Minimum Cost Multicuts for Image and Motion Segmentation

Dissertation
to obtain the degree of
Doctor of Natural Sciences
(Dr. rer. nat.)

submitted by
Amirhossein Kardoost

submitted to the School of Science and Technology
of the University of Siegen
Siegen 2023

Day of Colloquium	10th of February 2023

Dean of the Faculty	Prof. Dr. Holger Schönherr University of Siegen, Germany

Examination Committee

Chair	Prof. Dr. Andreas Kolb
Reviewer 1	Prof. Dr.-Ing. Margret Keuper
Reviewer 2	Prof. Dr.-Ing. Eddy Ilg
	Prof. Dr. Michael Möller

Declaration

I hereby declare in lieu of an oath that I have drawn up the present work without any undue assistance by third parties and without using any aids other than the ones specified. The data and concepts taken, either directly or indirectly, from any other sources have been marked, indicating the source.
The work has not been submitted to any other examining authority neither in Germany nor abroad and neither in the same nor in any similar form.
Use of the services of any PhD mediation institute or of any similar organisation has not been made.
Any family relationship, first-degree relationship, marriage, civil partnership or cohabitation to the proposed members of the PhD Commission do not exist.

submitted by
Amirhossein Kardoost
March 2023

Acknowledgements

Hereby, I would like to express my deepest gratitude to all people who helped me to reach the completion of this dissertation. Especially, I would like to thank my supervisor Margret Keuper for her patience and fantastic support with helpful discussions and taught me how to conduct high-quality research and do critical thinking. I would like to thank all my co-authors for their constructive help and fruitful discussions. I would like to thank Eddy Ilg for agreeing to be the co-examiner of this dissertation. Further, I would like to thank the examination committee.

Thanks to all my colleagues, Kalun Ho, Kiril Gashteovski, Daniel Ruffinelli, and Samuel Broscheit, for a lot of helpful advice and fruitful discussions. I thank Jovita Lukasik for proofreading this dissertation and Hosna Sattar for providing valuable advice during difficult situations.

I want to thank Annette Wiebusch and Christine Zizka for their help with the organizational questions. Further, I would like to thank the funding from the DFG project KE 2264/1-1, which helped me to pursue this project.

I would like to thank my family for all the love and mental support during this journey. My mother was always there for me in all difficult times and helped me to believe in myself. My father always supported me well, and one of his wishes was for me to achieve this level. Without my parents' support, I could have never reached this stage in my life.

I want to thank my loving wife for being the most supportive partner; emotionally, she was there in all the tough times during the writing of this dissertation. I am delighted that I met her, fell in love, and married her, which all happened during the time of this dissertation. I dedicate this dissertation to my wife (Sahar Mirsadri), my mother, and the memory of my father.

Abstract

Clustering and its application in computer vision, such as image, mesh data, video, and motion segmentation, are the main topics we discuss in this dissertation. The clustering of the entities plays a crucial role in higher-level tasks such as action recognition, robot navigation, scene understanding, and 3D reconstruction. One well-known and widely used clustering framework is the minimum cost *lifted* multicut problem. This framework has recently found many applications, such as image and mesh decomposition or multiple object tracking. It addresses such issues in a graph-based model, where real-valued costs are assigned to the edges between entities such that the minimum cut decomposes the graph into an optimal number of segments. Solving the multicut problem is NP-hard and computationally expensive. Therefore, we propose two variants of a heuristic solver (primal feasible heuristic), which greedily generate solutions within a bounded time. Driven by a probabilistic formulation of the minimum cost multicuts, we provide a measure for the uncertainties of the decisions made during the optimization. We argue that access to such uncertainties is crucial for many practical applications and evaluate the proposed uncertainty measure on image and motion segmentation.

To track the object masks in the video, we use low-level cues such as optical flow information and image boundaries and study the importance of such cues in providing competing and high-quality results. While high-end computer vision methods for this task rely on sequence-specific training of dedicated Convolutional Neural Network (CNN) architectures, we show the potential of a variational model based on generic video information from motion and color. The optical flow information is also used for the motion segmentation task, where observable motion in videos can give rise to the definition of objects moving with respect to the scene. This problem is usually tackled either by aggregating motion information in long, sparse point trajectories or directly producing dense segmentations per frame, relying on large amounts of training data. In this dissertation, we address the problem with the sparse motion trajectories and emphasize that generic cues such as optical flow information and image boundaries are crucial to address this and similar tasks. The complex motion patterns, such as out-of-plane rotation or scaling movement of the objects, add ambiguities to the segmentation problem. Utilizing the hyper-graphs resolve such

ambiguities by modeling translational motion to Euclidean or affine transformations. We evaluate our proposed methods on well-known datasets of the addressed task and show that the integration of the low-level cues improves the result on the higher-level tasks.

Zusammenfassung

Clustering und seine Anwendung in Computer Vision, wie Bild-, 3D Meshdaten-, Video- und Bewegungssegmentierung, sind die Hauptthemen, die wir in dieser Dissertation behandeln. Das Clustering von Entitäten spielt eine entscheidende Rolle bei übergeordneten Aufgaben wie Aktivitätserkennung, Roboternavigation, Szenenverständnis und 3D-Rekonstruktion. Ein bekanntes und weit verbreitetes Clustering-Verfahren ist das Minimum Cost *Lifted* Multicut Problem. Dieses Framework hat in letzter Zeit viele Anwendungen gefunden, wie z. B. die Zerlegung von Bildern und Meshes oder das Tracking von Objekten. Es behandelt solche Probleme in einem Graph-basierten Modell, bei dem den Kanten zwischen Entitäten reellwertige Kosten zugewiesen werden, sodass der minimaler Schnitt den Graphen in eine optimale Anzahl von Segmenten zerlegt. Die Lösung des Multicut-Problems ist NP-hart und rechenaufwändig. Daher schlagen wir zwei Varianten eines heuristischen Lösers (primal feasible Heuristik) vor, die innerhalb einer begrenzten Zeit Lösungen "greedy" erzeugen. Angetrieben durch eine probabilistische Formulierung des Minimum Cost Multicuts liefern wir ein Maß für die Unsicherheiten der Entscheidungen, die während der Optimierung getroffen werden. Wir argumentieren, dass der Zugang zu solchen Unsicherheiten für viele praktische Anwendungen von entscheidender Bedeutung ist und evaluieren das vorgeschlagene Unsicherheitsmaß im Kontext von Bild- und Bewegungssegmentierung.

Um die Objektmasken im Video zu verfolgen, verwenden wir niedriges Niveau-Hinweise wie optische Flussinformationen und Bildgrenzen und untersuchen die Bedeutung solcher Hinweise für die Bereitstellung konkurrierender und qualitativ hochwertiger Ergebnisse. Während High-End-Computer-Vision-Methoden für diese Aufgabe auf sequenzspezifisches Training spezieller Faltungsneuronales Netzwerk (CNN)-Architekturen angewiesen sind, zeigen wir das Potenzial eines Variationsmodells, das auf generischen Videoinformationen aus Bewegung und Farbe basiert. Die optischen Flussinformationen werden auch für die Bewegungssegmentierung verwendet, bei der beobachtbare Bewegungen in Videos zur Definition von Objekten führen können, die sich in Bezug auf die Szene bewegen. Dieses Problem wird in der Regel entweder durch die Aggregation von Bewegungsinformationen in langen, spärlichen Punkttrajektorien oder durch die direkte Erstellung von dichten Segmentierungen pro Bild angegangen, wobei große Mengen von Trainingsdaten benötigt

werden. In dieser Dissertation befassen wir uns mit dem Problem der spärlichen Bewegungstrajektorien und betonen, dass allgemeine Hinweise wie optische Flussinformationen und Bildgrenzen entscheidend sind, um diese und ähnliche Aufgaben zu lösen. Die komplexen Bewegungsmuster, wie z. B. Rotation außerhalb der Ebene oder Skalierung der Objekte, fügen dem Segmentierungsproblem Unklarheiten hinzu. Die Verwendung von Hypergraphen löst solche Mehrdeutigkeiten durch die Modellierung von Translationsbewegungen mit euklidischen oder affinen Transformationen. Wir evaluieren die von uns vorgeschlagenen Methoden an bekannten Datensätzen der adressierten Aufgabe und zeigen, dass die Integration der niedriges Niveau-Hinweise das Ergebnis bei den höherwertigen Aufgaben verbessert.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

Clustering is the task of grouping entities where the objects in one group (i.e. cluster) are more similar than those in different groups, Galushin and Kudinov [56]. In essence, we are concerned about the clustering problem in this dissertation, where the objects inside the clusters range from pixels in an image (in case of image segmentation) to motion trajectories (in case of motion segmentation). The clustering problems are projected to the graph structure, each node representing the object, and the edges correspond to the affinities between the objects. For instance, neighboring pixels (nodes on the graph) are connected via an edge. The graph's decomposition generates the entities' grouping for addressing problems.

In this dissertation, we study the application of the grouping task on different areas like image, mesh data, a stack of microscope images, motion, and video instance segmentation. Further, we investigate the formulations of relating these problems to the graph structure, study the approaches for decomposition of the graph, and provide uncertainty measures on the results.

1.1 Motivation

Video instance segmentation involves localizing, tracking, and segmenting the object instances through consecutive frames. It is a preliminary technique to handle high-level tasks such as object detection, action recognition, and 3D reconstruction of static and moving objects. In such a task, finding the salient objects and tracking them through consecutive frames is enhanced via appearance cues, pixel position, and color information under the usage of the optical flow information, Wannenwetsch et al. [158], Ilg et al. [65], Weinzaepfel et al. [159].

Mostly used information on the video instance segmentation task is the availability of the optical flow information, such as flow magnitude and direction. We studied the effect of the

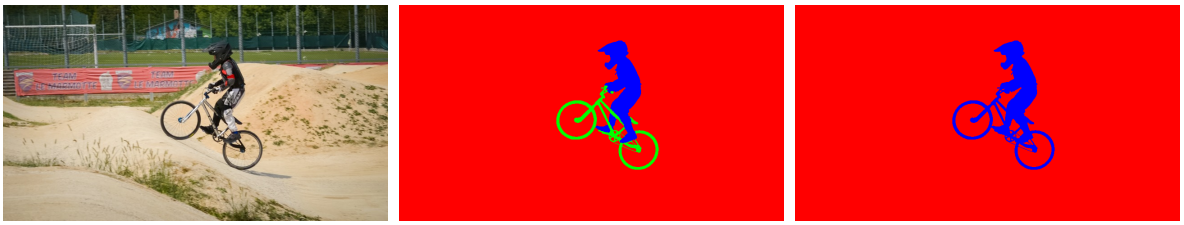


Fig. 1.1 Examples of the video object and motion segmentation are provided from the (**left**) 15th frame of the "bmx-bumps" sequence, where (**middle**) the person and the bike are segmented separately and are considered as separate objects from the DAVIS_{2017} [131]. (**right**) Due to the reason that the person and the bike move together and have the same motion pattern, they are assigned the same motion label (blue color) in the task of motion segmentation where the ground-truth segmentation is provided from DAVIS_{2016} [128].

low-level cues such as optical flow information and the boundary estimates on tracking the instance segmentations through consecutive frames in a semi-supervised manner (see our published work in [84]). Our approach can be combined with state-of-the-art Convolutional Neural Network (CNN)-based segmentations in order to improve their respective results and can even provide competitive results compared to such costly methods. Our method facilitates the segmentation of fine details and thin structures.

In the video object segmentation task, Tsai et al. [153], Price et al. [132], Nagaraja et al. [117], Paul et al. [126], the instance of the salient objects is tracked through the frames. For instance, a person who rides a bike is being tracked through frames, where both person and the bike are considered as two separate objects, see Figure 1.1. In the video instance segmentation task, the object instances are separated and classified into their respective categories. More specifically, each instance is considered as one object; i.e., in a video sequence with several pedestrians to be tracked, each pedestrian is one instance. A similar task to video segmentation is motion segmentation, as addressed in Brox and Malik [25], Li et al. [105], Shi et al. [137], which is a task of segmenting the salient *moving* objects in a video.

According to the Gestalt principle of common fate, Koffka [95], motion patterns of objects are often more homogeneous than their appearance and provide robust cues for moving object segmentation. Thus, from accurately estimated point-wise motions, object motion models can fit through the formulation of a point grouping problem over local motion similarities.

While the information gathered through multiple frames is helpful, handling a large amount of data is costly, mainly due to high computational costs and requiring massive resources to process the versatile data. The less computationally demanding task is image

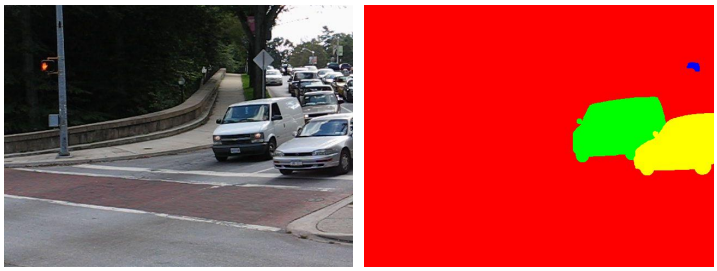


Fig. 1.2 The image and the ground-truth of the motion segmentation of one of the frames on the "cars9" sequence from FBMS59, Ochs et al. [123], are provided. The segments from the motion segmentation result are as large as a whole moving car or the part of the car moving at a distance.

segmentation like Nieuwenhuis and Cremers [119], Bae et al. [8], where the task is to partition an image domain into multiple disjoint components such that each component is a meaningful part of the image. In this task, the ambiguity on the number of salient objects increases due to not having access to enough information compared to the task of a video object and motion segmentation.

