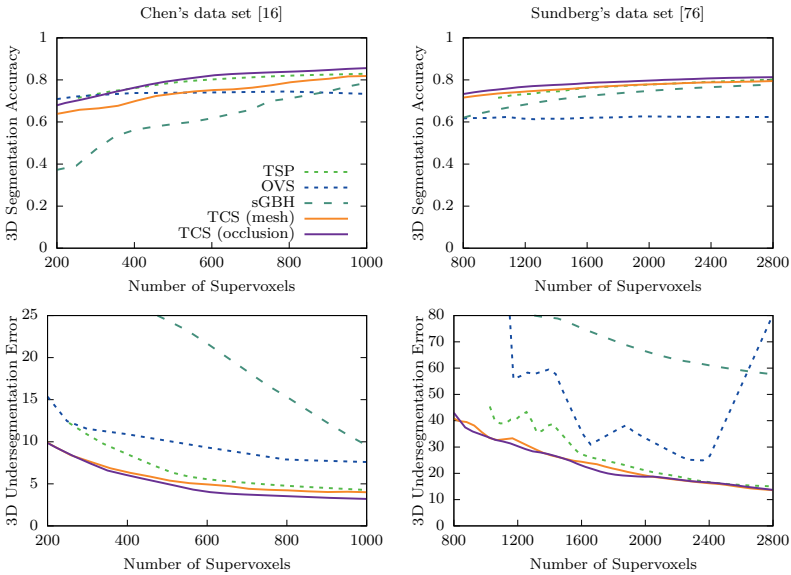


Figure 5.10: 3D segmentation accuracy and 3D undersegmentation error for Chen's [16] (left) and Sundberg's [76] (right) data set. While for the segmentation accuracy higher means better, for the undersegmentation error lower means better. In nearly all supervoxel ranges the proposed TCS (occlusion) method outperforms the other approaches.

graph. Thereby, the achieved accuracy is the worst for Sundberg's data set. The 3D undersegmentation error decreases for all approaches with finer segmentation granularity. The unusual behavior of OVS for Sundberg's data set may have its roots in the implementation provided by the authors. The automatically selected number of histogram bins used for the segmentation is hard coded for several fixed numbers of superpixels. Where the available superpixel numbers are more suitable for Chen's data set. These preset values may not work well on Sundberg's data set, as it has a higher resolution and a larger variety of complexity. This might also be the case for the variance of area metric in Figure 5.9 of the previous section. Both cases reveal a high dependency of OVS's segmentation quality on the selected parameters.

In terms of 3D segmentation accuracy, the TCS (occlusion) variant performs best on both data sets for nearly all ranges of supervoxels. The same holds true for the 3D undersegmentation error on both data sets. TCS (mesh) scores at least second

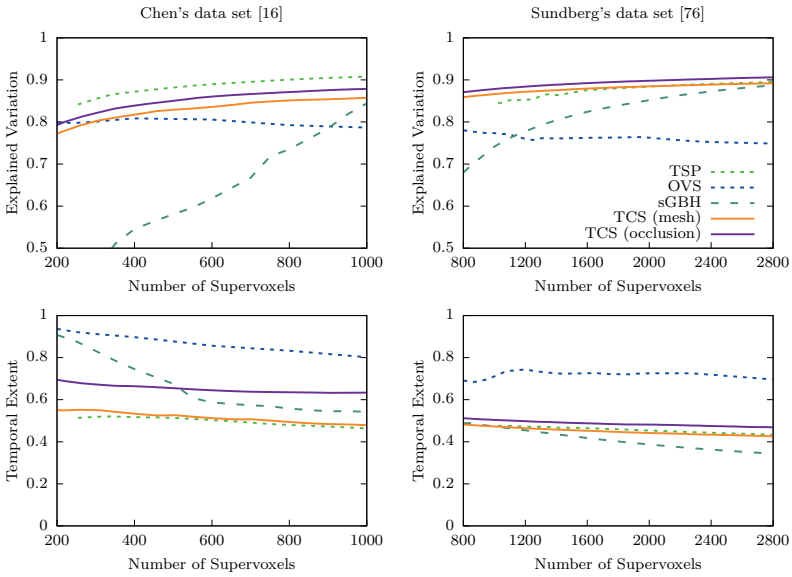


Figure 5.11: The the explained variation as well as the temporal extent of the spatio-temporal segments for Chen's [16] (left) and Sundberg's [76] (right) data set. Although OVS creates the segments with the longest temporal extent, it also produces a high segmentation error as it can be seen in Figure 5.10.

best in terms of 3D undersegmentation error. For Chen's data set its 3D segmentation accuracy is lower than the one produced by TSP and TCS (occlusion). The worst performance in terms of 3D undersegmentation error for nearly all segmentation granularities is shown by the sGBH approach.

Next, it is examined how well a segmentation captures the temporal connections of the image regions inherent in the video volume and how well it can represent the underlying image content. Therefore, the explained variation and the temporal extent of the spatio-temporal segments are plotted in Figure 5.11.

In the plots of the explained variation it can be seen that in general all approaches produce a better explained variation score with an increasing number of supervoxels. Only the score of the OVS approach decreases with a finer segmentation granularity. This tendency can be observed in both data sets. Further, it can be seen that the TSP approach produces the highest explained variation for the small data set of

Chen. For the bigger data set of Sundbergh, the TCS (occlusion) variant performs best while TCS (mesh) produces the second best explained variation for most supervoxel ranges. Here, the TSP variant is only at par with the TCS (mesh) variant for higher numbers of supervoxels.

Besides a low segmentation error, a spatio-temporal segmentation of high quality should capture the temporal connections as completely as possible. Hence, it should produce long enduring segments while having a high 3D segmentation accuracy and a low 3D undersegmentation error. The plots of the temporal extent in the second row of Figure 5.11 show that with increasing number of supervoxels the duration of the spatio-temporal segments in general decreases. For both data sets OVS generates the longest spatio-temporal segments. But as it can be seen in Figure 5.10, it also produces a low 3D segmentation accuracy and a high 3D undersegmentation error. This suggests that many of the supervoxels erroneously cross the spatio-temporal boundaries of the objects. The same holds true for the sGBH approach on Chen's data set. Here, the supervoxels have a longer duration for the coarser segmentation granularities but also a high error and low accuracy. TCS (occlusion), on the other hand, generates temporally long superpixel trajectories while simultaneously producing the lowest error and the highest accuracy of all other approaches for nearly all granularities. The segments of the TCS (mesh) approach have a shorter duration than the ones produces by the TCS (occlusion) variant while their segmentation error is nearly equal.

To summarize, the highest spatio-temporal segmentation quality in terms of 3D segmentation accuracy and 3D undersegmentation error is achieved by the TCS (occlusion) approach. Second best in these metrics for most supervoxel ranges is the mesh-based propagation variant of TCS. The low segmentation error of the two TCS variants is combined with the second and third highest temporal extent of the spatio-temporal segments. This combination can be beneficial in many applications like object tracking and ROI video coding as it better links image regions over time. The latter has e.g. been shown in [56]. Another application which benefits from temporally long segments is the task of interactive video segmentation which will be shown in Section 5.2. The only state-of-the-art approach which constantly creates temporally longer segments is OVS. But the segmentation error and accuracy achieved by this approach are worse for all data sets. This indicates that the segments often erroneously cross spatio-temporal object boundaries. These segmentation errors are especially harmful in the case of applying the method as a preprocessing step. This is because later-stage algorithms, which use the segments of the oversegmentation as their basic building blocks, can not easily recover from these errors afterwards.

A property not yet examined is the compliance of the superpixel flow with the image flow. The following section will therefore qualitatively compare the label

consistency of the two proposed TCS variants. Subsequently, the label consistency will be quantitatively evaluated and compared to the state-of-the-art competitors.

5.1.5 Superpixel Label Consistency

The occlusion-aware handling of the structural changes included in the TCS (occlusion) approach described in Section 4.2.2 should increase the consistency of the superpixel flow with the underlying image flow. To visualize the difference in flow consistency to the size-based handling of the TCS (mesh) variant, a qualitative comparison is shown in the Figures 5.12 and 5.13. The figures show frames of a sequence provided by [79] with a car moving from the right to the left. The top and bottom row show results from TCS (mesh) and TCS (occlusion), respectively. For visualization purposes a subset of the superpixels was manually marked and colored on one frame. The superpixels in the following frames are automatically selected through their temporal connections.

In the top row of Figure 5.12, it can be seen that the TCS (mesh) approach suffers from the *pushed superpixels* effect described in Section 4.2.2. The purely superpixel size-based approach to handle structural changes (see Section 4.2.1) is unaware of the occlusion boundary in front of the car. As the superpixels are not small enough to be terminated, they are pushed across the frame, while the car is moving. With the occlusion-aware handling of the TCS (occlusion) approach, the superpixels are terminated, and the pushing effect induced by the movement of the car is effectively eliminated. Consequently, the superpixel flow better complies with the underlying image flow.

A second instance, where the disocclusion awareness of the TCS (occlusion) approach leads to an increased label consistency, is shown in Figure 5.13. Here, frames from the end of the same sequence are depicted. Again, the results created by TCS (mesh) are shown in the top row and some superpixels are manually selected for the visualization. It can be observed that the superpixels from above the car move down after the car has passed by. This is due to the newly revealed space in the rear of the car. As not enough new superpixels are automatically created in this space, it is occupied by the next neighboring superpixels. These are successively moving down into the newly revealed space during the iterative optimization. In the shown case, the superpixel movement also leads to a segmentation error as one superpixels changes its affiliation from the background to the foreground object. In contrast to that, the TCS (occlusion) approach creates new superpixels in the disclosed image region (not selected for visualization). Thereby, successfully preventing the upper superpixels from moving down. The disoccluded regions as well as the occlusion boundary are detected during the propagation of the superpixel label map. The dis-/occlusion detection, implemented in the TCS (occlusion) variant, is only possible

Figure 5.12: Qualitative comparison of the label consistency of the two TCS variants on a sequence from [79]. A subset of superpixels was manually selected and colored for visualization purposes. It can be seen that the TCS (occlusion) variant correctly deletes the superpixels that are in front of the car whereas they are pushed aside in TCS (mesh).

due to the forward propagation of the individual superpixel shapes. Consequently, an equivalent handling of the structural changes can not easily be integrated into TCS (mesh).

To quantitatively assess the consistency of the superpixel flow and the image flow the approaches are applied to the synthetic MPI Sintel [13] data set. Figure 5.14 shows the resulting label consistency benchmark metric. In all previous experiments, the TCS (occlusion) variant was combined with the high quality optical flow of the Lucas/Kanade meets Horn/Schunck approach [12]. In Figure 5.14, an additional TCS (occlusion) variant is included which utilizes the optical flow produced by the original Horn and Schunck [38] approach. The former is denoted by TCS (occlusion)+LK/HS and the latter by TCS (occlusion)+HS, respectively.

In the trend of the graphs in Figure 5.14 it can be seen that the label consistency in general decreases with finer segmentation granularity. An exception from this is the sGBH approach for which the consistency increases with higher numbers of supervoxels. It asymptotically approaches the performance of TCS (mesh). The steepest decrease and the worst labels consistency in finer supervoxel granularities is shown by OVS. The best performance is reached by TSP which is closely followed by the TCS (occlusion)+LK/HS variant. TCS (mesh) and TCS (occlusion)+HS start

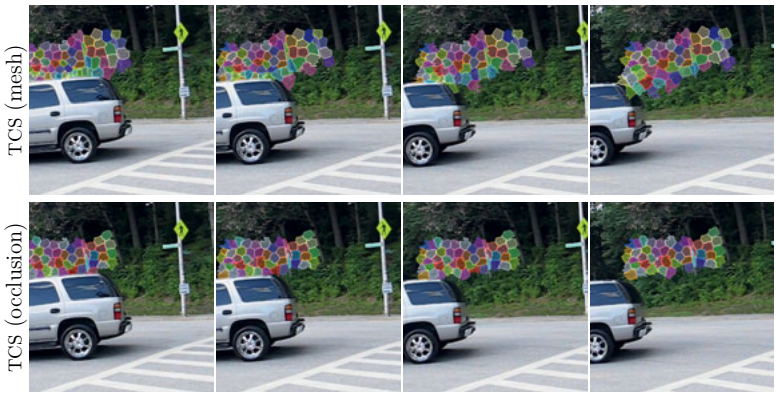


Figure 5.13: Label consistency comparison in case of disocclusion. It can be observed that the superpixels created by TCS (mesh) slip down to fill the newly visible image region behind the car. Simultaneously, the TCS (occlusion) variant reliably creates new superpixels which leads to a better compliance of the superpixel flow with the image flow.

with a label consistency close to TCS (occlusion)+LK/HS but fall off more quickly with finer segmentation granularities.

It should be noted that the two top performers (TSP and TCS (occlusion)+LK/HS) both utilize the high quality optical flow of [12] to propagate the superpixels onto new frames. This fact as well as the difference in performance between the two TCS (occlusion) variants show that the quality of the optical flow utilized for the propagation is of great importance for a high label consistency. A similar result in terms of 3D undersegmentation error is shown by the author of this thesis in [66]. There, the sensitivity to the optical flow quality of different spatio-temporal oversegmentation approaches is assessed. The results demonstrate that the TSP approach shows a significant decrease of spatio-temporal segmentation quality when low quality optical flow is utilized. In contrast to that, the impact on the compared TCS variant is barely noticeable in the experiments of [66].

In this subsection, it was shown that the forward-directed superpixel propagation in combination with the occlusion-aware handling of the structural changes is superior in terms of label consistency to the mesh-based propagation combined with a size-based handling. Further it was shown that the label consistency of the TCS (occlusion) variant is highly competitive to the best state-of-the-art method. Av-

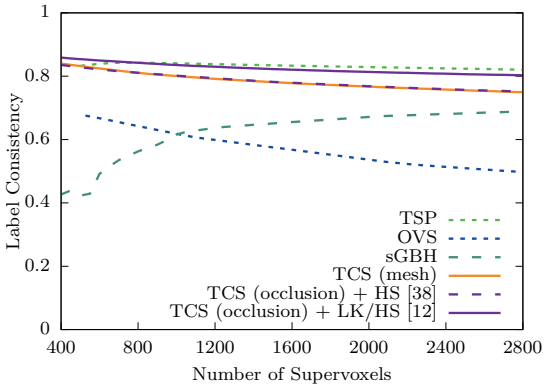


Figure 5.14: Results for the label consistency benchmark on the synthetic MPI Sintel data set provided by [13]. The results shows that the TCS (occlusion) approach improves the consistency of the superpixel flow with the underlying image flow when compared to the TCS (mesh) approach. This improvement is significantly higher when high quality optical flow as produced by [12] is used.

eraged over all segmentation granularities, the label consistency produced by TCS (occlusion) is only 1.2 % worse, in relative terms, when compared to the leading approach in terms of label consistency. The high label consistency of the top performer comes with the price of a higher computational complexity of the optical flow approach. Using the faster optical flow implementation of Horn and Schunck [38] approximately equalizes the label consistency of the two TCS variants. In the following section, the run time and computational complexity of the proposed framework will be further evaluated.

5.1.6 Complexity Considerations

This subsection will discuss the computational complexity of the proposed framework for temporally consistent superpixels. First, the complexity of the hybrid optimization scheme will be examined. Subsequently, the run time of the proposed approach will be experimentally evaluated and compared to the run time of the state-of-the-art approaches. Finally, the run time of the different propagation methods will be discussed.

To approximate the complexity of the hybrid-optimization, a reference to the closely related SLIC superpixel approach can be made. In [1], it is approximated

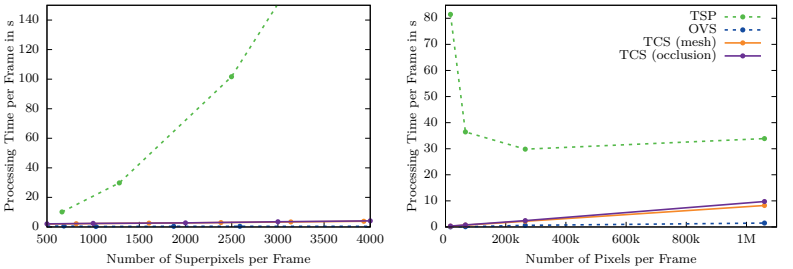


Figure 5.15: Average processing time per frame for the two variant of the proposed framework and two state-of-the-art approaches. Left: Different numbers of superpixels per frame where selected, while the frame resolution is kept constant. Right: An approximately constant number of superpixels per frame was selected (≈ 1000), while the frame resolution was altered.

to have a complexity of $\mathcal{O}(|\boldsymbol{n}|)$ where $|\boldsymbol{n}|$ is the number of pixels in the segmented image. Similarly, the complexity of the contour-based optimization of [74] can be approximated to depend only on the pixel count as ideally each pixel has to be visited only once per iteration. Consequently, the complexity of the hybrid-optimization can be approximated to be $\mathcal{O}(T|\boldsymbol{n}|)$ where T is the total number of frames in the video sequence. This shows that in comparison to the sGBH approach [93] whose complexity is stated with $\mathcal{O}(T|\boldsymbol{n}|\log(|\boldsymbol{n}|))$ the proposed approach has a lower computational complexity. As for the other state-of-the-art approaches no estimation of the complexity is provided by the authors, an experimental comparison is performed in the following. Because the hierarchy levels of the sGBH approach cannot be created separately, but only in a full segmentation hierarchy, the approach is excluded from further experiments.

The remaining competitors from the literature, i.e. TSP and OVS, process the video in a frame-by-frame manner. Hence, they also depend linearly on the number of frames T. In order to reveal the dependencies of the algorithms on the image as well as superpixel resolution, two experiments are performed. In a first experiment, the resolution of the video frames is kept constant, while the number of superpixels per frames is altered. In a second experiment, the number of superpixels is fixed, while the frame resolution is changed. In both experiments, the timing results for TSP as well as TCS (occlusion) are given without the time needed for the dense optical flow calculations utilized to propagate the superpixels. These would distort the dependencies on the different variables and will therefore be examined separately in the end of this section. The additional computational burden of the mesh-based

propagation scheme is negligible and will therefore be included in the following run times. All measurements were performed on equivalent hardware providing two Intel Xeon E5-2690v2 CPUs clocked at 3.00GHz and 128GB of random access memory. For each measurement a full segmentation of the training portion of Sundberg's data set is performed. This portion of the data set provides 40 sequences of approximately 120 frames each with a resolution of up to 1920×1080.

For the first experiment, the video frames are scaled down to half of the original horizontal and vertical resolution. Subsequently, each sequence is segmented into different numbers of superpixels per frame. The superpixel resolution produced by the TCS (occlusion) variant are selected to be 500, 1000, 2000, 3000 and 4000. Similar, sampling points are produces by the TCS (mesh) variant. Although the selected upper bound of the superpixel granularity is much higher than required for most applications, the dependencies on the variable are shown more clearly by going into this extreme scenario. The implementations provided by the authors of TSP and OVS do not allow to set the number of superpixels in an exact way. Hence, the parameters where selected to provide results at approximately equidistant points in an equivalent range. The averaged times needed to segment a single frame are plotted in the left graph of Figure 5.15. While the absolute run times will depend on the particular implementation, it can be seen that OVS as well as both TCS variants have a linear dependency on the number of superpixels per frame. The TSP approach, on the other hand, shows an exponential growth of run time with an increased number of superpixels. Between the two TCS variants only a minor difference can be observed where the mesh-based variant is slightly faster. As both implementation share most of their modules, the difference is due to the computationally less demanding strategy to handle structural changes in the video utilized by TCS (mesh).