One of the well-known frameworks for grouping and clustering is the minimum cost multicut (**MP**) framework (known as correlation clustering), Chopra and Rao [36], Deza et al. [40]. This framework is our main tool used to address the grouping problems where the entities are projected to the vertices in the graph and the edges are connected based on the affinity of the entities (see Section 2.5). This formulation does not require the number of objects as a priori and provides that by the nature of the formulation. Further, the formulation does not favor providing balanced or specific types of segments, meaning that the segments can be as small as part of an object, see Figure 1.2.

The generalization of the **MP** is proposed in Keuper et al. [89], considering long-range terms, named minimum cost *lifted* multicut (**LMP**).

Due to the NP-hard nature of the MP formulation, heuristic approaches are proposed, such as Kernighan-Lin with joins (KLj) and Greedy Agglomerative Edge Contraction (GAEC) [89]. The KLj approach is more accurate and provides the results in a reasonable amount of time, but GAEC is faster, while the results are less accurate. We propose two variants of a heuristic solver (primal feasible heuristic) in the published work in [82]. The variants are among the greedy approaches and generate solutions within a bounded amount of time which experimentally show faster convergence speed than the KLj [89] and provide better segmentation quality than the GAEC [89]. Although branch-and-bound algorithms, Andres et al. [3], as well as Linear Programming (LP) relaxations, Kim et al. [92], Kappes et al. [79], are feasible when applied to small problems, they do not scale easily as it is shown

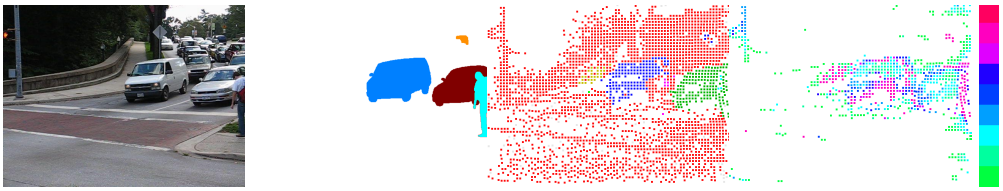


Image Ground-Truth Multicut Uncertainty

Fig. 1.3 Motion segmentation and the proposed uncertainty measure on a street scene are shown. The uncertainty is high on incorrectly segmented points, specifically the missed person.

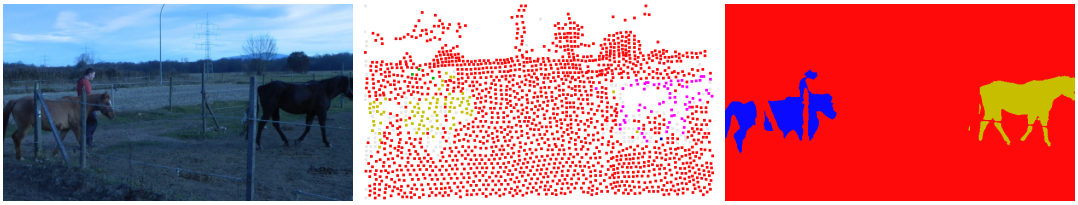


Fig. 1.4 Exemplary multi-label motion segmentation results showing (**left**) the image and its sparse (**middle**) and dense (**right**) segmentation are shown. The sparse segmentation is produced by Keuper et al. [88], and the dense segmentation is the result of our proposed self-supervised learning based model [81] which is based on U-Net [135] architecture.

in Levinkov et al. [102]. Instead, we propose a local search algorithm based on an efficient move-making algorithm [89].

The probabilistic formulation of the MP allows for the uncertainty estimation on the generated results, see Figure 1.3. Such information can be used in many tasks such as video object, motion, and image segmentation. Further, this work allows us to provide the segmentation uncertainty on the node level and can be incorporated into any heuristic solvers for the MP, (see our work in [83]).

Motion segmentation problem is addressed via the Recurrent Neural Networks (RNN) in our work [81], for assigning the affinity cost for the pair of motion trajectories (spatio-temporal curves) (see Section 2.4). Sparse motion segmentation, Ochs et al. [123], is considered, where point trajectories are not sampled at every point but at a defined density. Thereafter, the sparse results are densified by the *self-supervised* approach on a U-Net [135] based network, see Figure 1.4.

The usage of the *pair* of motion trajectories or comparing two trajectories at a time is not sufficient for segmenting higher-order motion patterns, such as out-of-plane rotation and scaling effects (zoom-in and -out) or moving the object towards the camera. In our work [101], such difficulties are addressed via the comparison of more than two trajectories

at a time leading to better results. Moreover, geometric model-fitting has been addressed by Evgeny Levinkov in this publication.

In the work of Ho et al. [60], which is led by Kalun Ho, the Multiple Object Tracking (MOT) problem is addressed by utilizing the spatio-temporal features through consecutive frames, and the LMP in a *self-supervised* manner. Further, an Auto-Encoder (AE) model generates a latent representation of the features. The method is used for pedestrian tracking tasks. We studied the usage of the MP framework on grouping problems such as image and motion segmentation. The objective function of the MP includes several constraints which can be injected into the loss function of the end-to-end trainable model. This model is trained sub-optimally in the work of Song et al. [143]. In our work which is led by Steffen Jung [74], the application of the edge detection to be used by image segmentation is addressed by utilizing the end-to-end trainable model incorporating the edge penalties from the multicut formulation into the objective function and via an adaptive Conditional Random Field (CRF) to get tighter constraints and more valid solutions.

1.2 Problem Statement

Video instance segmentation is a demanding task in many applications such as action recognition, robot navigation, and scene understanding. In this task, the objects interact with each other and move through consecutive frames. One approach is to get a temporal consistency on the objects and track them through the frames utilizing frame-by-frame optical flow information. Providing a reliable segmentation of the static and moving objects through space and time helps to produce the 3D reconstruction of all the objects. We propose a flexible and probabilistic graph-based framework for image and motion segmentation. More specifically, to address the video and motion segmentation, we use a problem formulation based on a generalized MP, Chopra and Rao [36], Deza et al. [40]. With this problem formulation, we get the advantage of solving the model selection problem, where the nature of the formulation determines the number of objects. The "optimal multicut" provides an optimal decomposition of the graph into an optimal number of clusters.

Observable motion in videos can give rise to the definition of objects moving with respect to the scene. The task of segmenting such moving objects is referred to as motion segmentation. There are two main approaches to address this problem, either by aggregating motion information in long and sparse point trajectories or by producing per frame dense segmentations relying on a model trained on a massive amount of training data. We propose a method [81] for self-supervised motion segmentation without depending on a large amount of training data and operating on single frames at the test time. The model for dense motion

segmentation is trained by the *noisy* self-produced labels which generally transfers well for the unlabeled pixels [81]. The MP framework is used for producing the noisy labels, which provide the sparse motion segmentation using the sparse point trajectories. Handling of sparse labels could be avoided if dense unsupervised motion segmentations were given. Although in principle, dense trajectories can be generated by the motion segmentation algorithm, the clustering algorithm does not scale linearly with the number of trajectories, and the computational cost explodes.

Comparing the pair of trajectories is used to cluster the planar motion patterns such as translation and in-plane rotation. The higher-order motion sub-spaces are used to compare more than two trajectories simultaneously to handle ambiguous situations like the out-of-plane rotation and scaling motions, for instance when the object moves toward the camera.

The MP is used on the pedestrian tracking task as well in Tang et al. [147], Ho et al. [60], Tang et al. [146], Keuper et al. [90]. In the work of Ho et al. [60] which is based on the prior work by Tang et al. [147], the bounding box position of the pedestrians is used to produce the tracklets. The appearance of the objects within the bounding boxes is the cue to provide affinity cost for the edges on the multicut graph. After that, the Klj [89] solver provides a grouping of the bounding boxes, where ideally, the bounding boxes in one cluster correspond to a specific pedestrian.

Accessing the confidence of the generated results on any problem such as image and motion segmentation leads to the improvement of the solution. The confidence level can be used for a higher-level interpretation of the results. Thanks to the structure and probabilistic formulation of the MP, we provide a measure for the decisions made for node labels during the optimization process [83]. Accessing such uncertainties is crucial in many applications such as motion and image segmentation.

1.3 Challenges

One of the difficulties dealing with video instance segmentation is recognizing the object saliency where the object is moving or attracting attention through consecutive frames. The reliable and well-segmented results are used on higher-level tasks such as the 3D reconstruction of the objects and proper action recognition. Further, on self-trained robots and autonomous cars, understanding of the scene relies on the ability of the detection and tracking of objects and motion patterns. Providing such information enhances the efficiency of such systems.

Another difficulty is providing clean and sufficient data for deep-learning-based approaches. Such approaches require a large amount of data, and producing such data is

troublesome. Indeed, providing clean data is very important because the training-based model will provide more reliable results than being trained on noisy data. Despite the availability of some datasets such as DAVIS_{2016} [128] (for a single object) and DAVIS_{2017} [131] (for single- and multi-object), where the pixel-level annotation for all the frames on the sequences are provided, there is a big lack on temporally consistent video annotation. Plenty of training-based models rely on proper training data; providing such data is time-consuming. However, with access to such data, it is not guaranteed whether the model is robust in unseen scenarios and can generalize to the data with different distributions than those used in the training stage. Currently, gaining acceptable results without relying on the huge amount of data is getting more attention, focusing primarily on unsupervised [43] and self-supervised [14] approaches.

One example of motion segmentation is a situation where both objects are moved together, like a person who rides a bike; both objects are assigned to the same motion cluster, see Figure 1.1, because they show the same motion pattern unless the person moves away from the bike. Therefore, the result of the motion segmentation depends on the number of visible actions within the video. One approach for motion segmentation is the usage of motion trajectories (see Section 2.4.1). They consist of the points belonging to the consecutive frames tracking a pixel position. Given any point on the motion trajectory, it is possible to travel backward and forward through the seen frames. The moment the object starts to move triggers a signal to separate the point belonging to the object from its surrounding on the current and prior frames. On the other hand, frameworks relying solely on the object appearance do not have access to the prior frames unless they perform based on the memory units such as in Tokmakov et al. [150].

Consider a case where the person is waving while the other part of the body is stable; in this scenario, a person's hand is considered as a separate motion pattern from the other body parts. In this case, the other body parts stick to the motion pattern of the background. In scenarios like this, using the object appearance features helps to join the hand with its corresponding body.

One challenge is determining the granularity level of the segmentation. Depending on the application, one might be interested in the segmentation of the body parts for body gesture detection or the segmentation of people on the scene for pedestrian tracking. In other words, a fundamental challenge is determining the level of segmentation, such as determining the actual object or object-part boundaries. The learning-based methods provide a proper solution to this problem, although this needs a considerable amount of clean image and video annotations.

Due to dealing with video, or accessing multiple frames, processing a large amount of data requires many processing resources, and handling such data is cumbersome. The pre-computed frame-wise super-pixels is used in Vazquez-Reina et al. [154] to solve the computational cost. Although the super-pixel computation is error-prone, propagation of the errors on different levels of the segmentation granularity, hinders the better quality of the results in the video segmentation.

In the MP framework, the entities are connected with their direct neighbors. In some scenarios where there is an ambiguity along the object boundary to make a better cut decision, the *lifted* multicut is used to connect the entities further with their non-neighboring ones without modifying the set of feasible solutions, Keuper et al. [89]. Due to the NP-hard nature of the problem, some heuristic solvers are proposed. Providing uncertainty measures on the decisions made by the solvers helps in making better decisions on the tasks.

1.4 Contributions

In this dissertation, we provide methodological contributions to several aspects of video and motion segmentation, which resulted in seven peer-reviewed publications. This section provides a brief overview of the main contributions.

1.4.1 Evaluation of Low-Level Video Cues

A lightweight variational framework for online tracking of object segmentation in videos based on optical flow and image boundaries is provided. While high-end computer vision methods for this task rely on sequence-specific training of dedicated CNN architectures, we show the potential of a variational model based on generic video information from motion and color. Such cues are usually required for tasks such as robot navigation or grasp estimation. We leverage them directly for video object segmentation and thus provide accurate segmentations at potentially very low extra cost. Our simple method can produce competitive results compared to the costly CNN-based methods with parameter tuning. We evaluate our method on the datasets DAVIS_{2016}, DAVIS_{2017}, and SegTrack v2. Furthermore, our approach can be combined with state-of-the-art CNN-based segmentations to improve their respective results. This work is published in *International Conference on Pattern Recognition (ICPR)*, 2021 [84].

1.4.2 Self-Supervised Densification of Sparse Motion Segmentations

Observable motion in videos can give rise to the definition of objects moving with respect to the scene. The task of segmenting such moving objects is referred to as motion segmentation and is usually tackled either by aggregating motion information in long, sparse point trajectories or by directly producing dense segmentations per frame relying on large amounts of training data. We propose a self-supervised method to learn the densification of sparse motion segmentation from single video frames. While previous approaches towards motion segmentation build upon pre-training on large surrogate datasets and use dense motion information as an essential cue for pixel-wise segmentation, our model does not require pre-training and operates at test time on single frames. It can be trained in a sequence-specific way to produce high-quality dense segmentations from sparse and noisy input. We evaluate our method on the well-known motion segmentation datasets FBMS59 and DAVIS_{2016}. This work is published in *Asian Conference on Computer Vision (ACCV)*, 2020 [81].