For the second experiment, the number of superpixels created by each approach is fixed to around 1000 per frame. The frame resolution is altered by downscaling the original resolution in three steps by cutting the horizontal and vertical resolution in half. The run times on the different resolutions are plotted in the right graph of Figure 5.15. Again it can be seen that the two TCS variants as well as OVS show a linear dependency on the number of pixels per frame and the mesh-based approach of TCS is slightly faster then the occlusion-aware variant. The run time of the TSP approach increases as well with an increased image resolution. But additionally, the run time also grows severely for lower image resolutions. Suggesting that superpixels with a low pixel count are problematic for the TSP approach.

Concluding, it can be said that OVS and TCS scale better than TSP with respect to the image resolution as well as to the superpixel resolution. Although OVS showed the best absolute run time, it should be noted that the final run time will eventually depend on the individual optimization of the implementation. Additionally,

Table 5.1: Average runtime needed to propagate a superpixel label map onto a new frame.

	Average runtime/frame
L/K meets H/S [12]	4757 ms
Horn and Schunck [38]	238 ms
Mesh-based propagation	54 ms

the created segmentations of OVS are of much lower quality as it has been shown in the previous sections. A part of the better segmentation quality and especially the better label consistency of TSP and TCS (occlusion) are a result of their more computational demanding superpixel propagation strategies. To estimate the additional computational burden that can be expected from the dense optical flow [12] utilized by TSP and TCS (occlusion), its average run time is evaluated. Therefore, the average run time needed to process an HD frame of Sundberg's data set [76] is enlisted in Table 5.1. Additionally, the average the run times of the original Horn and Schuck [38] approach and the mesh-based propagation are included in the table. The utilized central processing unit (CPU) implementations of the two dense optical flow methods are provided by [49] and [23]. The table shows that the propagation using the original Horn and Schunck method is nearly 20 times faster than the approach of Bruhn et al. [12]. But it has been shown in the previous section that utilizing the dense optical flow of lower quality also decreases the label consistency to the level of the TCS (mesh) approach. As this approach is over four times faster than the Horn and Schunck method it provides a valuable alternative. But it should be noted that the run time of the optical flow methods can be significantly reduced if a graphics processing unit (GPU) is available. By using the parallelization capabilities of a GPU implementation, such as [35], the run time can be reduced to achieve nearly real-time processing of the frames. The same holds true for the implementation of the hybrid optimization scheme of TCS which currently does not exploit the independent nature of the contour-based optimization. As the mean color values of the superpixels are only recalculated after all contour pixels have been checked for reassignment, the decision on the labeling of each boundary pixel only depends on the current labeling of its eight neighboring pixels. Therefore, the pixels of every second row and column can be processed simultaneously which gives the opportunity for parallelization as it was e.g. done in the single image approach of [28].

5.2 Demonstration: Interactive Video Segmentation

Segmenting a video into foreground and background is a basic step in many computer vision and computer graphics applications, such as object tracking, video editing, or video content analysis [91, 89, 83]. Looking from the user perspective, segmentation algorithms for this task can be divided into two major categories. First, there are fully automatic techniques in which the algorithm by itself determines the portion of an image or video that is the region of interest and should be segmented. Second, the object or region can be specified in a supervised manner, e.g. by a human operator who roughly marks the region of interest. This can be done either by a bounding box [69] or by marking parts of the foreground and background with so-called strokes [10], as depicted in the Figures 5.16 and 5.17. A special form of supervised segmentation is the interactive segmentation, where the user iteratively helps the automatic system to create an accurate segmentation of the region of interest. This is done by repeatedly correcting the segmentation, produced by the system, using additional strokes. As the user has to wait for the response of the system before he or she can do any additional strokes, it is crucial for these frameworks to provide a minimal response time. Two examples of an interactive segmentation task can be found in the Figures 5.18 and 5.19, where a single person should be segmented from the background. In the shown examples, strokes are exclusively provided for the first frame.

In the domain of single images well known approaches were proposed by Boykov et al. [10] and Rother et al. [69]. The segmentation problem is formulated as an energy minimization framework as described in Section 2.1.1 which is solved by finding the MAP solution using graph cut. For low-resolution images (e.g. 1 mega pixels) these algorithms are able to segment a single image on pixel level in a run time less than one second [69]. In the experiments of [10], the approach was also applied on a video sequence of 21 frames at a resolution of 255×189. A then state-of-the-art desktop computer needed several minutes to compute the initial segmentation from the first strokes. The usage of such a slow computer application can be tedious for the operator. In particular, when considering the utilized low resolution of the video material and the HD content which is ubiquitous even in customer products nowadays. As shown by Delong et al. [21] the run time of the min-cut/max-flow algorithm rises rapidly if the data of the problem does not fit into the physical memory. The amount of memory that is needed to segment a 120 frames sequence of HD content (i.e. 1920×1080) with the approach of [10] can easily rise up to 30 GB which is near the upper bound of memory today's desktop computers have built-in. With the emerging market for products which create 4K content the amount of data that needs to be processed will rise accordingly.

In [44] as well as [72], the authors present a variable grouping based on the energy

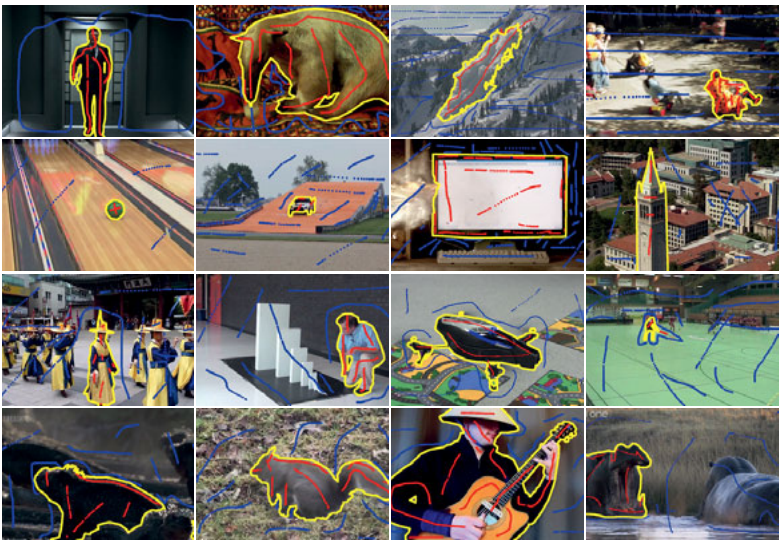


Figure 5.16: Example user strokes for the training set of Sundberg's [76] data set. The objective foreground objects are outlined by the yellow boundaries for visualization purposes. The foreground strokes are colored in red whereas the background strokes are colored in blue. (Frames were cropped for visualization purposes)

function to reduce the original problem size. It is shown that the grouping helps to increase the segmentation quality while decreasing the run time. But this variable grouping can only be applied up to some extent. To overcome this problem, it has become popular to utilize a superpixel segmentation of the input material. A popular representative for this approach is the interactive video cutout of Wang et al. [88]. To boost the performance of the system, they use a two-staged hierarchical mean shift clustering as a preprocessing step. The graph is then build on the oversegmented video volume. This reduces the size of the graph that needs to be processed through the graph cut framework. The system allows the user to paint strokes on single frames as well as along the time axis of the video volume. Additionally, it contains a postprocessing step for creating a spatio-temporally coherent alpha matte to visually optimize the blending of segmented objects onto a new background. In their framework named LIVEcut, Price et al. [59] individually oversegment the video frames using the watershed method [86] and incorporate appearance, motion, and shape features just like the framework proposed by Bai et

Figure 5.17: Additional example user strokes for the training set of Sundberg's [76] data set created by non-expert human subjects. Only for the first frame of each sequence user strokes were generated. (Frames were cropped for visualization purposes)

al. [4] did. While the former uses the graph cut framework like [88] to find a global optimal cut through the graph, the latter uses local overlapping classifiers which are propagated to new frames using optical flow information. More recently, Dondera et al. [24] adopt the framework of [45] to produce temporally more coherent superpixels. Afterwards, a spatio-temporal graph is built-up using optical forward and backward flow information. Their main contribution is to use the information of an occlusion boundary detector to modify the superpixel graph at occlusion boundaries. Subsequently, they partition the spatio-temporal superpixel graph into foreground and background using graph cut similar to [88] and [59].

In the graph cut-based approaches of [59] and [24], the spatio-temporal graph is

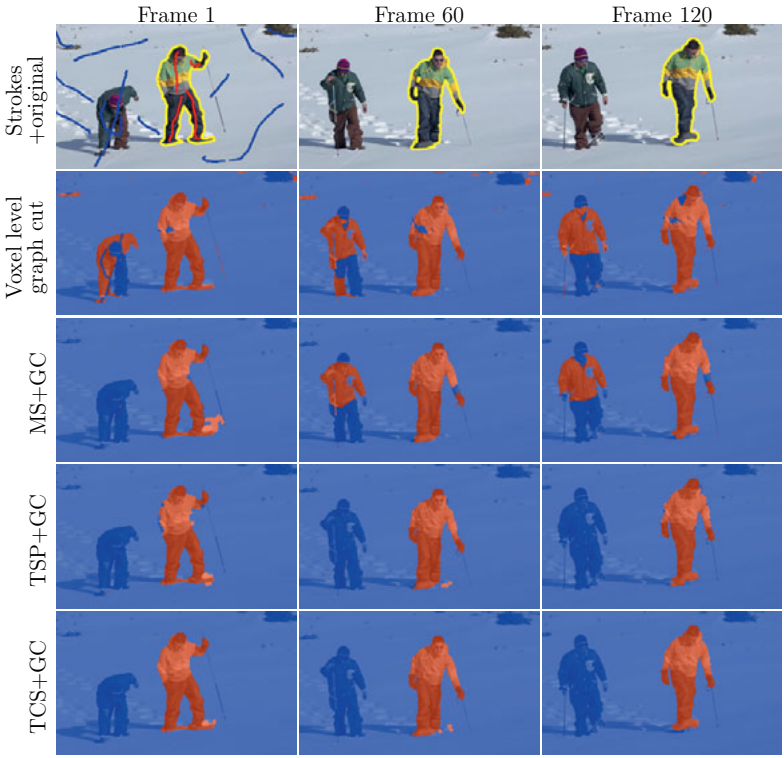


Figure 5.18: Example of an interactive video segmentation. First row: original frames of the *snow shoes* sequence with the overlayed user strokes. The foreground strokes are painted in red whereas the background strokes are colored in blue. The foreground object requested to be segmented by the user is outlined with a yellow boundary for visualization purposes. Second row: segmentation result when graph cut is applied on a voxel level graph. Further rows: results of graph cut applied on a graph built from different types of oversegmention of the input video. (Frames were cropped for visualization purposes)

built from a per frame superpixel segmentation. Thus each node of the graph represents a superpixel whose extent is limited to one frame. While this significantly reduces the graph size when compared to a voxel level graph a further reduction can

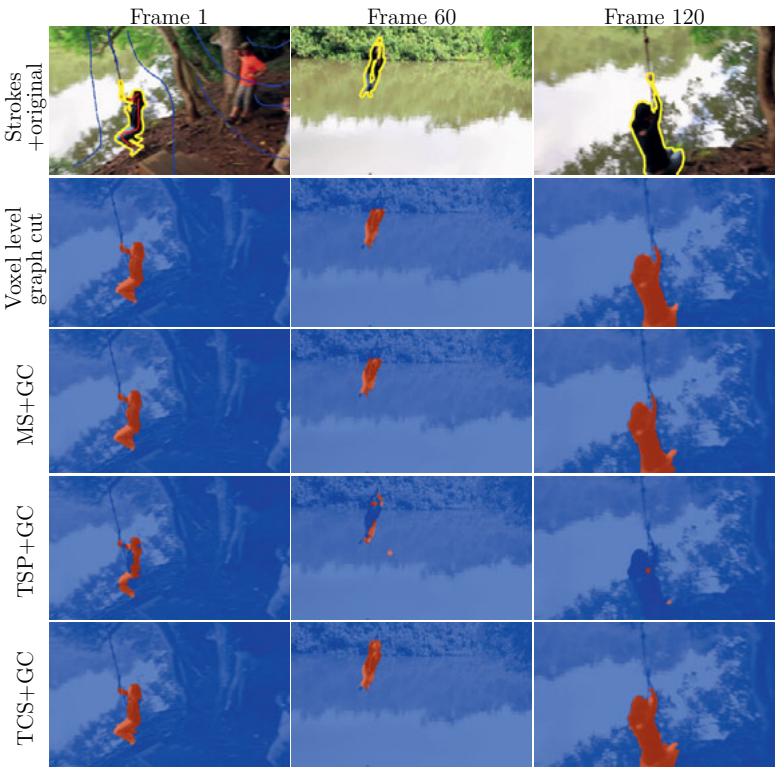


Figure 5.19: Further example of an interactive segmentation. First row: original frames of the *swing* sequence with the overlayed user strokes and main object marked for visualization purposes. Second row: segmentation result when graph cut is applied on a voxel level graph. Further rows: results of graph cut applied on a graph built from different types of oversegmention of the input video. Note that the upper part of the segmented person in the middle frame is only segmented correctly by the combination with the proposed TCS framework. Only the hand is missing. (Frames were cropped for visualization purposes)

be achieved by utilizing the temporal-connectedness of the temporally consistent superpixels as they are proposed in this thesis. Here, each node represents a temporally consistent superpixel which further reduces the number of nodes and edges

in the graph. In this section, it is shown that besides reducing the time needed to find an optimal cut of the graph, the usage of the long-range temporal superpixel trajectories can additionally increase the overall segmentation quality in the application of interactive video segmentation. The rest of this section is structured as follows. In Section 5.2.1 a short overview of a superpixel-based framework for interactive video segmentation is given. Subsequently, Section 5.2.2 will evaluate the usage of the proposed temporally consistent superpixels in comparison to the voxel level approach and other oversegmentation methods.

5.2.1 Interactive Video Segmentation using Superpixels

This section revisits a framework for interactive video segmentation on voxel level as well as the usage of an oversegmentation of the input material. The discussed framework consists of two main components. First, the oversegmentation step which either generates per frame superpixels or supervoxels. The second component builds the spatio-temporal graph which is cut by utilizing the graph cut framework. Thereby, the graph can be either built up on the oversegmentation or on the voxel level to enable a comparison between the two approaches.

The problem of binary segmentation is modeled using the discrete energy function (2.10) introduced in Section 2.1.1. In the voxel level case, the energy function is represented as the sum of unary potentials $D_i(b_i)$, for individual voxels with index i, and pairwise potentials $V_{i,j}(b_i, b_j)$ for neighboring voxels

$$E_{\text{ivs}}(\boldsymbol{b}) = \sum_{i \in \boldsymbol{n}} D_i(b_i) + \lambda_{\mathcal{G}} \sum_{(j,i) \in \boldsymbol{c}_6} V_{i,j}(b_i, b_j). \tag{5.9}$$

Here, $\boldsymbol{b}$ is the labeling of the voxels, $\boldsymbol{n}$ is the set of voxels in the video volume, and $\boldsymbol{c}_6$ corresponds to the set of neighboring voxels. As neighborhood system a six connected neighborhood is utilized, where each voxel is connected to the four neighboring voxels in its own frame and one adjacent voxel in the frames immediately preceding and following it. The weighting term $\lambda_{\mathcal{G}}$ is set equally to the value chosen in Section 5.1.2.

The unary potential $D_i(b_i)$ is defined as the negative log-likelihood using a Gaussian Mixture Model [69] with five kernels

$$D_i(b_i) = -\log p(\vec{x}_i^{\mathcal{C}} | b_i = \tilde{b}). \tag{5.10}$$

Where $\vec{x}_i^{\mathcal{C}}$ is the color vector of the voxel i and $\tilde{b} \in \{FG, BG\}$. The likelihoods are directly learned from the user labeled foreground and background pixels, i.e. from the user strokes. The pairwise potential $V_{i,j}(b_i, b_j)$ is equally defined as in Section 4.2.2:

$$V_{i,j}(b_i, b_j) = \Gamma(b_i, b_j) \cdot \exp(-\beta |\vec{x}_i^{\mathcal{C}} - \vec{x}_j^{\mathcal{C}}|^2). \tag{5.11}$$

The energy function E'_{ivs} which is build upon an oversegmentation of the video volume is defined by a surjective map $m : \boldsymbol{n} \to \boldsymbol{k}$ that maps each voxel to its corresponding supervoxel. Thus, the energy function to build-up the graph from the oversegmentation reads

$$E'_{\text{ivs}}(\boldsymbol{d}) = \sum_{i \in \boldsymbol{n}} D_i(d_{m(i)}) + \lambda_{\mathcal{G}} \sum_{(j,i) \in \boldsymbol{c}_6} V_{i,j}(d_{m(i)}, d_{m(j)}). \tag{5.12}$$

Where $\boldsymbol{d}$ is the labeling of the supervoxels. The mapping of the voxel level edges and nodes to the supervoxel level dramatically reduces the problem size and thus the processing time of the graph cut algorithm.

5.2.2 Segmentation Quality and Runtime Evaluation

In this section, the applicability of the proposed method for temporally consistent superpixels in a framework for interactive video segmentation is evaluated. Therefore, the oversegmentation resulting from the TCS approach is combined with the graph cut framework as described in the previous section. This combination will be denoted as TCS+GC. To assess its performance, it is compared to the voxel level implementation of the framework as proposed in [10] and a version with a graph built-up from frame-wise generated superpixels using mean shift (see Section 2.1.4). Additionally, it is compared with a combination of graph cut and the TSP approach [14]. The combination of graph cut with mean shift and TSP will be denoted with MS+GC and TSP+GC, respectively. The latter was included in the comparison to investigate the benefits of the long-range superpixel trajectories. This is because, TSP achieves a comparable spatio-temporal segmentation quality, while creating shorter superpixel trajectories than TCS. For this demonstration the TCS (occlusion) variant was chosen over TCS (mesh), because it produces longer superpixel trajectories and a slightly better spatio-temporal segmentation quality, as it has been shown in Section 5.1.4. Further, a fair comparison to the TSP approach is enabled, as both approaches rely the high quality optical flow of Bruhn et al. [12]. Finally, it should be noted that the time it takes to preprocess the video into an oversegmentation is of minor importance in the case of an interactive video segmentation. This is because the critical phase begins when the user has painted his or her strokes. As the user needs to see the intermediate segmentation result, before he can put further strokes to refine the segmentation, he has to wait for the graph cut to finish. Thereby, the execution of the graph cut stalls the work flow of the user entirely. Hence, the run time of the graph cut should be minimized in this scenario, while the segmentation quality should not suffer. The preprocessing, i.e. oversegmentation, on the other hand, can be done in advance in an *overnight* fashion as it was phrased by Wang et al. [89]. Consequently, the processing time of the oversegmentation is less critical in this domain.