1.4.3 Higher Order Minimum Cost Multicuts

The minimum cost lifted multicut problem is a generalization of the multicut problem and is a means to optimize a decomposition of a graph w.r.t. both positive and negative edge costs. It has been shown to be useful in a large variety of applications in computer vision thanks to the fact that multicut-based formulations do not require the number of components given a priori; instead, it is deduced from the solution. However, the standard multicut cost function is limited to pairwise relationships between nodes, while several important applications either require or can benefit from a higher-order cost function, i.e., hyper-edges. This paper proposes a pseudo-boolean formulation for a higher-order motion segmentation [101]. It is based on a formulation of any-order minimum cost lifted multicuts, which allows partitioning of an undirected graph with pairwise connectivity such as to minimize costs defined over any set of hyper-edges. As the proposed formulation is NP-hard and the branch-and-bound algorithm, Andres et al. [3], (as well as obtaining lower bounds) is too slow in practice, we propose an efficient local search algorithm for inference into resulting problems. We demonstrate the versatility and effectiveness of our approach in several applications: 1) We define a geometric multiple model fitting, more specifically, a line fitting problem on all triplets of points and group points that belong to the same line together. 2) We formulate homography and motion estimation as a geometric model fitting problem where the task is to find groups of points that can be explained by the same geometrical transformation. 3) In motion segmentation, our model allows going from modeling translational motion to

Euclidean or affine transformations, which improves the segmentation quality in terms of F-measure.

This work is partly based on a prior conference publication by Keuper [87] and was done with the equal contribution of Evgeny Levinkov, who contributed mainly to the geometric model fitting part, while I contributed primarily to the motion segmentation task. This work is published in *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2022 [101].

1.4.4 Efficient Solvers for Minimum Cost Multicuts

Despite its complexity, the minimum cost lifted multicut problem has found many applications in recent years, such as image and mesh decomposition or multiple object tracking. Its solutions decompose a graph into an optimal number of segments optimized w.r.t. a cost function defined on a superset of the edge set. While the currently available solvers for this problem provide high-quality solutions in terms of the task to be solved, they can have long computation times for more difficult problem instances. Here, we propose two variants of a heuristic solver (primal feasible heuristic), which greedily generate solutions within a bounded amount of time. Evaluations of image and mesh segmentation benchmarks show the high quality of these solutions. This work is published in *ACCV*, 2018 [82].

1.4.5 Uncertainty Prediction in Minimum Cost Multicuts

The minimum cost lifted multicut approach has proven practically good performance in a wide range of applications such as image decomposition, mesh segmentation, multiple object tracking, and motion segmentation. It addresses such problems in a graph-based model, where real-valued costs are assigned to the edges between entities such that the minimum cut decomposes the graph into an optimal number of segments. Driven by a probabilistic formulation of minimum cost multicuts, we provide a measure for the uncertainties of the decisions made during the optimization. We argue that access to such uncertainties is crucial for many practical applications and conduct an evaluation by means of sparsifications on three different, widely used datasets in the context of image decomposition (BSDS500) and motion segmentation (DAVIS_{2016} and FBMS59) in terms of Variation of Information (VI) and Rand index (RI). This work is published in *Uncertainty in Artificial Intelligence (UAI) conference*, 2021 [83].

1.4.6 Contributions as a Co-author

In addition to these core contributions, I contributed as a co-author to two further publications in related domains, which are briefly summarized below.

Application to Unsupervised Person Tracking

Multiple Object Tracking (MOT) is a long-standing task in computer vision. Current approaches based on the tracking by detection paradigm either require some sort of domain knowledge or supervision to associate data correctly into tracks. This work presents a self-supervised multiple object tracking approach based on visual features, and minimum cost lifted multicuts. Our method is based on straight-forward spatio-temporal cues extracted from neighboring frames in image sequences without supervision. Clustering based on these cues enables us to learn the required appearance invariances for the tracking task and train an AutoEncoder to generate suitable latent representations. Thus, the resulting latent representations can serve as robust appearance cues for tracking even over large temporal distances where no reliable spatio-temporal features can be extracted. Despite being trained without using the provided annotations, we show that our model generates competitive results on the challenging MOT Benchmark for pedestrian tracking. This work has been conducted under the project lead of Kalun Ho and published in *ACCV*, 2020 [60].

Learning to Optimize

While the formulation of a Multicut Problem (MP) from independently estimated costs per edge is highly flexible and intuitive, solving the MP is NP-hard and time-expensive. As a remedy, recent work proposed to predict edge probabilities with awareness to potential conflicts by incorporating cycle constraints in the prediction process. We argue that such formulation while providing the first step towards end-to-end learnable edge weights, is suboptimal since it is built upon a loose relaxation of the MP. Therefore, we propose an adaptive CRF that allows progressively considering more violated constraints and, consequently, issuing solutions with higher validity. Experiments on the BSDS500 benchmark for natural image segmentation and electron microscopic recordings show that our approach yields more precise edge detection and image segmentation. This work is based on the master thesis of Sebastian Ziegler under my co-supervision and has been consolidated further by Steffen Jung. It is published in the *German Conference on Pattern Recognition (GCPR)*, 2022. [74].

1.5 Own Publications

Peer-Reviewed Conference Articles

1 Amirhossein Kardoost, Sabine Müller, Joachim Weickert, Margret Keuper, "Object Segmentation Tracking from Generic Video Cues". In: International Conference on Pattern Recognition (ICPR), 2021. [84]

2 Amirhossein Kardoost, Kalun Ho, Peter Ochs, and Margret Keuper, "Self-supervised Sparse to Dense Motion Segmentation". In: Proceedings of the Asian Conference on Computer Vision (ACCV), 2020. [81]

3 Amirhossein Kardoost and Margret Keuper. "Solving Minimum Cost Lifted Multicut Problems by Node Agglomeration". In: Proceedings of the Asian Conference on Computer Vision (ACCV), 2018. [82]

4 Amirhossein Kardoost and Margret Keuper. "Uncertainty in Minimum Cost Multicuts for Image and Motion Segmentation". In: Uncertainty in Artificial Intelligence (UAI), 2021. [83]

5 Kalun Ho, Amirhossein Kardoost, Franz-Josef Pfreundt, Janis Keuper, Margret Keuper, "A Two-Stage Minimum Cost Multicut Approach to Self-Supervised Multiple Person Tracking". In: Proceedings of the Asian Conference on Computer Vision (ACCV), 2020. [60]

6 Steffen Jung, Sebastian Ziegler, Amirhossein Kardoost, Margret Keuper. "Optimizing Edge Detection for Image Segmentation with Multicut Penalties". In: German Conference on Pattern Recognition, 2022. [74]

Peer-Reviewed Journal Article

7 Evgeny Levinkov*, Amirhossein Kardoost*, Bjoern Andres, Margret Keuper (* equal contribution), "Higher-Order Multicuts for Geometric Model Fitting and Motion Segmentation", IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2022. [101]

Chapter 2

Preliminaries

This chapter explains the prior knowledge required for following the subsequent chapters. We start by explaining the datasets and their evaluation metrics used for our methods. Afterward, we describe the low-level cues for video processing and, more explicitly, the consecutive frames. One approach to tracking the objects or pixels on the frames is utilizing the *optical flow* information. Subsequently, the well-known methods for motion and video instance segmentation are explained. On motion segmentation, the spatio-temporal curves, also known as point trajectories, and their use cases are illustrated.

The main optimization framework used for segment generation in this dissertation is the minimum cost multicut (**MP**) and the generalization of it, minimum cost *lifted* multicut (**LMP**). The objective function, the set of constraints, and the well-known heuristic solvers, are explained. Further, the applications of this framework and the basic knowledge of using it for motion segmentation are illustrated.

2.1 Datasets

In this dissertation, the proposed methods are evaluated on different datasets. In the section below, the characteristics and evaluation metrics of these datasets are explained.

2.1.1 Video Segmentation

DAVIS Benchmark. The original version of the Densely Annotated Video Segmentation, DAVIS_{2016} dataset from Perazzi et al. [128], is focused on high-precision binary video object segmentation tracking of rigid and non-rigid moving objects. It contains 30 train and 20 validation sequences. The pixel-wise binary ground-truth segmentation is provided per frame for each sequence. It has different challenges such as light change, object occlusion,

dis-occlusion, and fast motion. Even though this dataset is produced for object segmentation, it is commonly used to evaluate motion segmentation because only one object is moving in each sequence, which makes the motion pattern of the foreground object to be different from the background motion, see Figure 2.1.

The more recent dataset, DAVIS_{2017} from Pont-Tuset et al. [131], also includes the segmentation of multiple objects. It consists of 90 sequences, which are separated into 60 train and 30 validation sequences. The dataset consists of more complicated scenarios where the objects occlude/dis-occlude each other, see Figure 2.1.

The DAVIS datasets are evaluated in terms of boundary accuracy (also known as F-measure) and Jaccard's index, also known as Intersection over Union (IoU).

In the binary segmentation task, given the predictions from the algorithm and the ground-truth segmentation, the precision (P), recall (R), and F-measure (F) are computed as in the Equations (2.1), (2.2), and (2.3), respectively. The Jaccard's index (J) is defined in the Equation (2.4).

In the binary segmentation results where the pixel label is the same as the ground-truth labeling, depending on the label sign (positive or negative), true positive (tp) or true negative (tn) values are computed, counting the number of correctly labeled pixels. Otherwise, when the label of the segmentation result does not agree with the ground-truth labeling, depending on the label of the segmented pixel, false positive (fp) or false negative (fn) values are computed. Utilizing these values, we get the precision (P), recall (R), and F-measure (F) as follows.

$$P = \frac{tp}{tp + fp} \tag{2.1}$$

$$R = \frac{tp}{tp + fn} \tag{2.2}$$

Precision and recall measures are not directly comparable, but they represent cues for under- or over-segmentation, Levinkov et al. [101]. The F-measure (or F1-score) in Equation (2.3) is a harmonic mean of the precision and recall. While F-measure evaluates the segmentation in terms of boundary accuracy, the boundaries of the segmentation result never perfectly match the ground-truth segmentation.

$$F\text{-}measure = 2 \times \frac{P \times R}{P + R} \tag{2.3}$$

Given the segmentation result, A, and the ground-truth segmentation, B, the Jaccard's index, Equation (2.4), accounts for those parts which overlap on the ground-truth labeling.

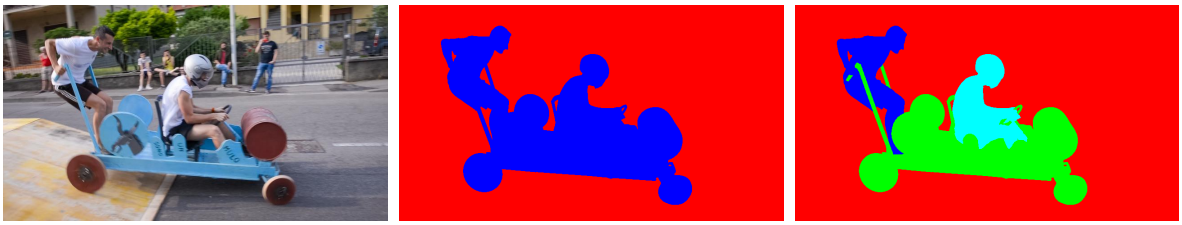


Fig. 2.1 (**left**) An exemplary image from the "soapbox" sequence is shown. (**middle**) The binary object segmentation is provided from the DAVIS_{2016} [128] dataset. Due to the reason that all the objects are moving together, this dataset can be used for the motion segmentation task. (**right**) The multi-object segmentation is shown for the same sequence from DAVIS_{2017} [131].

Fig. 2.2 The images and ground-truth segmentation for two sequences from the SegTrack v2 dataset [104] are shown.

$$J = \frac{|A \cap B|}{|A \cup B|} \tag{2.4}$$

SegTrack v2 Dataset. The dataset from Li et al. [104] consists of 14 sequences with ground-truth annotation per frame and object. The sequences contain different object characteristics, motion blur, occlusions, complex deformations, low resolution, and quality for both binary and multi-object scenarios, see Figure 2.2. The standard evaluation metric is Intersection over Union (IoU) defined in Equation (2.4).

2.1.2 Motion Segmentation

FBMS59 Dataset. The Freiburg-Berkeley Motion Segmentation, FBMS59 dataset from Ochs et al. [123], an extended version of the BMS-26 benchmark from Brox and Malik [25], is specifically designed for the motion segmentation task and consists of 29 train and 30 test sequences. The sequences contain between 19 to 800 frames and show the motion of possibly multiple foreground objects. The sequences cover camera-shaking, rigid/non-

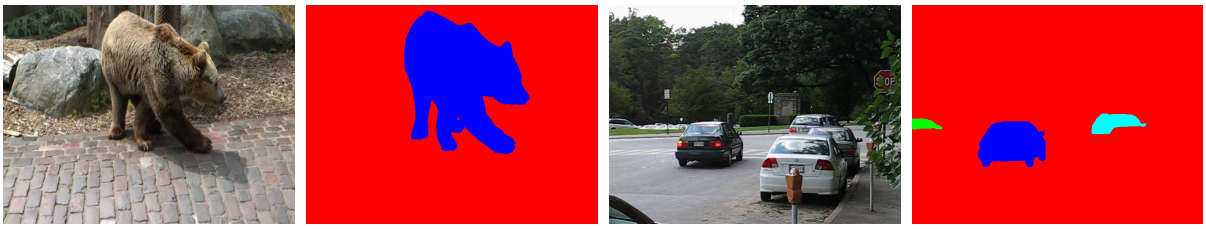


Fig. 2.3 The motion segmentation task from two sequences from the FBMS59 dataset [123] and their ground-truth segmentation are shown. The "bear" (**left** and **middle-left**) shows out-of-plane rotation movement, and the two cars (**middle-right** and **right**) move in the same direction, showing the same motion pattern.

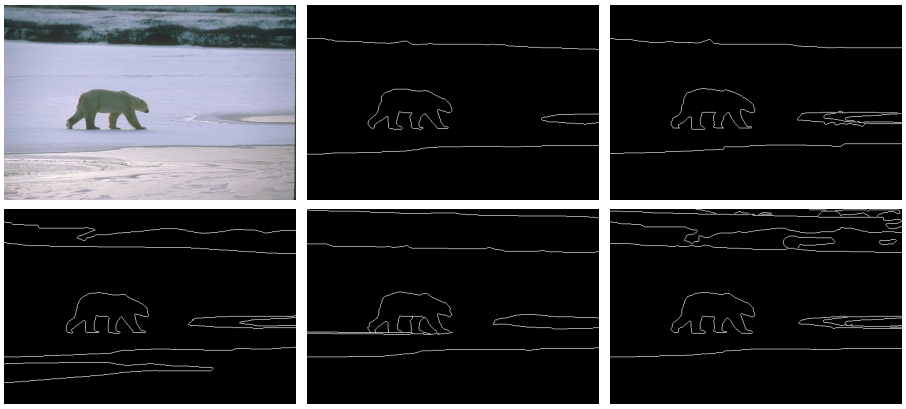


Fig. 2.4 One of the images on the test set of BSDS500 dataset [5] is shown as well as the five different manual boundary segmentations.

rigid motion, as well as occlusion/dis-occlusion of single and multiple objects. The dataset provides manual annotations for all moving objects in the videos for every 20th frame and the *ground-truth definition* files that down-weight annotated segments in some scenes, Levinkov et al. [101], see Figure 2.3. The results on this dataset are evaluated based on the precision (P) Equation (2.1), recall (R) Equation (2.2), and F-measure (F) Equation (2.3).

Hungarian Matching. In order to evaluate the motion segmentation result with the ground-truth segmentation, it is compulsory to match the segments. Therefore, the minimum weight bipartite matching or the Hungarian Matching from Kuhn [98] is used, Ochs et al. [123]. The algorithm finds the matching segments such that the number of misclassified points belonging to the segments is minimized. The point is considered correctly classified if its segment label corresponds to the matched ground-truth segmentation.

VSB100 Dataset. A unified Video Segmentation Benchmark, VSB100 from Galasso et al. [55], is originally proposed as a video segmentation dataset where the task is to mimic human boundary level annotations, i.e. the segments do not necessarily have a notion of objectness, Levinkov et al. [101]. The motion subtask of the dataset is used in Chapter 5 and is evaluated in terms of boundary precision and recall (BPR) and the region metric volume precision and recall (VPR) [101]. VPR assigns a spatio-temporal volume between the ground-truth segmentation and the segmented results via the proposed approach to measure their overlap, Galasso et al. [55]. BPR measures the quality of the segmentation boundary by casting the boundary detection as a boundary from non-boundary pixel classification.

2.1.3 Image Segmentation

The Berkeley Segmentation Dataset (BSDS500) from Arbeláez et al. [5] consists of 200 train, 200 test, and 100 validation images, where for each image, five different human-made annotations are provided, see Figure 2.4.

The images are evaluated by comparing the segmentation result with the different ground-truth segmentations and regarding the region boundaries [5]. There is a precision-recall framework from Martin et al. [112] for the evaluation of the segmentation result, where the results are compared with the human-made annotations. The segmentation results are measured based on the Variation of Information (VI), Meilă [115], and Rand Index (RI) [115], a similarity measure between pairs of segmentations or clusters. Therefore, the region of the segmentation and boundaries are evaluated. VI measures the distance between two clusterings, i.e. the results, and the human-made annotations. It relates to mutual information, and the lower value of VI corresponds to better results. Based on the explanation in Section 2.1.1, the values of tp, tn, fp, and fn are used to compute RI as in Equation (2.5).

$$RI = \frac{tp + tn}{tp + fp + tn + fn} \tag{2.5}$$

2.1.4 Mesh Segmentation

The Princeton Shape Segmentation benchmark from Chen et al. [32] provides 380 meshes consisting of 19 object categories with probabilistic human-made ground-truths [32]. The Rand Index (RI), Meilă [115], and Variation of Information (VI) [115] are two metrics used in this dataset.

2.1.5 Neuronal Structures Segmentation

In the ISBI 2012 challenge, a stack of electron microscopy images of neuronal structures, Cardona et al. [27], Carreras et al. [28], is provided for 3D segmentation. The images are produced by a serial section transmission electron microscope taken from the Drosophila larva ventral nerve cord. The methods are evaluated based on the Rand Index (RI), and Variation of Information (VI).

2.2 Optical flow

The optical flow field $\mathbf{f} : \Omega \to \mathbb{R}^2$ is a function assigning a displacement vector to every point in the image domain Ω. For every point $\mathbf{x} \in \Omega$ in frame t, the optical flow $\mathbf{f}_t(\mathbf{x})$ is the displacement to the most likely location of $\mathbf{x}$ at time $t+1$. Similarly, the backward optical flow $\mathbf{b}_{t+1}(\mathbf{y})$ of any point $\mathbf{y} \in \Omega$ in frame $t+1$ is the displacement to its most likely location in frame t.

To estimate the change in the movement of the pixel positions, one approach is to constrain the match of the attributes of the pixels on the pair of frames. In the video, the 3D world scene is projected into the 2D space (image-domain Ω). Optical flow is a vector field consisting of the horizontal and vertical changes in the pixel positions. Therefore, each vector consists of a magnitude and direction showing the amount of movement of the pixel and its direction between two frames, see Figure 2.5.

In the simple case, considering the static state of the illumination and color of the objects, the horizontal (u) and vertical (v) movement of the pixel at the position (x,y) in the frame I at time t to the next frame at time $t+1$ is computed as follows, Lucas and Kanade [107] (known as Lucas and Kanade approach),

$$I(x,y,t) - I(x+u, y+v, t+1) = 0 \tag{2.6}$$

assuming that the u and v are small and I is smooth [107], the Equation (2.6) is linearised by the first order Taylor expansion as follows [107],

$$I_x u + I_y v + I_t = 0 \tag{2.7}$$

The optical flow estimation provides the amount of the change of the corresponding pixels on the consecutive frames, I_x and I_y correspond to the partial derivative of image

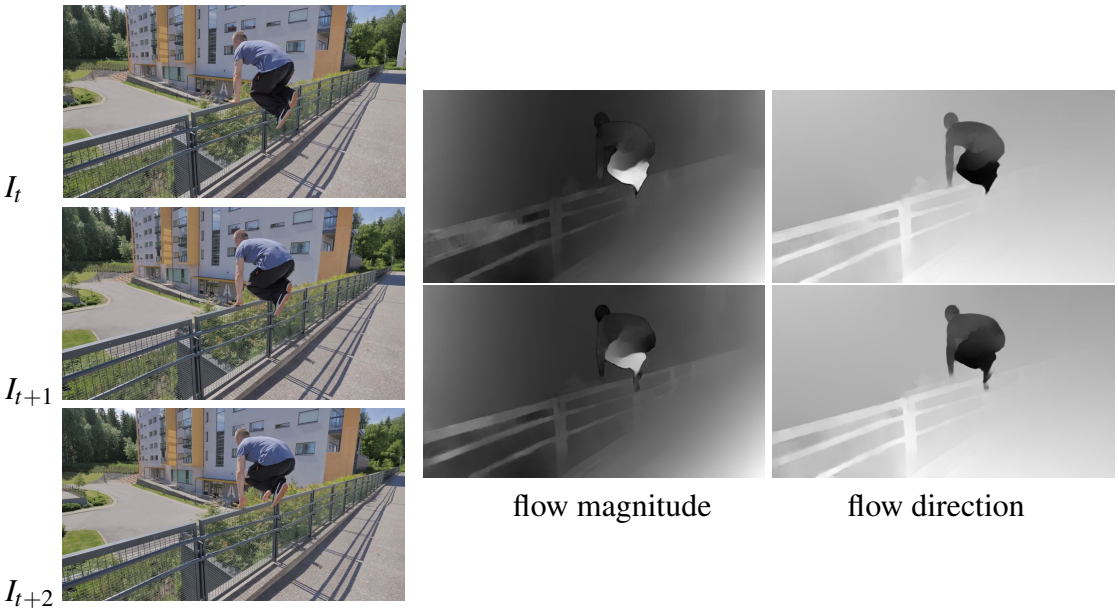


Fig. 2.5 Visualization of direction and magnitude of the optical flow fields $\mathbf{b_{t+1}}$ and $\mathbf{b_{t+2}}$ from example frames of the "parkour" sequence of DAVIS_{2016} [128] are shown. Consecutive flow information is highly correlated.

with respect to x and y direction, and I_t represents the partial derivative of the image with respect to the time dimension. However, such an algorithm is mathematically sound, but in reality, the colors of the objects encounter a drastic change and the objects get occluded or dis-occluded, which makes the problem much more challenging to handle.

Due to the progress in the deep learning-based approaches, the optical flow estimation is addressed via several methods such as FlowNet2 proposed by Ilg et al. [65]. FlowNet2 improves on the flow estimation results of the FlowNet from Dosovitskiy et al. [42] and a network for the small displacement estimation of the objects is added. The stack of the encoder-decoder networks is utilized and trained by the FlyingChairs, Dosovitskiy et al. [42], and the FlyingThings3D, Mayer et al. [113], datasets.

Off-the-shelf deep learning based approaches to low-level tasks such as boundary prediction in Maninis et al. [109], Xie and Tu [161] and optical flow estimation in Ilg et al. [65, 66] produce highly accurate image and motion boundaries, see Figure 2.6. At the same time, such low level information is a basic component in state-of-the-art approaches to robot navigation as in Kendoul et al. [85], Zingg et al. [170], McGuire et al. [114], grasp estimation as in Hasegawa et al. [58], and visual SLAM from Lim et al. [106]. Thus, their computation comes at little to no extra cost in many practical settings.

Label propagation by optical flow information has been previously used for example in Nagaraja et al. [117], Tsai et al. [153], Price et al. [132], Paul et al. [126]. Optical

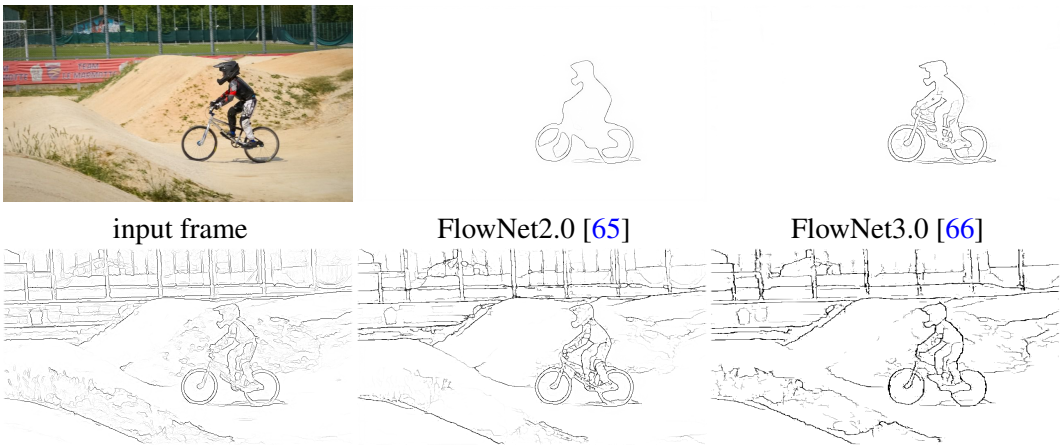


Fig. 2.6 We depict different boundary estimations in the "bmx-bumps" sequence of DAVIS_{2016} [128]. Such low-level cues help in guiding higher-level tasks such as video segmentation.

flow magnitudes are employed as additional input for the network, e.g., in Khoreva et al. [91], Voigtlaender and Leibe [155], to provide additional motion and saliency cues. However, the exact localization quality of the optical flow is hardly used. Notably, He et al. [59] use optical flow to create patch correspondences in a video. This provides diverse training data and helps to improve the training of deep neural networks.

Several methods propagate the labels through consecutive frames based on information from the optical flow as in Drayer and Brox [44], Tsai et al. [153], Khoreva et al. [91]. Badrinarayanan et al. [7] and Jain and Grauman [70] use in addition to the flow information a Gaussian Mixture Model (GMM) and a Markov Random Field (MRF), respectively.

Access to better optical flow information enhances the quality of the relying tasks on such cues. In the field of motion and video instance segmentation, as we deal with the consecutive frames, the optical flow information is helpful to enhance the tracking and segmentation results as we show in Chapter 3.

2.3 Image and Video Segmentation

Image Segmentation is the process of partitioning the image domain into multiple segments. One of the approaches to image segmentation is the usage of the U-Net architecture proposed by Ronneberger et al. [135]. A U-Net is an encoder-decoder network with skip connections. During encoding, characteristic appearance properties of the input are extracted and are learned to be associated with objectness. In the decoding phase, the extracted

properties are traced back to locations causing the observed effect, while details from the downsampling phase are taken into account to ease the localization. In Ronneberger et al. [135], end-to-end training of the model on few images perform acceptable results on the ISBI dataset [27, 28] (see Section 2.1.5). We used the same network to provide the dense motion segmentation based on the sparsely segmented images in Chapter 4.

Despite the usage of the deep learning-based approaches, various variational formulations have been proposed for multi-label segmentation in still images, e.g. in Nieuwenhuis and Cremers [119], Bae et al. [8]. In Nieuwenhuis and Cremers [119], image segmentation from user scribbles is addressed in a variational framework considering the spatial and color information of the pixels. In Müller et al. [116] such methods have been applied to produce dense video segmentations from sparse seeds in a frame-by-frame manner, based on automatically generated seeds from point trajectories, Keuper et al. [88], Keuper [87] (see Section 2.4.1).