The usual way to evaluate an interactive video segmentation framework, as e.g. done in [24], is to let users segment video sequences and measure the time and number of interactions they need to reach a satisfying segmentation result. As the purpose of this section is not to propose an all new interactive video segmentation framework, no complete user study was performed for this evaluation. Instead, a different method is applied in which all approaches are fed with the same user strokes. The resulting segmentations of the video volume are than compared to a ground truth segmentation. Thereby, the initial segmentation quality is evaluated which is achieved after a user has drawn his initial strokes. This implicitly assumes that by improving the quality of the initial segmentation an equivalent or better final segmentation can be achieved with fewer user effort. Such an improvement in initial segmentation quality should eventually result in a shorter overall time the user has to be in the loop.

For the evaluation, all approaches are applied to the sequences from the training portion of Sundberg's [76] data set. In contrast to Section 5.1, the HD versions of the sequences were chosen for the experiments described in this section. For each sequence the multi-label ground-truth segmentations provided by [30] are converted into a binary version by manually selecting the labels representing the main object. Each algorithm is set up to create approximately 3000 superpixels per frame which approximately corresponds to approximately 300 pixels per superpixel.

The strokes used to initialize the segmentation framework were created by two non-expert users. Therefore, a custom graphical user interface was designed to enable the users to interactively segment a single frame on pixel level. Only the first frame of each sequence was segmented in the procedure. At no time the users had access to any results of the video segmentation. The users were briefly educated in the operation of the interface and afterwards asked to put strokes on the foreground and background of the shown images. The object of interest was shown to them in a separate space of the interface. After the users provided some initial strokes, the graph cut framework was applied on a 2D pixel level graph to generate an initial binary segmentation of the image. The resulting segmentation was evaluated against the ground truth segmentation. If the misclassification rate for the image was below a previously defined threshold, the strokes were stored for the subsequent experiments. Otherwise, the segmentation result was shown to the user and additional strokes were requested. The process was iterated until the target misclassification rate on the frame was achieved. This method of stroke creation was chosen as it appears to be a realistic scenario for an interactive video segmentation application. There, the user could be first ask to segment the region of interest on one frame before he is enabled to put additional strokes on other frames or along the time axis as in [88]. Examples of the user strokes created for this thesis are depicted in Figure 5.16 as well as Figure 5.17. The target misclassification rate for the creation of the strokes was arbitrarily selected to be 5 %.

The two sets of final strokes were subsequently utilized to generate a segmentation

of the whole video volume using the different combinations of oversegmentations and graph cut. Qualitative results produces by the combinations are depicted in Figure 5.18 and 5.19. In Figure 5.18, it can be seen that the graph cut algorithm, applied on the voxel level graph as well as on the mean shift-based graph, fails to segment the single person in the *snow shoes* sequence. While it seems to be an easy segmentation task, with white background and two distinguishable foreground objects, the second person is segmented as foreground as well. Although the user explicitly marked it as background. The two other approaches perform better on this task. Thereby, the segmentation quality of TCS+GC is slightly better than for TSP+GC. Here, the boundary of the right person is not exactly met in some areas. In the *swing* sequence depicted in Figure 5.19, the advantage of the long-range superpixel trajectories, as created by the proposed framework, can be seen. Through the long-range segments, the head of the swinging person is segmented correctly, while the shorter and incorrect segments of the TSP approach lead to an increased segmentation error when compared to the voxel level approach. Additional qualitative examples can be found in Figure 5.20 and 5.21.

A quantitative comparison of the approaches can be found in Table 5.2. Here, the misclassification rates for all 40 sequences are listed. For each sequence, the misclassification rate was calculated for the frames with available ground truth information. Thereby, the results were averaged over the four available ground truth segmentations per sequence as well as the two sets of user strokes. The last row of the table contains the mean misclassification rate over all sequences. For the *tarantula* sequence, the voxel level graph cut did not successfully finish after 48 h of processing. Therefore, is was excluded from the calculation of the mean misclassification rate as well as the average processing time that can be found at the end of this section. The misclassification rates for this sequence, produced by the oversegmentation-based approaches, are listed in the table only as an addition.

The results of the table show that the misclassification rate produced by the TCS+GC combination is lower than the rate of the other combinations in a majority of cases. Only in a very small number of cases, the rate is increase by TCS+GC when compared to the voxel level approach. In the majority of cases, the increase in error rate is below one percent in absolute terms. The highest increase is for the *gray squirrel* sequence, where the error is increases by 1.85 % in absolute terms. It should be noted that for this sequence all oversegmenation-based approaches increase the error rate. The combination of mean shift and graph cut even leads to an increase by 11.25 % in absolute terms. In other cases, the application of graph cut with an oversegmentation improves the misclassification rate by over 20 % in absolute terms, such as for the *trampoline* and the *zoo* sequence. Overall, the TCS+GC approach produces the lowest misclassification rate of all evaluated combinations.

[1]The voxel level code did not successfully finish for this sequence after 48 h of processing. The sequence is therefore excluded from all mean calculation for all approaches.

Table 5.2: Misclassification rate for the 40 training sequences of [76]. Each approach was initialized with the same user strokes generated for the first frame. The proposed TCS approach performs best for the majority of the sequences.

		Algorithm			
		Voxel level graph cut	MS+GC	TSP+GC	TCS+GC
Sequences	alec baldwin	**0.48**	0.54	1.86	0.84
	anteater	4.79	3.71	3.71	**3.62**
	avalanche	10.11	3.11	2.51	**2.11**
	big wheel	2.47	**1.71**	1.73	1.73
	bowling	**0.34**	0.47	0.61	0.39
	campanile	8.80	8.25	4.49	**2.40**
	car jump	2.62	1.92	2.46	**1.88**
	chrome	20.95	14.60	11.68	**11.23**
	deoksugung	1.50	1.95	0.53	**0.47**
	dominoes	4.39	**2.69**	2.99	2.71
	drone	1.83	1.71	1.90	**1.52**
	excavator	5.54	4.92	3.54	**3.14**
	floorhockey	4.03	2.10	**1.35**	2.07
	galapagos	27.00	15.43	10.33	**9.18**
	gray squirrel	**5.66**	16.91	8.85	7.51
	guitar	2.77	3.20	2.54	**1.84**
	hippo fight	16.18	7.21	3.49	**2.34**
	horse riding	2.57	2.05	2.24	**1.70**
	juggling	2.77	1.68	1.70	**1.59**
	kia commercial	10.29	3.44	**3.37**	5.52
	knot	**2.45**	3.98	4.28	3.07
	lion2	14.71	14.20	13.99	**13.51**
	lion	2.12	1.68	1.66	**1.64**
	lukla airport	0.84	**0.45**	0.56	0.51
	pouring tea	4.20	3.59	2.92	**1.51**
	rock climbing	1.11	1.28	1.55	**0.94**
	roller coaster	**16.77**	19.78	17.87	17.45
	rolling pin	8.59	3.70	3.89	**3.07**
	sailing	1.92	3.29	1.12	**0.98**
	sea snake	28.20	28.65	28.68	**27.52**
	sea turtle	3.45	**2.15**	3.91	3.05
	sitting dog	**5.09**	12.79	5.79	6.02
	snow shoes	5.14	3.40	**0.51**	2.29
	soccer	2.33	0.29	**0.14**	0.15
	space shuttle	18.05	15.31	**9.75**	10.25
	swing	0.55	0.78	0.89	**0.42**
	tarantula	–[1]	10.15	**9.95**	10.01
	tennis	3.26	2.07	**1.22**	1.59
	trampoline	25.78	6.03	**1.37**	1.62
	zoo	32.43	15.98	**3.27**	3.57
	mean	8.00	6.08	4.49	**4.18**

Table 5.3: Average processing time per sequence needed to execute the graph cut. During this time, the user has to wait before he can put more strokes on the frames of the video sequence.

Algorithm	**Average time/sequence**
Voxel level graph cut	1875.15 s
MS+GC	0.648 s
TSP+GC	0.0058 s
TCS+GC	0.0066 s

In comparison to the voxel level graph cut the error rate is improved by over 47 % in relative terms.

In addition to the improvement in segmentation quality, a further benefit of the temporally consistent superpixels is the improved execution time of the graph cut. This can be seen in Table 5.3, where the processing times required to find the minimal cut in the graph are listed. The times are averaged over all sequences (except for the *tarantula* sequence). The given durations approximately correspond to the time a user would have to wait before another set of strokes could be drawn. The benchmark was performed on a dual Intel Xeon E5-2690 @ 3.00GHz with 128 GB of random access memory to be able to hold the huge amount of data necessary for the voxel level graph cut approach. The table shows that the average time, a user would have to wait for the voxel level graph cut to finish, is over 31 minutes. This renders the approach virtually inapplicable for HD sequences. It further shows that the TCS+GC combination accelerates the processing time of the graph cut by a factor of nearly 100 when compared to the mean shift combination. This is due to the further reduction of the original problem size as the temporal connections of the superpixels over multiple frames are exploited. The same holds true for the TSP approach. Of course the processing time of the oversegmentation will need to be considered as well when it is integrated into an application. But as stated by Wang et al. [88] the preprocessing can be done *overnight*. Thereby, the critical phase is when the user has drawn his or her strokes and waits for the intermediate segmentation results. Nevertheless, the average run time of the preprocessing will be given for completeness in the following.

The average processing time of the proposed TCS method is 22.4 minutes per sequence. The mean shift segmentation (using the implementation of [84]) and the TSP approach need 22.8 minutes and over 7 hours per sequence, respectively. It should be noted that for the mean shift as well as for the TCS approach, the measurements where performed using single-threaded applications. But both methods have the potential to be accelerated using a multi-threaded or a specialized GPU implementation. The same holds true for the optical flow calculation, as it was al-

ready mentioned in Section 5.1.6. Therefore, the processing time of the optical flow was excluded from the measurement. The time measurements show that only the overall processing times of MS+GC and TCS+GC sum up to a lower value than the processing time of the voxel level graph cut. The TSP+GC combination on the other hand significantly increases the overall processing time. As a conclusion it can be said that its better run time performance and its higher segmentation quality make the TCS+GC combination the preferable choice for an interactive video segmentation.