Video Segmentation is the process of partitioning the video sequence into multiple objects available in at least one frame. One of the approaches for video segmentation is based on the label propagation by optical flow, which has been previously used, for example, in Tsai et al. [153], Price et al. [132], Nagaraja et al. [117], Paul et al. [126]. Unlike Price et al. [132], which exclusively utilize temporal coherence, Nagaraja et al. [117] only uses color consistency. The problem of label propagation in videos has also been addressed by deep learning-based approaches such as in Perazzi et al. [127], Voigtlaender and Leibe [155], Bao et al. [13], Maninis et al. [110], Siam et al. [141]. Such networks are trained on specific datasets and the first frame annotation of a sequence to produce a segmentation of subsequent frames.

In Maninis et al. [110], a spatio-temporal Markov Random Field (MRF) model is defined over pixels to produce temporally consistent video object segmentation. In their approach, spatial dependencies among pixels are encoded by a Convolutional Neural Network (CNN) trained for the specific target sequence. In contrast, the OSVOS-S approach from Maninis et al. [110] can be considered entirely complementary. They propose a one-shot video object segmentation framework that explicitly does not rely on any temporal consistency within the data, such that object occlusions and dis-occlusions can be handled particularly well. In contrast, OSVOS-S [110] successively transfers generic, pre-trained, semantic information to the task of video object segmentation by learning the appearance of the annotated (single) object of the test sequence.

2.4 Motion Segmentation

Motion segmentation is the task of segmenting motion patterns through consecutive frames of a video sequence. This is in contrast to semantic segmentation, where one seeks to assign pixel-wise class labels in an image. Further, it is different from video instance segmentation where one is interested in tracking the object instances. Thus, for motion segmentation, we need motion information and at least two frames to be visible to distinguish between motion segments. According to the Gestalt principle of common fate from Koffka [95], motion patterns of objects are often more homogeneous than their appearance and provide robust cues for moving object segmentation.

Observable motion in videos gives rise to the definition of objects moving with respect to the scene. The task of segmenting such moving objects is referred to as motion segmentation and is usually tackled either by aggregating motion information in long, sparse point trajectories, (see Section 2.4.1), or by directly producing per frame dense segmentations relying on large amounts of training data.

Common end-to-end trained CNN based approaches to motion segmentation are based on single frame segmentations from optical flow such as in Tokmakov et al. [149, 150], Maczyta et al. [108], Fragkiadaki et al. [49], Siam et al. [142]. Tokmakov et al. [150] make use of large amounts of synthetic training data, Dosovitskiy et al. [42], to learn the concept of object motion. Further, Tokmakov et al. [149] combine these cues with an ImageNet, Deng et al. [39], pre-trained appearance stream and achieve long-term temporal consistency by using a Gated Recurrent Unit (GRU) optimized on top. A binary video segmentation model is trained and used to distinguish between static and moving elements. Siam et al. [142] use a single convolutional network to jointly model motion and appearance cues for autonomous driving. A frame-wise classification problem is formulated in Fragkiadaki et al. [49] to detect motion saliency in videos. In Maczyta et al. [108] for each frame, multiple figure-ground segmentations are produced based on motion boundaries. A moving objectness detector trained on image and motion fields is used to rank the segment candidates. Variational formulations based on optical flow are used in Cremers [37], Lao and Sundaramoorthi [99], where Lao and Sundaramoorthi [99] employ a strong geometric model.

Addressing motion segmentation is different from video object segmentation, as in motion segmentation, different motion patterns are segmented, which makes connected objects seem like one object if they move together with the same motion pattern. For instance, we refer to the case of a person riding a horse, see Figure 2.7. In contrast, in video object segmentation we deal with two separate objects like in Hu et al. [63], Tsai et al. [153], Papazoglou and Ferrari [125].

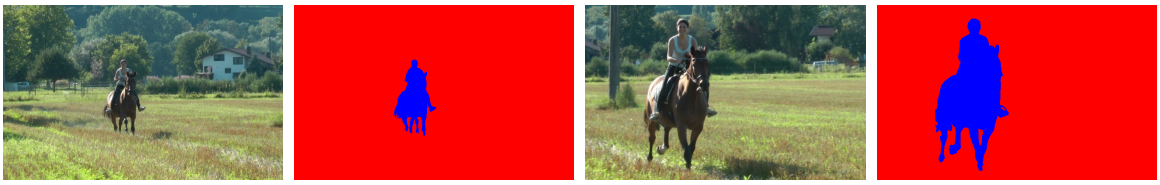


Fig. 2.7 Motion segmentation example is provided for frames 1 and 160 of the "horses01" sequence (with their ground-truth motion segmentation) in the FBMS59 [123] dataset. Due to the reason that the person and the horse move together and have the same motion pattern, the same motion label (blue color) is assigned.

In the deep learning-based approach of Tokmakov et al. [150] for motion segmentation, two-stream networks are used for encoding the position and appearance information of the objects. One of the streams is the appearance network, where a pre-trained deep convolutional deeplab, Chen et al. [30], based network on the PASCAL VOC 2012, Everingham et al. [46], is used. Another stream estimates the motion of the objects using the MP-Net from Tokmakov et al. [149], and the CNN is pre-trained for the motion segmentation task. This appearance and motion information is combined with the memory module, a GRU, Cho et al. [34], based model, trained on the training split of the DAVIS_{2016} dataset [128]. The memory module memorizes the position and appearance of the objects to provide reliable results when the motion field estimation fails. Similarly, Jain et al. [69] employ a realistic dataset extracted from pairs of frames from the ImageNet video dataset from Russakovsky et al. [136] to learn object motion and appearance cues. While these approaches directly yield pixel-accurate segmentations, they replace explicit motion model assumptions with vast training data and can not inherently determine the number of moving objects.

In DyStaB from Yang et al. [165] unsupervised moving object segmentation is addressed by partitioning the motion field with respect to a mutual-information-based objective and learning object models from the segments. Yang et al. [162] propose a self-supervised transformer model to segment optical flow fields into primary objects and background in a generative way. The combination of appearance-based detectors and geometric motion segmentation is used to segment *rigid* motions in Yang and Ramanan [163].

Methods like Bideau and Learned-Miller [21], Cremers [37] approach the problem in a probabilistic way. In Bideau and Learned-Miller [21], motion segmentation is approached in a probabilistic way, and the camera motion is subtracted from each frame for improved training. In Bideau et al. [22], the idea of prior camera motion subtraction is used to allow for better CNN training for frame-wise segmentation. Bideau et al. [23] propose a multi-step procedure in which first the camera motion, fit using random sampling consensus (RANSAC), Fischler

and Bolles [48], in the first frames, is subtracted, then a set of rigid motion models is fitted, and last object segmentation proposals from CNNs are used to combine the rigid motion parts into objects. In Irani and Anandan [67], a unified approach is proposed to handle the moving object detection problem separately in the 2D and 3D scenes based on geometric interpretations and the parallax motion analysis, Levinkov et al. [101].

2.4.1 Trajectory based Motion Segmentation

A different line of work for motion segmentation relies on point trajectories (point trajectories are defined in this section). In Ochs et al. [123] motion trajectories are created based on optical flow information, and a graph representing the trajectories and their affinities is created. This graph represents the sparsely tracked points over the consecutive frames. The produced graph is clustered based on several algorithms, such as spectral clustering, J. Shi and Malik [68]. In this approach, the number of clusters m is required and needs to be determined. Correspondingly, the m eigenvectors belonging to the m largest eigenvalues are selected, each representing a specific motion pattern. The solution to spectral clustering provides a sparse motion segmentation. To provide a dense result, the sparsely labeled points are used with the variational framework to produce the dense results, i.e. the label and the position of the points are used to leak the labels to the unlabeled points in an iterative manner [123].

Here, long-term motion information is first used to establish sparse but coherent motion segments over time. Dense segmentations are usually generated in a post-processing step such as Ochs et al. [123] or our proposed motion segmentation approach in [81]. However, in contrast to end-to-end CNN-based motion segmentation approaches, trajectory-based methods have the desirable property of directly handling multiple motion patterns.

The Euclidean difference between two local motion descriptors such as optical flow vectors or point trajectories measures how well the behavior of the two entities can be described by a single translational motion model. A simple model with only pairwise potentials can yield good performance in practice, Ochs et al. [123], Keuper et al. [88]. While being successful in providing segmentations in simple scenarios, more complex motion patterns can not be resolved with only pairwise potentials. For example, scaling effects (such as zooming in/out of the camera or movement of objects toward the camera), out-of-plane rotation, and highly non-rigid motion of object parts hinder providing high-quality motion segmentations. Therefore, higher-order motion models that can simultaneously compare more than two motion vectors are required.

Two motion vectors can be used to estimate the transformations describing translation, rotation, and scaling. Therefore, given any three points, via residual errors, it is possible to

estimate the motion of the points by one Euclidean transformation. The motion differences can be described by the costs of at least order three. Four motion vectors can be used to estimate affine motion differences and at least five motion vectors are required to assign the cost to differences in homographies, Keuper [87].

Point Trajectory. Point trajectories are spatio-temporal curves represented by their frame-wise sub-pixel-accurate (x,y)-coordinates. They can be generated by tracking points using optical flow by the method of Brox et al. [25]. The resulting trajectory set aims for a minimal target density (e.g. one trajectory in every 8 pixels). Trajectories are initialized in a video frame and end when the point cannot be tracked reliably anymore, e.g. due to occlusion. To achieve the desired density, possibly new trajectories are initialized throughout the video. Using trajectories brings the benefit of accessing the object motion in prior frames. The partial derivative of the sub-pixel-accurate (x,y)-coordinates with respect to the time dimension produces the **motion trajectories**.

Point trajectory based motion segmentation algorithms have proven to be robust and fast in Ochs et al. [123], Keuper et al. [88], Keuper [87], Brox and Malik [25], Fragkiadaki et al. [52], Shi et al. [137], Rao et al. [133]. By a long-term analysis of a whole video shot at once by the means of such trajectories, even objects that are static for most of the time and only move for few frames can be identified, i.e. the model would not "forget" that a car has been moving, even after it has been static for a while. The same argument allows articulated motion to be assigned to a common moving object.

Point trajectories build the basis for many motion segmentation methods such as Brox and Malik [25], Fragkiadaki et al. [49, 51], Ochs et al. [123], Keuper et al. [88]. For a video of length N, Brox and Malik [25] yield n point trajectories p_i with the maximum length N, where n depends on the desired sampling rate. Due to occlusions and mistakes in the optical flow estimation, most trajectories are significantly shorter than N, and some trajectories start after the first frame to ensure even point sampling throughout the sequence.

2.5 Minimum Cost Multicuts

The minimum cost multicut problem (**MP**) by Chopra and Rao [36], also known as correlation clustering, is a binary *edge* labeling problem defined on a graph with real-valued edge costs. The edges define an affinity between the entity (node) pairs, which participate in the cut/join decisions of the multicut solver. Specifically, the costs and the constraints of the multicut problem lead to the pairwise decisions on the entity pairs in the graph. The feasible solutions to the MP propose decompositions of the graph, and the optimal solution corresponds to the

Maximum A Posteriori probability (MAP) estimate, Andres et al. [3]. Yet, the problem is shown to be NP-hard, Bansal et al. [12]. While optimal solutions or solutions within bounds of optimality can be found for small instances, Demaine et al. [38], Swoboda and Andres [145], Andres et al. [2, 3], most practical applications depend on heuristic solvers such as Keuper et al. [89], Beier et al. [17, 16]. The branch-and-bound, Andres et al. [3], algorithms, as well as Linear Programming (LP) relaxations, Kim et al. [92], Kappes et al. [79], are feasible when applied to small problems, but they do not easily scale, Levinkov et al. [102].

The formulation is flexible and can easily be adapted to any clustering problem. Specifically, entities are represented by nodes in a graph, and real-valued costs are assigned to edges corresponding to the node affinities. The minimum cost multicut then decomposes the graph into an optimal number of segments.

Despite the requirement for the model selection and providing the number of clusters to the spectral clustering, in the MP approach, the number of clusters is determined implicitly by the nature of the problem. This formulation is attractive because the feasible solutions of the multicut problem relate one-to-one to the decompositions of a graph, and the MP does not favor balanced solutions a priori, J. Shi and Malik [68]. Furthermore, multicut algorithms are easy to use: They take as input a graph and, for every edge, a real-valued cost of the incident vertices being in distinct components and output a 01-labeling of the edges, which induces a decomposition of the graph. Assigned real-valued cost of the incident vertices ideally reflects the logit of the probability of the edge being cut, i.e. $\log \frac{1-p_e}{p_e} + \log \frac{1-p^*}{p^*}$, for a cut probability p_e at edge e, and a prior probability $p^* \in (0,1)$ of cuts.

In the following, the minimum cost multicut problem [36, 40] is defined. Given any graph $G = (V,E)$, a cost function $c : E \to \mathbb{R}$ and *edge* labels $y : E \to \{0,1\}$, the optimization problem states in Equation (2.8) is the instance of the multicut problem with respect to the graph G and costs c.

$$\min_{y \in \{0,1\}^E} \sum_{e \in E} c_e y_e \tag{2.8}$$

$$s.t. \quad \forall C \in cycles(G) \quad \forall e \in C : y_e \leq \sum_{e' \in C \setminus \{e\}} y_{e'}.$$

The inequality constraints stated over all cycles of G ensure that the edge labeling y induces a decomposition of G. It is sufficient to ensure these constraints on the subset of all chordless cycles in Chopra and Rao [36]. Cycle inequalities ensure that there is not only one

cut edge in any chordless cycle. The multicut problem allows assigning a cost or reward for every edge $e \in E$ to be cut.

On the image segmentation task, the nodes can represent the pixels, and the neighboring pixels represent the edges of the graph. Similarly, on the point clouds, after triangulation of the points and acquiring the mesh data, each triangle and its neighbors correspond to the nodes and edges of the graph, respectively. Further, on the task of object tracking or motion segmentation, each motion trajectory (see Section 2.4.1) represents a node on the graph, and the trajectories meeting each other on the same or neighboring frames create an edge on the graph.

2.5.1 Minimum Cost Lifted Multicut Problem

The minimum cost *lifted* multicut problem (**LMP**) has first been proposed by Keuper et al. [89], where its promise for the decomposition of pixel grid graphs as well as 3D shape meshes have been shown. It has been successfully applied in fields like multiple object tracking, Tang et al. [147], and motion segmentation, Keuper [87], and became known to produce state-of-the-art results on the segmentation of electron microscopic stacks of neuronal structures, Beier et al. [15, 18]. However, the multicut cost function, Demaine et al. [38], Bansal et al. [12], can assign a cost or a reward only to direct neighbors in the graph, which can be a severe limitation in specific applications. For example, in the case of image segmentation and a 4-connected graph, the final solution is likely to deteriorate significantly since inter-pixel edge probability estimates tend to be noisy. Introducing additional (*lifted*) edges into the multicut objective allows to capture of information in a non-local neighborhood but preserves the original feasible set of solutions, Keuper et al. [89]. It is a generalization of the MP, Chopra and Rao [36], Deza et al. [40], see Figure 2.8 (b).

The LMP is defined with respect to a graph $G = (V, E)$ and a graph $G' = (V, E')$ with $E \subseteq E'$ and a cost function $c' : E' \to \mathbb{R}$ which allows to assign, for every edge in E' a cost or reward for being cut. Its feasible solutions relate one-to-one to decompositions of the graph G. Rigorously, for any undirected graph $G = (V, E)$, any $F \subseteq \binom{V}{2} \setminus E$ and any $c' : E \cup F \to \mathbb{R}$, the linear program written in Equation (2.9) - (2.12) is an instance of the LMP with respect to G, F and c'.

$$\min_{y \in \{0,1\}^{E'}} \sum_{e \in E'} c'_e y_e \tag{2.9}$$

$$s.t. \quad \forall C \in cycles(G) \quad \forall e \in C : y_e \leq \sum_{e' \in C \setminus \{e\}} y_{e'} \tag{2.10}$$

$$\forall vw \in F \quad \forall P \in vw - paths(G) : y_{vw} \leq \sum_{e \in P} y_e \tag{2.11}$$

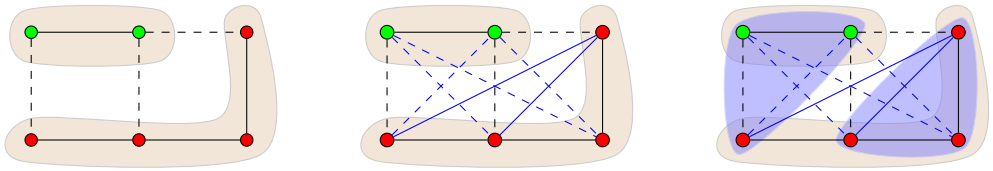
$$\forall vw \in F \quad \forall C \in vw - cuts(G) : 1 - y_{vw} \leq \sum_{e \in C} (1 - y_e) \tag{2.12}$$

The linear inequalities in Equations (2.10) - (2.12) constrain y such that $\{e \in E | y_e == 1\}$ is a multicut of G. They ensure further that, for any edge $uv \in F, y_{uv} = 0$ iff there exists a path (uv-path) in the original graph G along which all edges are connected, i.e., labeled 0.

However, there are two main limitations of the multicut problem: 1) It allows specifying costs or rewards only for direct neighbors in the graph, which limits the expressiveness of the cost functions only to local neighborhoods. An important difference of the lifted edges is that they only define a cost between vertices but not connectivity, thus preserving the original feasible set of solutions. See Horňáková et al. [62] for detailed proof that the LMP is not simply equivalent to a multicut problem with more edges. 2) It allows specifying only pairwise edge costs, which is not enough for some applications. For example, for the motion segmentation task, we need 3rd-order costs, while homography estimation requires 5th-order costs [101]. Kim et al. [92] and Kappes et al. [79] proposed a formulation that allows to specify costs of arbitrary order, see Figure 2.8 (c). They used a cutting-plane algorithm to solve emerging models, which is impractical for large real-world instances due to the linear programming-solver's bottleneck. While this formulation is extremely useful, it turned out to be truly harder than the NP-hard MP, Horňáková et al. [62].

2.5.2 Existing Solvers

A canonical approach to solving instances of the MP is to solve the linear programming relaxation of the multicut polytope, Chopra and Rao [36] as, for example, proposed in Demaine et al. [38]. This procedure, with subsequent thresholding to produce feasible solutions, has found wide applications Kim et al. [92, 93], Kappes et al. [78]. Yet, despite its NP-hardness, the MP has been solved to optimality for some practically relevant instances such as Andres et al. [3], Kappes et al. [77]. Yet, for larger problem instances, heuristic



(a) **(b)** **(c)**

Fig. 2.8 **(a)** An example of a graph decomposition and its encodings. Switching green and red labels will produce a different encoding for the same decomposition. Dashed lines, in turn, constitute a multicut of a graph and uniquely define its decomposition. **(b)** An example of a *lifted* graph decomposition and its encoding. Blue lines denote lifted edges that connect vertices that are not direct neighbors in the graph. **(c)** An example of 3rd-order costs that consider three nodes at a time (light blue triangles) for a better join/cut decision. If a higher-order cost does not correspond to a clique in the graph, we add lifted edges in our work proposed in [101].

solvers such as Cut,Glue & Cut (CGC) by Beier et al. [17], Kernighan–Lin (KL) [86, 77] and its variant Kernighan–Lin with joins (KLj) by Keuper et al. [89], or the Fusion Moves approach by Beier et al. [16] need to be employed. They have been shown to generate reasonable solutions in practice, although no theoretical guarantee can be provided.

In the following, two of the solvers, specifically Greedy Agglomerative Edge Contraction (GAEC) and KLj from Keuper et al. [89] are provided in Algorithms 1 and 2, respectively. In GAEC, it starts from the decomposition of the graph into single nodes. The nodes are joined iteratively, such that in each iteration, the objective function is reduced maximally. The algorithm terminates if there is not any join available which decreases the objective function [89]. In Algorithm 1, $G = (V, E)$ represents the main graph, $G' = (V, E \cup E')$ the *lifted* graph, with the set of costs c assigned on the edges.

In KLj, the *update_bipartition* algorithm, Algorithm 3, is used to look into the transformations of the nodes between the clusters and selects the transformation which decreases the objective function maximally [89]. It does so by performing either of the following cases: 1. moving a connected set of nodes between two neighboring components, 2. moving a connected set of nodes to a new component, or 3. by merging two components. This is illustrated more in (Section 5.4). This can be implemented efficiently based on the original algorithm from Kernighan and Lin [86] by keeping track of the hypothetical costs of moving vertices between neighboring components as given in Algorithm 2. The output of the algorithm is the 01 labeling of the edges.

Algorithm 1: GAEC [89]

$\mathscr{E} := E$, $\mathscr{E}' := E'$, $\mathscr{V} := V$
$\mathscr{G} := G$ and $\mathscr{G}' := G'$
foreach $ab \in \mathscr{E}'$ **do**
    $\chi_{ab} := c_{ab}$
while $\mathscr{E} \neq \emptyset$ **do**
    $ab := \underset{a'b' \in \mathscr{E}}{\operatorname{argmax}} \chi_{a'b'}$
    if $\chi_{ab} < 0$ **then**
        break
    contract ab in $\mathscr{G}$ and $\mathscr{G}'$
    foreach $ab \neq ab' \in \mathscr{E}'$ **do**
        $\chi_{ab'} := \chi_{ab'} + \chi_{bb'}$

Algorithm 2: KLj [89]

$\mathscr{E} := E \cup E'$, $\mathscr{V} := V$
$\mathscr{G} := G'$
while *no_changes* **do**
    foreach $ab \in \mathscr{E}$ **do**
        if *has_changed*(a) *or* *has_changed*(b) **then**
            update_bipartition$(\mathscr{G}, a, b)$
    foreach $a \in \mathscr{V}$ **do**
        if *has_changed*(a) **then**
            while *no_changes* **do**
                update_bipartition$(\mathscr{G}, a, \emptyset)$

2.5.3 Applications

Formulations of the image segmentation problem like Arbeláez et al. [5] as MP have been considered in Alush and Goldberger [1], Andres et al. [2, 3, 4], Bagon and Galun [9], Beier et al. [16, 17], Kappes et al. [78, 79, 80], Kim et al. [92, 93], Nowozin and Jegelka [120], Yarkony et al. [166, 167]. Further, the minimum cost (lifted) multicut problem has been widely applied in computer vision for applications ranging from image segmentation, Keuper et al. [89], to multiple object tracking, Keuper [87], Horňáková et al. [62], Tang et al. [147], Hornáková et al. [61], connectomics, Beier et al. [18], cell tracking [134], instance segmentation, Kirillov et al. [94], motion segmentation, Keuper et al. [88], Keuper [87], and pose estimation, Pishchulin et al. [130].

In this dissertation, the formulation is widely applied in different applications such as motion, mesh data, and image segmentation. The application of such a framework is used, and we also proposed two efficient heuristic solvers and a measure for the uncertainties of the decisions made during the optimization.

2.6 Motion Segmentation Using Minimum Cost Multicuts

Formulations of MPs have been successfully used for motion segmentation, like in Keuper et al. [88], Keuper [87]. This application aims to segment motion patterns of the foreground objects with respect to the scene and irrespective of the camera motion, scaling movements, and out-of-plane rotation of the objects, Ochs et al. [123]. One widely used paradigm to tackle this problem is to define motion trajectory, (see Section 2.4.1), where the trajectories

Algorithm 3: update_bipartition

$\mathscr{G} = (\mathscr{V}, \mathscr{E} \cup \mathscr{E}', c)$ partitions a and b, and edge costs c
$D^{a \cup b}$ = compute_differences($\mathscr{E}, \mathscr{E}', a, b$)
$BDY \leftarrow$ compute_boundary_nodes($\mathscr{E}', a, b$)
$\Delta_{join} \leftarrow$ compute_gain_from_joining($\mathscr{E}', \mathscr{E}, a, b$)
$S^{|a \cup b|} = 0$ // cumulative gain
$M = [.]$ // empty move vector
for $i \leftarrow 1$ **to** $|BDY|$ **do**
    $v^* \leftarrow \text{argmax}_v\ (D_{(a \cup b) \cap BDY})$
    M.push_back(v^*)
    $G = \mathscr{G}$
    foreach $e \in G.edges(v*)$ **do**
        if $same\_partition(G.nodes(e) \backslash v^*)$ **then**
            $D_{v^*} \leftarrow D_{v^*} - 2c_e$
        else
            $D_{v^*} \leftarrow D_{v^*} + 2c_e$
    $S_i = S_{i-1} + D_{v^*}$
    $BDY \leftarrow$ update_boundary($v^*, \mathscr{E}', a, b$)
$k \leftarrow \text{argmax}_i S_i$ // best number of moves
if $\Delta_{join} > S_k$ **and** $\Delta_{join} > 0$ **then**
    join_partitions(y, a, b);
else if $S_k > 0$ **then**
    move_nodes(y, a, b, k)

are cast to the nodes in a graph, and their affinities are used to compute costs on the edges, by calculating the pairwise differences on point trajectories (second-order costs). Such differences are calculated only for trajectories with at least two frames in common, Keuper et al. [88], and based on motion, color, and spatial distance cues.

The motion distance is computed as in Equation (2.13). The motion difference of two trajectories p_i and p_j at time t is computed as follows, Ochs et al. [123].

$$d_t^{\text{motion}}(p_i, p_j) = \frac{\|\partial_t p_i - \partial_t p_j\|}{\sigma_t}. \tag{2.13}$$

Here, $\partial_t p_i$ and $\partial_t p_j$ represent the partial derivatives of p_i and p_j with respect to the time dimension. The σ_t represents the variation of the optical flow defined in Ochs et al. [123]. The motion distance of two trajectories is defined by the maximum motion available over time, Keuper [87],

$$d^{\text{motion}}(p_i, p_j) = \max_t d_t^{\text{motion}}(p_i, p_j). \tag{2.14}$$

Color and spatial distances d^{color} and d^{spatial} are computed as average distances over the joint lifetime of two trajectories, Keuper [87], as proposed in Keuper et al. [88]. The costs are computed by non-linear combination of these three cues.

$$\begin{aligned} c_{ij} \quad = \quad & \max\big(\bar{\theta}_0 \quad + \quad \theta_1 \quad d^{\text{motion}}(p_i, p_j) \\ & \qquad\quad + \quad \theta_2 \quad d^{\text{spatial}}(p_i, p_j) \\ & \qquad\quad + \quad \theta_3 \quad d^{\text{color}}(p_i, p_j)\,, \\ & \theta_0 \quad + \quad \theta_1 \quad d^{\text{motion}}(p_i, p_j) \qquad \big) \end{aligned} \tag{2.15}$$

As proposed in Keuper et al. [88], weights and intercept values θ are assigned[1].

Due to the reason that the clustering algorithm does not scale linearly with the number of trajectories, tracking all the pixels generates a considerable amount of trajectories, which consecutively, makes the clustering algorithm explode computationally. Therefore, one often considers sparse motion segmentation, Ochs et al. [123], where point trajectories are not sampled at every point but a defined density. With this approach, the segmentations do not cover all the image pixels.