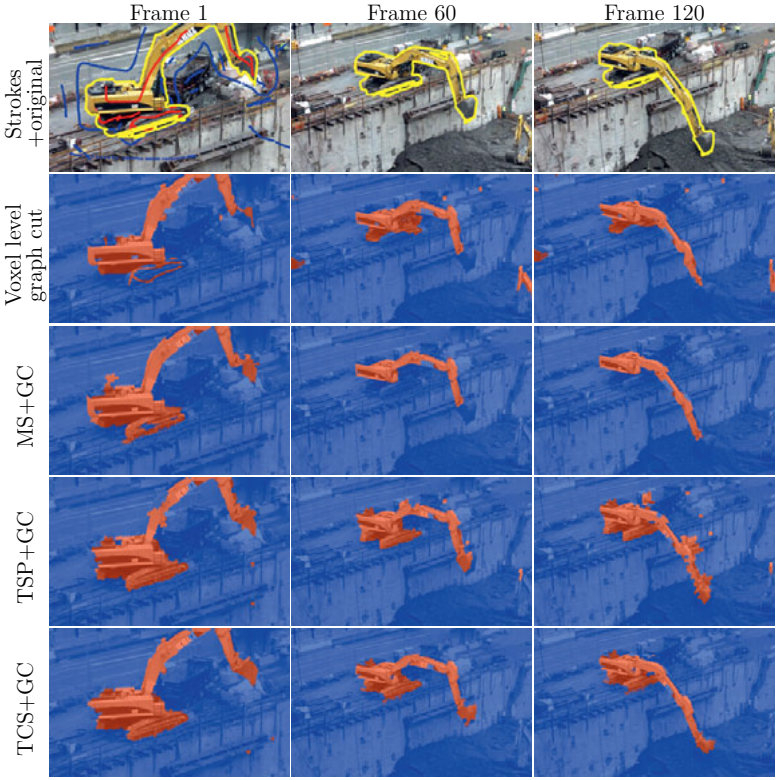


Figure 5.20: Segmentation example showing the results on the *excavator* sequence for all approaches. Note that in the approach utilizing the voxel level graph the chains, the shovel, and the upper part of the main body of the excavator are not segmented correctly. (Frames were cropped for visualization purposes)

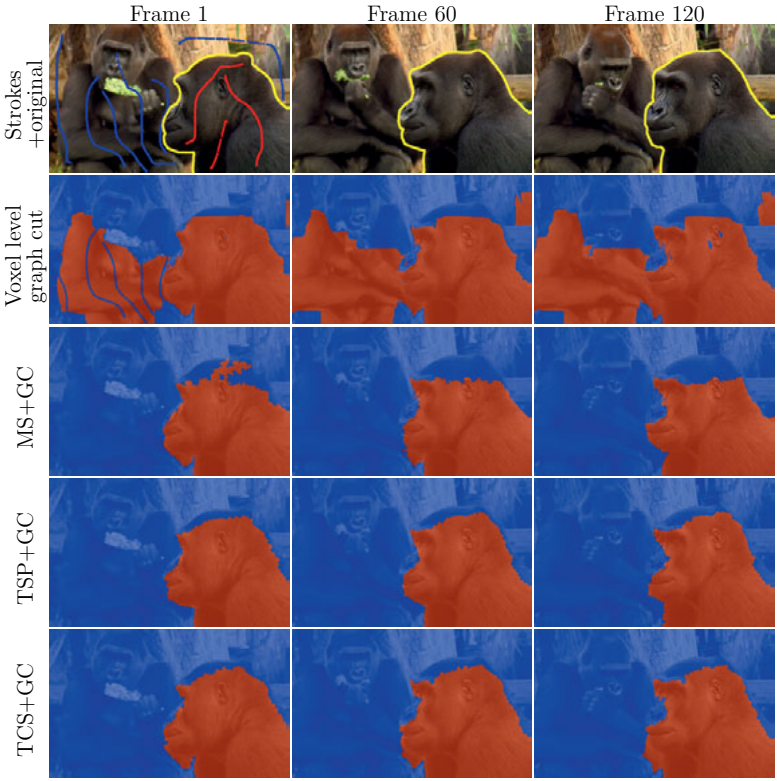


Figure 5.21: Further segmentation example showing the *zoo* sequence. Here, the voxel level approach fails on assigning the second gorilla to the background. The mean shift-based approach correctly segments it as background but looses the upper part of the gorilla's head.(Frames were cropped for visualization purposes)

Chapter 6

Conclusions

The huge amount of pixel data that needs to be handled when processing video material has made it a common preprocessing step, in todays video processing pipelines, to create an oversegmentation of the input material. These intermediate representations group single pixels into larger image primitives often referred to as superpixels or supervoxels. Thereby, the number of image primitives that need to be handled during the execution of the main algorithm is reduced massively. This saves processing time and minimizes the memory footprint. It further enables the extraction of region-based features, such as color or optical flow histograms. As the boundaries of the created regions comply with the visible contours in the image, the provided spatial support for the features is automatically adapted to object boundaries.

Among the fastest and best performing approaches for single images is a class of clustering-based superpixel algorithms. The algorithms model each superpixel by the mean feature vectors of its assigned pixels in a five dimensional feature space. The feature vectors consist of the concatenation of the color dimensions with the two spatial dimensions of the pixels. For the optimization of the segmentation, a cost function is defined which minimizes the variance of the differences between the mean and the pixel feature vectors. The optimization is performed by alternatively assigning the pixels to the best fitting superpixel model and a subsequent update of the model. Due to the simple model and the efficiency of the optimization strategy, the computational burden of these methods is low. In previous works from the literature, it was tried to extent these image-based approaches to the video domain by extending the feature space with a temporal dimension. To integrate the temporal dimension into the optimization, the cost function is extended by a term penalizing a large variance of the spatio-temporal segments in the temporal direction. Due to the nature of this term, short-range spatio-temporal segments are preferred, similar to the compactness enforced by superpixel algorithms in the spatial domain.

This thesis extends the image-based approach to the video domain while extracting as much of the temporal connections of the image regions inherent in the video volume as possible. Simultaneously, the spatio-temporal segmentation error is minimized. The first contribution of this thesis is the introduction of a novel model to

represent the temporally consistent superpixels which allows for long-range spatio-temporal segments. The proposed hybrid model features a global color center for each superpixel, that is shared among all frames, and one local spatial center per frame. The former has the effect that the superpixels stick to an image region with the same color over time. The latter allows the superpixels to retain their compact shape on frame level while keeping their ability to adapt to fast object movements. A contour evolving optimization ensures the spatial coherency of the superpixels during the whole optimization process.

To be able to process the video material in a streaming fashion, a sliding window is employed which comprises only a limited number of frames at all times. As the sliding window is shifted successively over the video volume, new frames which enter the window have to be initialized. During the initialization of the frames, the propagation of the superpixel segmentations onto the new frames is important for a correct and consistent segmentation of the video volume. The second contribution of this thesis is the introduction of two new approaches for the propagation of superpixel segmentations onto new frames. The first approach constructs a triangle mesh across the frame using a sparse set of tracked features points. Through the tracking of the feature points the triangle mesh is deformed. By morphing the underlying superpixel segmentation according to the deformation of the mesh, the superpixels are warped to roughly fit the new frame content. The second approach propagates the individual shapes of the superpixels by using high-quality, dense optical flow. By propagating the whole superpixel shapes in a forward-directed manner, the possibility arises to directly detect overlap and openings in the propagated superpixel segmentation. These occurrences are seen as indicators for structural changes in the scene induced by occlusion and disocclusion. During the propagation, the indicators are collected and utilized in a later processing stage to adapt the spatio-temporal segmentation to the structural changes.

In order to cope with the structural changes, this thesis further proposes two different methods to adapt the superpixel segmentation by terminating and creating new superpixels. If the structural changes are not handled properly, segmentation errors are introduced and the consistency of the superpixel flow with the underlying image flow decreases. Through the detection of overlap and openings in the propagated superpixel segmentation, the proposed framework is able to decide which superpixel gets occluded over time and where new image content is disclosed. To determine the exact occlusion boundary of the overlapping regions, it is proposed to apply the graph cut segmentation framework on these areas. By deleting the superpixels, which are completely occluded according to the observed overlap, the method improves the compliance of the superpixel flow with the underlying image flow and ensures homogeneously sized superpixels. Due to the loss of the ability to detect the overlap and gaps in the case of the triangle mesh-based superpixel propagation, a second approach for the handling of structural changes is proposed in this thesis. The method is purely based on the superpixel size and can therefore be

combined with any propagation method. While this approach proves to be efficient and provides a competitive spatio-temporal segmentation quality, it decreases the consistency of the superpixel flow with the underlying image flow.

In a thorough evaluation, the two variants of the framework, denoted with TCS (occlusion) and TCS (mesh), are compared to three state-of-the-art streaming-capable approaches for spatio-temporal oversegmentation. The evaluation is performed on the common data sets and by applying established benchmark metrics. These include metrics for the per frame segmentation quality as well as the spatio-temporal segmentation quality. Further, the consistency of the superpixel flow with the underlying image flow is regarded. The parameters of the proposed method are jointly optimized for multiple objectives on independent training material. The experiments show that the two variants of the proposed framework achieve a competitive per frame segmentation quality, while TCS (occlusion) outperforms the state of the art in terms of spatio-temporal segmentation quality. The evaluation shows further that both proposed TCS variants produce temporally longer segments than competitive approaches while achiving lower or competitive spatio-temporal segmentation error. State-of-the-art approaches with temporally longer segments severely increase the segmentation error which indicates that they overlap spatio-temporal object boundaries. The TCS (occlusion) variant achieves a label consistency which is highly competitive with the best state-of-the-art method from the literature. While producing a label consistency which is on average only 1.2 % worse then the result of the leading approach in terms of label consistency, its run time scales better as it only depends linearly on the number of pixels as well as superpixels. It is shown that the high label consistency depends on the high-quality of the utilized optical flow. Without that the label consistency decreases to the level of the TCS (mesh) variant which is still higher than two other approaches from the literature. As the spatio-temporal segmentation error of the TCS (mesh) variant is untouched by the lower label consistency it provides a useful alternative for many applications that do not rely on label consistency.