While in Keuper et al. [88], Keuper [87] the partitioning of trajectories into motion segments use the multicut problem, other approaches employ sparse subspace clustering, Elhamifar and Vidal [45], or spectral clustering and normalized cuts, Ochs and Brox [122], Shi and Malik [138], Fragkiadaki and Shi [50]. In spectral clustering, a model selection is needed to determine the final number of segments. In contrast, the minimum cost multicut formulation allows for direct optimization of the number of components via repulsive cost.

In this setting, the solution to the motion segmentation problem is sparse, which requires further computation to produce dense results. There are different methods to provide dense motion segmentations from the sparse results, like the variational approach from Ochs et al. [123] and our proposed self-supervised approach [81] (see Section 4). By evaluating the motion segmentation datasets, such as DAVIS_{2016} [128] and FBMS59 [123], (see Section 2.1), and show that there is a gap between the sparse and dense results. This is because on the method of Ochs et al. [123] the variational approach makes the labels of the sparse results leak to unwanted regions, e.g., loosely textured areas of the image. The advantage of using the trajectory-based motion segmentation via multicuts is that it generates

[1] Specifically, $\bar{\theta}_0 = 6$, $\theta_0 = 2$, $\theta_1 = \theta_3 = -0.02$ and $\theta_2 = -4$.

the multi-label results, while in the training-based approach of Tokmakov et al. [149] the outcomes are binary, which makes no distinction between different motion patterns.

Despite the efficiency of the approach of Keuper et al. [88], using the pairwise differences is not able to handle complicated situations like the out-of-plane rotation, scaling motion of the objects (when the object moves towards the camera), and when the objects move simultaneously in the same direction, refer to Figure 2.3. Such cases lead to either over-segmentation of the motion patterns or assigning the same label to similar motion patterns. To address such ambiguities, the higher order motion sub-spaces is used to compare more than two trajectories at a time, Keuper [87], Levinkov et al. [101], (see Section 5), where we show the efficiency of our approach by evaluating on DAVIS_{2016} [128], FBMS59 [123], and VSB100 [55] datasets (see Section 2.1).

Utilizing the multicut approach for motion segmentation, we produce uncertainty measures on the segmentation results in [83], (see Section 7), driven by a probabilistic formulation of minimum cost multicuts. Access to such uncertainties is crucial for motion segmentation to correct the erroneous results. Further, we investigate the potential benefits of the predicted uncertainties for improving on the self-supervised sparse to dense motion segmentation results [81] (see Section 7).

In the next chapter, Chapter 3, we dive into the solutions for the *video instance segmentation*, and we study the effect of low-level cues such as the optical flow and edge maps information on the task. In Chapter 4, we study the *motion segmentation* task via the usage of motion trajectories, where we provide a sparse segmentation of the motion trajectories and densify them in a self-supervised manner. We extend our model for motion segmentation to solve complex cases like out-of-plane motion patterns or scaling effects of the objects via hypergraphs in Chapter 5. After discussing the video and motion segmentation tasks, we investigate the minimum cost (*lifted*) multicut formulation and propose two new heuristic solvers in Chapter 6. By studying the probabilistic behavior of the minimum cost multicut, in Chapter 7, we offer an informative uncertainty measure on the decisions made by minimum cost solvers on the image and motion segmentation tasks. Finally, in Chapter 8, limitations, possible future work, and conclusions are provided.

Chapter 3

Video Instance Segmentation - Evaluation of Low-Level Video Cues

In this chapter, we study the effect of the low-level cues such as optical flow information and the boundary estimates on tracking the instance segmentations through consecutive frames. More specifically, we propose a lightweight variational framework for online tracking of object segmentations in videos based on optical flow and image boundaries. While high-end computer vision methods for this task rely on sequence-specific training of dedicated convolutional neural network (CNN) architectures, we show the potential of a variational model based on generic video information from motion and color. Such cues are usually required for many tasks such as robot navigation or grasp estimation. We leverage them directly for video object segmentation and thus provide accurate segmentations at potentially very low extra cost. Our simple method can provide competitive results compared to the costly CNN-based methods with parameter tuning. Furthermore, we show that our approach can be combined with state-of-the-art CNN-based segmentations to improve their respective results. We evaluate our method on the datasets DAVIS_{2016}, DAVIS_{2017} and SegTrack v2 (see Section 2.1). This work is published in the International Conference on Pattern Recognition (ICPR), 2021 [84].

3.1 Introduction

Object detection and segmentation play a crucial role in applications such as grasp estimation, Hasegawa et al. [58], affordance detection, Nguyen et al. [118], or human-robot interaction, Siam et al. [141]. While these steps are generally challenging on their own, they become even more so when we assume automotive settings in dynamic environments. Then,

potentially moving objects of interest are to be segmented and tracked from video under camera ego-motion. High-end computer vision algorithms on this task usually rely on an object and video-specific training such as Nagaraja et al. [117], Perazzi et al. [127], Voigtlaender and Leibe [155] of CNNs and show, with few exceptions, limited applicability to online settings while they come at high computational costs.

Despite generally good results, for example, on the challenging DAVIS video segmentation benchmark, Perazzi et al. [128], Wang et al. [157], the boundary localization and occlusion handling are far from being solved by these approaches. However, off-the-shelf deep learning-based approaches to low-level tasks such as boundary prediction, Maninis et al. [109], Xie and Tu [161], and optical flow estimation, Ilg et al. [65, 66], produce highly accurate image and motion boundaries. At the same time, such low level information is a basic component in state-of-the-art approaches to robot navigation, Kendoul et al. [85], Zingg et al. [170], McGuire et al. [114], grasp estimation, Hasegawa et al. [58], and visual SLAM, Lim et al. [106]. Thus, their computation comes at little to no extra cost in many practical settings.

In this chapter, we provide a lightweight variational formulation that can leverage low-level cues such as boundary and optical flow estimations from generic models and incorporate them into a simple frame-by-frame label propagation framework.

Our model facilitates the segmentation of fine details and thin structures. Furthermore, since our framework allows for modular integration of low-level cues, it can function as an evaluation platform for such cues with respect to video segmentation applications.

None of the currently evaluated optical flow or boundary estimation methods are trained or finetuned on the relevant datasets used in this chapter. We thus prove the potential of *generic* low-level cues for object segmentation tracking and show that the gap to highly optimized CNN methods is negligible.

Additionally, we evaluate the proposed variational method as a postprocessing step for such highly optimized CNN-based models currently defining the state-of-the-art on the DAVIS_{2016} [128] and DAVIS_{2017} dataset [131], (see Section 2.1), and show an improvement of the segmentation quality in this scenario. This experiment proves that off-the-shelf boundary and motion estimates carry complementary information currently not captured in dedicated CNN-based methods.

Contributions. In summary, our main contributions are:

– We provide a lightweight formulation for video object segmentation using variational label propagation. Once optical flow and boundary estimates are computed (*without sequence/dataset specific finetuning*), our approach facilitates the online generation of tracked video object segmentations within milliseconds.

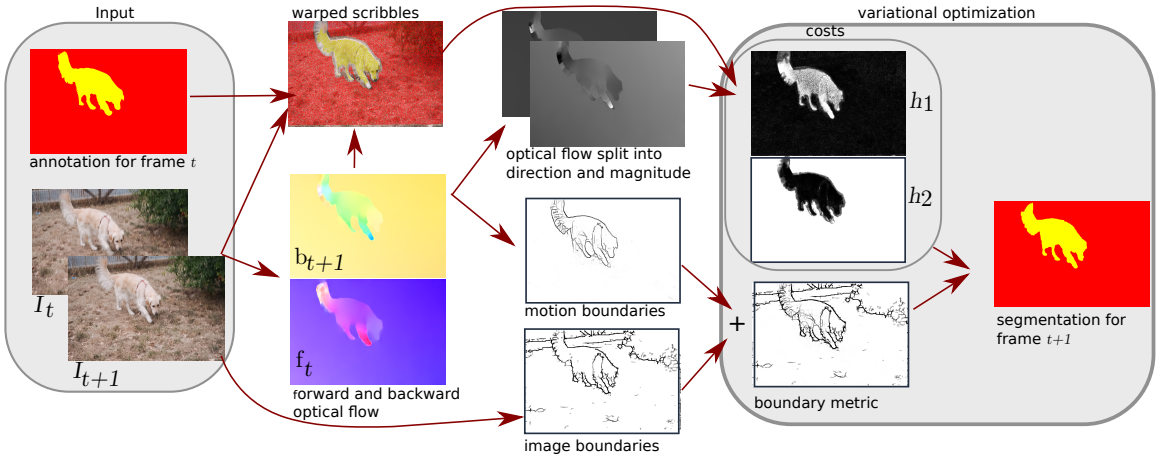


Fig. 3.1 Visualization of the proposed workflow is given. Starting from the images in frame t and $t+1$ and an initial annotation, scribbles are extracted based on optical flow. Then, warped scribbles, image color, and optical flow values are used to generate label costs and boundary estimates to be fed into a variational segmentation framework which produces the complete segmentation of frame $t+1$.

- We incorporate optical flow and color features to facilitate the retrieval of lost objects due to intermediate tracking mistakes or full object occlusion.
- We study the effect of different state-of-the-art boundary, Dollár and Zitnick [41], Xie and Tu [161], Maninis et al. [109], and optical flow estimation methods in our proposed formulation, Ilg et al. [66, 65].
- Our approach can be used to refine state-of-the-art CNN-based results whenever available.

We evaluate our method on the video object segmentation datasets DAVIS_{2016} [128], DAVIS_{2017} [131] and SegTrack v2 [104] (see Section 2.1) and provide an ablation study on the impact of all employed cues on DAVIS_{2016}. Our approach yields competitive results on SegTrack v2 and can compete with several but not all CNN-based methods on the DAVIS benchmarks. Used as a post-processing, our formulation allows us to improve existing results and provides fine object details.

3.2 Related Work

Various variational formulations have been proposed for multi-label segmentation in still images, e.g. in Nieuwenhuis and Cremers [119], Bae et al. [8]. In Nieuwenhuis and Cremers [119], image segmentation from user scribbles is addressed in a variational framework

considering the spatial and color information. In Müller et al. [116] such methods have been applied to produce dense video segmentations from sparse seeds in a frame-by-frame manner, based on automatically generated seeds from point trajectories, Keuper et al. [88], Keuper [87], (see Section 2.4.1). In contrast to Nieuwenhuis and Cremers [119], Müller et al. [116], we directly introduce highly informative low-level cues into the variational formulation. Our variational formulation is derived from Nieuwenhuis and Cremers [119]. Yet, we don't require user scribbles and use optical flow to propagate labels semi-densely across frames.

Label propagation by optical flow has been previously used for example in Tsai et al. [153], Price et al. [132], Nagaraja et al. [117], Paul et al. [126]. Unlike Price et al. [132], which exclusively utilizes temporal coherence, Nagaraja et al. [117] only uses color consistency. Our approach employs optical flow to propagate the labels through consecutive frames and, additionally, to provide information for the data term of our formulation (see Section 3.3.2).

The problem of label propagation in videos has also been addressed by deep learning approaches like Perazzi et al. [127], Voigtlaender and Leibe [155], Bao et al. [13], Maninis et al. [110], Siam et al. [141]. Such networks are trained on specific datasets and the first frame annotation of a sequence to produce a segmentation of subsequent frames. Optical flow magnitudes are employed as additional input for the network, e.g. in Voigtlaender and Leibe [155], to provide additional saliency cues. However, the exact localization quality of optical flow is hardly used. Notably, He et al. [59] use optical flow to create patch correspondences in a video to improve the training of deep neural networks. Jampani et al. [71] use the similarity of features in neural networks to disseminate information in videos. In Maninis et al. [110], a spatio-temporal Markov Random Field (MRF) model is defined over pixels to produce temporally consistent video object segmentation. In their approach, spatial dependencies among pixels are encoded by a CNN trained for the specific target sequence. In contrast, the OSVOS-S approach from Maninis et al. [110] can be considered to be fully complementary. They propose a one-shot video object segmentation framework that explicitly does not rely on any temporal consistency within the data, such that object occlusions and dis-occlusions can be handled particularly well. In contrast, OSVOS-S [110] successively transfers generic, pre-trained, semantic information to the task of video object segmentation by learning the appearance of the annotated (single) object of the test sequence. To show the benefit of our model for the refinement of CNN predictions, we particularly evaluate on OSVOS-S, which can be considered most complementary.

3.3 Proposed Approach

In the following, we describe the details of our approach. Figure 3.1 gives an overview of its workflow. We assume that an image sequence $I_1, \dots, I_\ell$ is given, where ℓ is the number of frames. We further assume that we are given a full annotation $\mathscr{S}^t = \cup_{i=1}^n \mathscr{S}_i^t$ with n segments for frame t, where $\cap_{i=1}^n \mathscr{S}_i^t = \emptyset$. The task is to subsequently infer the segmentation of the remaining frames using this first annotation. The proposed method computes optical flow in forward and backward direction to infer label scribbles for I_{t+1} from I_t and $\mathscr{S}^t$. Optical flow is also used, along with pure image information, to generate costs for all labels and to extract motion boundaries for exact object delineation. In conjunction with generic image boundaries, these cues are used to generate the full segmentation of I_{t+1} using a variational formulation.

3.3.1 Confident Label Propagation with Optical Flow

The optical flow is explained in Section 2.2 and is used for assigning a displacement vector to every point in the image domain Ω. Given the pair of frames, for points $\mathbf{y}$ that are visible in both frames t and $t+1$, the distance $d(\mathbf{f},\mathbf{b},\mathbf{y}) = \|\mathbf{y} - \mathbf{f}_t(\mathbf{b}_{t+1}(\mathbf{y}))\|_2$ equals to zero, see Figure 3.2. For those points, the label from location $\mathbf{b}_{t+1}(\mathbf{y})$ in frame t can be directly transferred to location $\mathbf{y}$ in frame $t+1$. Whenever a point $\mathbf{y}$ is occluded in frame t, this is no longer valid. In these cases, $d(\mathbf{f},\mathbf{b},\mathbf{y}) > 0$. For those locations, the label of $\mathbf{y}$ in $t+1$ needs to be inferred from other cues.