Finally, the advantage of the increased temporal length of the superpixel trajectories is shown in a demonstration by applying the proposed framework to the task of interactive video segmentation. For the evaluation a data set is created which features user strokes for the first frame of 40 video sequences. During the evaluation, the strokes are utilized to initialize the interactive segmentation framework with different oversegmentations as a basis. The experiments show that through the usage of the long-range temporally consistent superpixels, proposed in this work, the segmentation quality and the processing time are tremendously improved. It is shown that by using TCS as a basis, the segmentation error is decreased by over 47 % in relative terms when compared to a voxel level graph. Compared to the second best performing oversegmentation approach from the literature, the segmentation error is still decreased by about 7 % in relative terms. Simultaneously, the

approach computes the oversegmentation in only 22.4 minutes per sequence, while the comparison approach needs over 7 hours per sequence and scales worse than TCS. Additionally, the average time that is needed to compute the minimal cut in the graph cut framework is reduced from over 31 minutes to under 7 ms through the usage of TCS. When compared to a per frame applied mean shift oversegmentation, the computation of the graph cut is over 98 times faster with TCS. This is due to the substantial reduction of the graph size.

Concluding it can be said that the proposed framework for temporally consistent superpixels achieves better or competitive segmentation performance than all state-of-the-art approaches. It better reveals the temporal connections of the image regions inherent in the video volume while reducing the segmentation error to a new minimum. Simultaneously, the computational time is on the lower end compared to the state-of-the-art algorithms and scales only linearly with the number of pixels and superpixels. Previous algorithms with a comparable segmentation quality are much slower and scale worse than the proposed one. Future extensions of the framework could improve the label consistency of the TCS (mesh) variant by detecting occlusion and disocclusion as in the TCS (occlusion) variant of the framework. Further a hierarchical optimization strategy as utilized in other state-of-the-art algorithms could be used to further optimize the computational performance of the framework.

Bibliography

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. SLIC Superpixels Compared to State-of-the-art Superpixel Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.

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

[3] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour Detection and Hierarchical Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, May 2011.

[4] X. Bai, J. Wang, D. Simons, and G. Sapiro. Video snapcut: Robust video object cutout using localized classifiers. In *Proceedings of SIGGRAPH Asia*, pages 70:1–70:11, New York, NY, USA, 2009.

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

[6] J. Bigun, G. H. Granlund, and J. Wiklund. Multidimensional orientation estimation with applications to texture analysis and optical flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(8):775–790, August 1991.

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

[8] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[9] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004.

[10] Y. Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Proceedings of the IEEE*

International Conference on Computer Vision, volume 1, pages 105–112. IEEE, 2001.

[11] G. Bradski. The opencv library. *Dr. Dobb's Journal of Software Tools*, 2000.

[12] A. Bruhn, J. Weickert, and C. Schnörr. Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61(3):211–231, 2005.

[13] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *Proceedings of the European Conference on Computer Vision*, pages 611–625, October 2012.

[14] J. Chang, D. Wei, , and J. W. Fisher III. A video representation using temporal superpixels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2051–2058, June 2013.

[15] P. Charbonnier, L. Blanc-Feraud, G. Aubert, and M. Barlaud. Two deterministic half-quadratic regularization algorithms for computed imaging. In *Proceedings of the IEEE International Conference on Image Processing*, volume 2, pages 168–172, November 1994.

[16] A. Y. C. Chen and J. J. Corso. Propagating multi-class pixel labels throughout video frames. In *Proceedings of the Western New York Image Processing Workshop*, pages 14–17, Novmber 2010.

[17] Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799, August 1995.

[18] D. Comaniciu and P. Meer. Robust analysis of feature spaces: color image segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 750–755, June 1997.

[19] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[20] B. Delaunay. Sur la sphère vide. *Bulletin of Academy of Sciences of the USS*, 7:793–800, 1934.

[21] A. Delong and Y. Boykov. A scalable graph-cut algorithm for n-d grids. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2008.

[22] A. Djelouah, J. S. Franco, E. Boyer, F. L. Clerc, and P. Pérez. Multi-view object segmentation in space and time. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2640–2647, December 2013.

[23] P. Dollár. Piotr's Computer Vision Matlab Toolbox (PMT). http://vision.ucsd.edu/ pdollar/toolbox/doc/index.html.

[24] R. Dondera, V. Morariu, Y. Wang, and L. Davis. Interactive video segmentation using occlusion boundaries and temporally coherent superpixels. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, pages 784–791, 2014.

[25] L. Duan and F. Lafarge. Image partitioning into convex polygons. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3119–3127, 2015.

[26] P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[27] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8(3):399–404, 1956.

[28] O. Freifeld, Y. Li, and J. W. Fisher III. A fast method for inferring high-quality simply-connected superpixels. In *Proceedings of the IEEE International Conference on Image Processing*, pages 2184–2188, 2015.

[29] K. Fukunaga and L. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory*, 21(1):32–40, 1975.

[30] F. Galasso, N. S. Nagaraja, T. J. Cárdenas, T. Brox, and B. Schiele. A unified video segmentation benchmark: Annotation, metrics and analysis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3527–3534, December 2013.

[31] F. Galasso, R. Cipolla, and B. Schiele. Video segmentation with superpixels. In *Proceedings of the Asian Conference on Computer Vision*, pages 760–774, 2012.

[32] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984.

[33] D. M. Greig, B. T. Porteous, and A. H. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 271–279, 1989.

[34] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph-based video segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2141–2148, 2010.

[35] P. Gwosdek, H. Zimmer, S. Grewenig, A. Bruhn, and J. Weickert. A highly efficient gpu implementation for variational optic flow based on the euler-lagrange framework. In *Proceedings of the European Conference on Computer Vision Workshops*, pages 372–383, 2010.

[36] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the Alvey Vision Conference*, volume 15, pages 147–151, 1988.

[37] D. Hoiem, A. A. Efros, and M. Hebert. Geometric context from a single image. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 1, pages 654–661, 2005.

[38] B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

[39] A. Jain, S. Chatterjee, and R. Vidal. Coarse-to-fine semantic video segmentation using supervoxel trees. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1865–1872, December 2013.

[40] V. Kolmogorov and R. Zabih. Computing visual correspondence with occlusions using graph cuts. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2, pages 508–515, 2001.

[41] V. Kolmogorov and R. Zabin. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–159, 2004.

[42] S. Kumar and M. Hebert. Discriminative random fields: A discriminative framework for contextual interaction in classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1150–1157, 2003.

[43] C. L. Lawson. Software for C^1 surface interpolation. In *Mathmatical Software III*, volume 3, pages 161–194, 1977.

[44] N. Lermé, F. Malgouyres, and L. Létocart. Reducing graphs in graph cut segmentation. In *Proceedings of the IEEE International Conference on Image Processing*, pages 3045–3048, 2010.

[45] A. Levinshtein, C. Sminchisescu, and S. Dickinson. Optimal contour closure by superpixel grouping. *Proceedings of the European Conference on Computer Vision*, pages 480–493, 2010.

[46] A. Levinshtein, C. Sminchisescu, and S. Dickinson. Spatiotemporal closure. *Proceedings of the Asian Conference on Computer Vision*, pages 369–382, 2010.

[47] A. Levinshtein, A. Stere, K. Kutulakos, D. Fleet, S. Dickinson, and K. Siddiqi. Turbopixels : Fast superpixels using geometric flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2290–2297, 2009.

[48] J. Lezama, K. Alahari, J. Sivic, and I. Laptev. Track to the future: Spatio-temporal video segmentation with long-range motion cues. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3369–3376, 2011.

[49] C. Liu. *Beyond Pixels: Exploring New Representations and Applications for Motion Analysis*. PhD thesis, Massachusetts Institute of Technology, 2009.

[50] M. Y. Liu, O. Tuzel, S. Ramalingam, and R. Chellappa. Entropy-rate clustering: Cluster analysis via maximizing a submodular function subject to a matroid constraint. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(1):99–112, Jan 2014.

[51] M.-Y. Liu, O. Tuzel, S. Ramalingam, and R. Chellappa. Entropy rate superpixel segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2097–2104, June 2011.

[52] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[53] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, volume 2, pages 674–679, 1981.

[54] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2, pages 416–423. IEEE, 2001.

[55] H. Meuel, M. Munderloh, and J. Ostermann. Low bit rate roi based video coding for hdtv aerial surveillance video sequences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 13–20, 2011.

[56] H. Meuel, M. Reso, J. Jachalsky, and J. Ostermann. Superpixel-based segmentation of moving objects for low bitrate roi coding systems. In *Proceedings of the IEEE International Conference on Advanced Video and Signal-Based Surveillance*, pages 395–400, August 2013.

[57] A. Moore, S. Prince, J. Warrell, U. Mohammed, and G. Jones. Superpixel lattices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[58] F. Perbet, B. STENGER, and M. Atsuto. Homogeneous Superpixels from Markov Random Walks. *IEICE TRANSACTIONS on Information and Systems*, 95(7):1740–1748, 2012.

[59] B. L. Price, B. S. Morse, and S. Cohen. Livecut: Learning-based interactive video segmentation by evaluation of multiple propagated cues. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 779–786, 2009.

[60] X. Ren and J. Malik. Learning a classification model for segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 1, pages 10–17, October 2003.

[61] X. Ren and J. Malik. Tracking as repeated figure/ground segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[62] X. Ren. Local grouping for optical flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[63] M. Reso, J. Jachalsky, B. Rosenhahn, and J. Ostermann. Temporally consistent superpixels. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 385–392, December 2013.

[64] M. Reso, B. Scheuermann, J. Jachalsky, B. Rosenhahn, and J. Ostermann. Interactive segmentation of high-resolution video content using temporally coherent superpixels and graph cut. In *Advances in Visual Computing*, pages 281–292. Springer, 2014.

[65] M. Reso, J. Jachalsky, B. Rosenhahn, and J. Ostermann. Superpixels for video content using a contour-based em optimization. In *Proceedings of the Asian Conference on Computer Vision*, pages 692–707, 2014.

[66] M. Reso, J. Jachalsky, B. Rosenhahn, and J. Ostermann. Fast label propagation for real-time superpixels for video content. In *Proceedings of the IEEE International Conference on Image Processing*, pages 902–906. IEEE, 2015.

[67] G. Roig, X. Boix, R. D. Nijs, S. Ramos, K. Kuhnlenz, and L. V. Gool. Active MAP Inference in CRFs for Efficient Semantic Segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2312–2319, 2013.

[68] A. Rosenfeld. Digital topology. *American Mathematical Monthly*, pages 621–630, 1979.

[69] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, 23(3):309–314, August 2004.

[70] C. Rother, T. Minka, A. Blake, and V. Kolmogorov. Cosegmentation of image pairs by histogram matching-incorporating a global constraint into mrfs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 993–1000. IEEE, 2006.

[71] B. Scheuermann and B. Rosenhahn. Slimcuts: Graphcuts for high resolution images using graph reduction. In *Proceedings of the International Conference Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 219–232, 2011.

[72] B. Scheuermann, M. Schlosser, and B. Rosenhahn. Efficient pixel-grouping based on dempster's theory of evidence for image segmentation. In *Proceedings of the Asian Conference on Computer Vision*, volume 7724, pages 745–759, November 2012.

[73] A. Schick, M. Fischer, and R. Stiefelhagen. Measuring and evaluating the compactness of superpixels. In *Proceedings of the International Conference on Pattern Recognition*, pages 930–934, 2012.

[74] A. Schick, M. Fischer, and R. Stiefelhagen. An evaluation of the compactness of superpixels. *Pattern Recognition Letters*, 43:71–80, 2014.

[75] J. Shi and C. Tomasi. Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, June 1994.

[76] P. Sundberg, T. Brox, M. Maire, P. Arbeláez, and J. Malik. Occlusion boundary detection and figure/ground assignment from optical flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2233–2240, 2011.

[77] J. Tighe and S. Lazebnik. Superparsing: scalable nonparametric image parsing with superpixels. In *Proceedings of the European Conference on Computer Vision*, pages 352–365, 2010.

[78] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, 1991.

[79] R. Tron and R. Vidal. A benchmark for the comparison of 3-d motion segmentation algorithms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[80] D. Tsai, M. Flagg, A. Nakazawa, and J. M. Rehg. Motion coherent tracking using multi-label mrf optimization. *International Journal of Computer Vision*, 100(2):190–202, 2012.

[81] M. Van den Bergh, X. Boix, G. Roig, and L. Van Gool. Seeds: Superpixels extracted via energy-driven sampling. *International Journal of Computer Vision*, 111(3):298–314, 2015.

[82] M. Van den Bergh, G. Roig, X. Boix, S. Manen, and L. Gool. Online video seeds for temporal window objectness. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 377–384, 2013.

[83] A. Van den Hengel, A. Dick, T. Thormählen, B. Ward, and P. H. S. Torr. VideoTrace. *ACM Transactions on Graphics*, 26(3):86, July 2007.

[84] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/, 2008.

[85] O. Veksler, Y. Boykov, and P. Mehrani. Superpixels and supervoxels in an energy optimization framework. *Proceedings of the European Conference on Computer Vision*, pages 211–224, 2010.

[86] L. Vincent and P. Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598, 1991.

[87] C. Vogel, K. Schindler, and S. Roth. Piecewise rigid scene flow. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1377–1384, December 2013.

[88] J. Wang, P. Bhat, R. A. Colburn, M. Agrawala, and M. F. Cohen. Interactive video cutout. *ACM Transactions on Graphics*, 24(3):585–594, 2005.

[89] J. Wang, Y. Xu, H.-Y. Shum, and M. F. Cohen. Video tooning. *ACM Transactions on Graphics*, 23(3):574–583, 2004.

[90] P. Wang, G. Zeng, R. Gan, J. Wang, and H. Zha. Structure-Sensitive Superpixels via Geodesic Distance. *International Journal of Computer Vision*, 103(1):1–21, May 2013.

[91] S. Wang, H. Lu, F. Yang, and M.-H. Yang. Superpixel tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1323–1330, November 2011.

[92] C. Xu and J. J. Corso. Evaluation of super-voxel methods for early video processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1202–1209, 2012.

[93] C. Xu, C. Xiong, and J. J. Corso. Streaming hierarchical video segmentation. In *Proceedings of the European Conference on Computer Vision*, pages 626–639, 2012.

[94] J. Zhang, C. Kan, A. G. Schwing, and R. Urtasun. Estimating the 3D Layout of Indoor Scenes and Its Clutter from Depth Sensors. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1273–1280, 2013.

[95] C. L. Zitnick, N. Jojic, and S. B. Kang. Consistent segmentation for optical flow estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2, pages 1308–1315, 2005.

[96] C. L. Zitnick and S. B. Kang. Stereo for Image-Based Rendering using Image Over-Segmentation. *International Journal of Computer Vision*, 75(1):49–65, February 2007.

Curriculum Vitae

Matthias Reso

09.01.1985	geboren in Hannover, Deutschland

Beruf

seit 09/2017	*Research Scientist* bei Fyusion Inc in San Francisco, Kalifornien, USA
12/2016 - 08/2017	Freiberuflicher Entwicklungsingenieur in Coppenbrügge
01/2012 - 11/2016	wissenschaftlicher Mitarbeiter am *Institut für Informationsverarbeitung* an der *Leibniz Universität Hannover*

Studium

2005 - 2011	Studium der Informationstechnik an der Universität Paderborn, Abschluss mit Diplom (Dipl.-Ing.)

Bundeswehr

2004 - 2005	Grundwehrdienst

Schulbildung

2001 - 2004	Fachgymnasium für Technik der Eugen Reintjes Schule in Hameln, Abschluss mit allgemeiner Hochschulreife
1997 - 2001	Albert Einstein Gymnasium in Hameln
1995 - 1997	Orientierungsstufe in Salzhemmendorf
1993 - 1995	Grundschule in Bisperode
1991 - 1993	Grundschule in Neuenfelde

Online-Buchshop für Ingenieure

VDI nachrichten

BUCHSHOP

Online-Shops

Fachliteratur und mehr - jetzt bequem online recherchieren & bestellen unter: www.vdi-nachrichten.com/Der-Shop-im-Ueberblick

Täglich aktualisiert: Neuerscheinungen VDI-Schriftenreihen

Im Buchshop von vdi-nachrichten.com finden Ingenieure und Techniker ein speziell auf sie zugeschnittenes, umfassendes Literaturangebot.

Mit der komfortablen Schnellsuche werden Sie in den VDI-Schriftenreihen und im Verzeichnis lieferbarer Bücher unter 1.000.000 Titeln garantiert fündig.

Im Buchshop stehen für Sie bereit:

VDI-Berichte und die Reihe **Kunststofftechnik:**

Berichte nationaler und internationaler technischer Fachtagungen der VDI-Fachgliederungen

Fortschritt-Berichte VDI:

Dissertationen, Habilitationen und Forschungsberichte aus sämtlichen ingenieurwissenschaftlichen Fachrichtungen

Newsletter „Neuerscheinungen":

Kostenfreie Infos zu aktuellen Titeln der VDI-Schriftenreihen bequem per E-Mail

Autoren-Service:

Umfassende Betreuung bei der Veröffentlichung Ihrer Arbeit in der Reihe Fortschritt-Berichte VDI

Buch- und Medien-Service:

Beschaffung aller am Markt verfügbaren Zeitschriften, Zeitungen, Fortsetzungsreihen, Handbücher, Technische Regelwerke, elektronische Medien und vieles mehr – einzeln oder im Abo und mit weltweitem Lieferservice

Die Reihen der Fortschritt-Berichte VDI:

1 Konstruktionstechnik/Maschinenelemente
2 Fertigungstechnik
3 Verfahrenstechnik
4 Bauingenieurwesen
5 Grund- und Werkstoffe/Kunststoffe
6 Energietechnik
7 Strömungstechnik
8 Mess-, Steuerungs- und Regelungstechnik
9 Elektronik/Mikro- und Nanotechnik
10 Informatik/Kommunikation
11 Schwingungstechnik
12 Verkehrstechnik/Fahrzeugtechnik
13 Fördertechnik/Logistik
14 Landtechnik/Lebensmitteltechnik
15 Umwelttechnik
16 Technik und Wirtschaft
17 Biotechnik/Medizintechnik
18 Mechanik/Bruchmechanik
19 Wärmetechnik/Kältetechnik
20 Rechnerunterstützte Verfahren (CAD, CAM, CAE CAQ, CIM ...)
21 Elektrotechnik
22 Mensch-Maschine-Systeme
23 Technische Gebäudeausrüstung

ISBN 978-3-18-386110-1