For real world image sequences, optical flow estimations are often not perfectly accurate such that $d(\mathbf{f},\mathbf{b},\mathbf{y}) > 0$ for almost all $\mathbf{y} \in \Omega$. Thus, there is need for a heuristic on the matching confidence, which is defined by $\text{conf}(\mathbf{f},\mathbf{b},\mathbf{y})$ in Equation (3.1), Sundberg et al. [144]. In practice, we assume the optical flow matching to be confident (i.e. $\text{conf}(\mathbf{f},\mathbf{b},\mathbf{y}) = 1$) if $d(\mathbf{f},\mathbf{b},\mathbf{y})$ is sufficiently small and set

$$\text{conf}(\mathbf{f},\mathbf{b},\mathbf{y}) = \begin{cases} 1, & \text{if } d(\mathbf{f},\mathbf{b},\mathbf{y}) < \tau, \\ 0, & \text{otherwise,} \end{cases} \tag{3.1}$$

with a small threshold τ. For confident regions in frame t, labels from uniformly sampled points are propagated to frame $t+1$ and considered scribble points. Refer to the Figure 3.1 (left) for a visualization. These propagated labels are used for the data term (cost) creation of the variational formulation (see Section 3.3.2). Additionally, the direct warping of labels in regions with high confidence renders our approach very efficient: we only need to infer

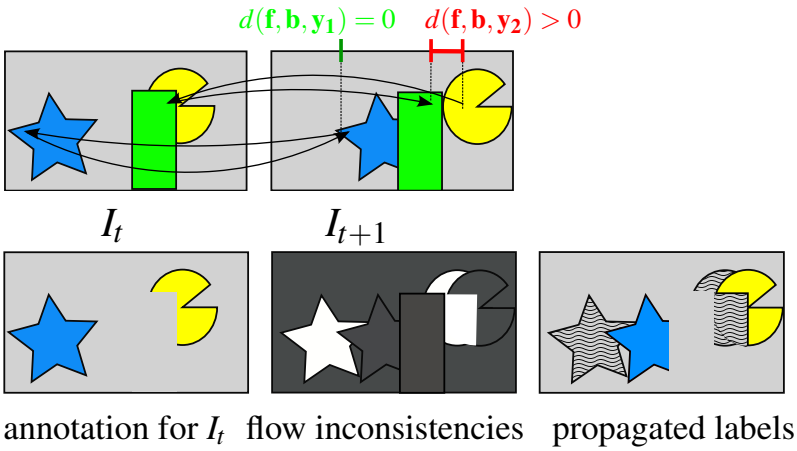


Fig. 3.2 Visualization of the forward-backward consistency of the optical flow and the employed label warping is provided. For input frames I_t and I_{t+1} (row 1), we check the point motion according to the backward and forward optical flow fields $\mathbf{b}_{t+1}$ and $\mathbf{f}_t$ for cycle consistency. For disoccluded points $\mathbf{y}$ in I_{t+1} the distance $d(\mathbf{f},\mathbf{b},\mathbf{y})$ is large. In corresponding regions, no labels can be propagated.

labels for a small fraction of the image. The intuition of this procedure is the following: Whenever we are certain that points $\mathbf{y}$ in frame $t+1$ and $\mathbf{x}$ in frame t refer indeed to the same real-world object point, the label of $\mathbf{x}$ should be propagated irrespective of any other features. It is sufficient to infer labels for non-confident regions.

3.3.2 Variational Formulation

We follow the work of Chambolle et al. [29] and formulate the multiple-label segmentation problem as a minimal partitioning problem. The objective is to partition the image domain $\Omega \subset \mathbb{R}^2$ into $\Omega_1, \ldots \Omega_n \subset \mathbb{R}^2$ such as to optimize

$$\min_{\Omega_1,\ldots,\Omega_n \subset \Omega} \frac{\lambda}{2} \sum_{i=1}^{n} \mathrm{Per}(\Omega_i;\Omega) + \sum_{i=1}^{n} \int_{\Omega_i} h_i(\mathbf{x})\, d\mathbf{x}, \qquad (3.2)$$
$$\text{s.t.} \quad \Omega = \cup_{i=1}^{n} \Omega_i, \quad \forall i \neq j \quad \Omega_i \cap \Omega_j = \emptyset.$$

The potential functions $h_i : \mathbb{R} \to \mathbb{R}_+$ represent the costs for each individual pixel to be assigned to label i, and $\mathrm{Per}(\Omega_i;\Omega)$ is the perimeter of region i in Ω. Usually, the perimeter is measured according to an underlying image-induced metric, Nieuwenhuis and Cremers [119]. The regularization parameter λ steers the penalization of longer boundaries. For an

image $I : \Omega \to \mathbb{R}_+^d$ with d channels, a common weighting function is

$$\mathrm{Per}(\Omega_i;\Omega) = \int_{\Omega_i} \exp(-\gamma|\nabla I(\mathbf{x})|)\, d\mathbf{x} \tag{3.3}$$

where ∇I is the Jacobian of I, $|\nabla I|$ denotes its Frobenius norm, and γ is a positive scalar. If partial annotations $\mathscr{S}_i \subset \Omega$ of I are provided for labels i, the potential functions h can be defined as spatially varying color or feature distributions like in Nieuwenhuis and Cremers [119]

$$h_i(\mathbf{x}) = -\log \frac{1}{|\mathscr{S}_i|} \int_{\mathscr{S}_i} G_{\rho_i} \cdot G_\sigma \, d\mathbf{x}_{S_i}, \tag{3.4}$$

with $G_{\rho_i} = k_{\rho_i}(\mathbf{x} - \mathbf{x}_{S_i})$ and $G_\sigma = k_\sigma(I(\mathbf{x}) - I(\mathbf{x}_{S_i}))$. Here $|\mathscr{S}_i|$ denotes the area occupied by label i, k_σ and k_{ρ_i} are Gaussian distributions in the feature and the spatial domain, respectively. Usually, color is used as pixel features. It can be complemented for example with cues from optical flow such as its magnitude or direction.

The subscripts ρ_i and σ denote the respective standard deviations. The parameter ρ_i is assigned based on the Euclidean distance of the unlabeled feature points in I to each of the scribble points for label i and σ is assigned experimentally. By $\mathbf{x}_{S_i}$ and $I(\mathbf{x}_{S_i})$ we represent the position (x,y) and feature (e.g. *RGB*) information of the partial annotation $\mathscr{S}_i \subset \Omega$ in the image I, respectively. Equation (3.4) shows how spatial and color features of the partial annotations are used to generate costs for each label i in image I.

3.3.3 Flow Magnitude and Flow Direction

Besides being useful for the tracking of ego-motion and 3D scene reconstruction, for example, by visual SLAM [106], optical flow information, (see Section 2.2), is a straight-forward cue for video label propagation. Here, we additionally leverage optical flow information in the data term h_i Equation (3.4) to create the label costs. Such information is expected to provide: 1) cues for object saliency and 2) cues for the object label, since motion only changes gradually over time. Hence, we concatenate the normalized flow magnitude ($\mathbf{f}_{\mathrm{mag}}$) and direction ($\mathbf{f}_{\mathrm{dir}}$) to the original color information of the image in frame $t+1$ for the segmentation. With this additional information, $I : \Omega \to \mathbb{R}_+^3$ in Equation (3.4) is replaced by

$$J := \begin{bmatrix} I \\ \alpha \cdot \mathbf{f}_{\mathrm{mag}} \\ \theta \cdot \mathbf{f}_{\mathrm{dir}} \end{bmatrix}, \tag{3.5}$$

where $\mathbf{f}_{\text{mag}} : \Omega \to \mathbb{R}_+$, $\mathbf{f}_{\text{dir}} : \Omega \to \mathbb{R}_+$ and α and θ are weighting factors which are assigned as 0.5 to account for the strong expected correlation between color values. Thus, the range of values in the *RGB* channels are between 0 and 255 while values in $\mathbf{f}_{\text{mag}}$ and $\mathbf{f}_{\text{dir}}$ range between 0 and 127.5. Finetuning of these parameters for specific datasets is possible and will most likely improve the results. However, the proposed approach attempts not to fit such parameters to any specific dataset for simplicity. It is important to notice that the produced optical flow estimations are not always accurate. More specifically, the flow information is produced with models trained on common optical flow benchmark datasets, which differ from the datasets we evaluate and use in our approach. Yet, our model benefits even from noisy optical flow estimations.

3.3.4 Boundary Term

In the variational formulation from Equation (3.2), the perimeter "Per" is computed based on an image induced metric such as given in Equation (3.6). This metric can be replaced by more evolved, learning-based boundary estimations $\mathscr{E} : \Omega \to \mathbb{R}_+$ such as Dollár and Zitnick [41], Maninis et al. [109], Xie and Tu [161]. For example, Müller et al. [116] propose to weight the region boundaries according to pseudo-probabilities

$$g(\mathbf{x}) = \exp(\mathscr{E}(\mathbf{x})^{\beta} / \bar{\mathscr{E}}), \tag{3.6}$$

with $\bar{\mathscr{E}} := \frac{2}{\Omega} \int_{\Omega} |\mathscr{E}(\mathbf{x})| d\mathbf{x}$ and $\beta > 0$, and employ boundary estimates from Dollár and Zitnick [41] for the generation of object segmentations.

However, any approach to image (e.g. HED [161]), object (e.g. COB [109]), or motion (e.g. FlowNet3.0 [66]) boundary estimation could be used. We study the effect of each of the mentioned boundary estimation methods on the validation set of DAVIS_{2016} in Section 3.5.1. For this study, off-the-shelf trained models of Dollár and Zitnick [41], Xie and Tu [161], Maninis et al. [109], Ilg et al. [66] are employed to generate boundary estimates (refer to Figure 2.6). Please notice that none of the employed models is trained nor finetuned on the datasets under consideration.

3.3.5 Lost Object Retrieval

In the video object segmentation scenario, objects can become partially or fully occluded for several video frames. In this case, the respective label gets lost. Figure 3.3 illustrates this problem: The foreground object (a soccer ball) moves to the left and is partially covered by a

frame 1 frame 5 frame 6

Fig. 3.3 The "soccerball" sequence from DAVIS$_{2016}$ [128] provides an example of an object reappearing after occlusion. For such objects, no labels can be propagated.

tree. As it reappears, no labels can be propagated. We propose a simple approach to Lost Object Retrieval (LOR), which fixes this issue in many practical scenarios.

We create partial annotations of the missing object using the confidence values from Equation (3.1) and the color information given in the annotated keyframe. As soon as the object reappears (i.e. is disoccluded) in frame I_{t+1} to be segmented, the confidence of the label propagation in the respective image area should be low, since $d(\mathbf{f},\mathbf{b},\mathbf{y})$ is high in case of disocclusion (see Figure 3.2).

Thus, we select all positions with low confidence for the label propagation as candidates for LOR. Then, we compute the color similarity of the positions in I_{t+1} with the lost object's mean color extracted from the annotated keyframe. Finally, we create partial labels for the object using the calculated color similarities by selecting the points with a color distance below a predefined threshold (here set to 5.0). Such a low value is necessary to prevent wrong partial label generation and ensures that we retrieve lost objects whenever the color similarity of respective regions is high. This approach works well in practice. However, it might fail when different objects on a video are similar in color. Here, we apply LOR in the binary segmentation scenario only.

3.4 Implementation Details

Parameter Settings. Our approach has several parameters. The γ in Equation (3.3) is trivially set to $\frac{1}{255}$ and the normalization factors α and θ in Equation (3.5) are set to 0.5 for all datasets. The λ in Equation (3.2) is determined via grid search in the interval $\{5,10,...,60\}$ for the first two images of each sequence: We assume that the object size does not change drastically in two consecutive frames, and thus select the λ yielding the smallest deviation in size between the foreground objects in the generated segmentation and the first frame annotation. We set τ in Equation (3.1) to a fixed value of 5 for DAVIS$_{2016}$ and DAVIS$_{2017}$. For SegTrack v2, τ is set to the mean optical flow magnitude to account for strong motion variations. The value σ in Equation (3.4) is set to 64 for all sequences in all

Table 3.1 Our results for different boundary estimation methods on the DAVIS_{2016} validation set are given. Motion boundaries (MB)s from Ilg et al. [66] are studied when combined (**w/** MB) or not combined (**w/o** MB) to each of the boundary detectors.

	$F(\%)$		$J(\%)$	
Method	**w/o** MB	**w/** MB	**w/o** MB	**w/** MB
SED [41]	56.72	64.71	62.64	69.09
HED [161]	59.77	67.33	63.65	70.99
COB [109]	**60.47**	**68.39**	**63.71**	**71.58**

datasets. We expect that parameter finetuning would improve the results further. Yet, we skip this step for simplicity.

Optimization. Assigning labels to each object is an optimization problem that we solve using the iterative primal-dual algorithm of Chambolle et al. [29]. Stopping criteria for this iterative approach are based on a maximum number of iterations initially set to 3000. It is increased to 6000 whenever the calculated objective value in iteration 3000 is above 600000. The optimization stops earlier when the decrease in objective value between consecutive iterations is below 10. The computation time for the optimization is proportional to the number of iterations and objects to be segmented.

3.5 Experiments and Results

We evaluate our method on binary and multi-object segmentation tasks on the DAVIS_{2016} [128], DAVIS_{2017} [131] and SegTrack v2 [104] datasets. The datasets and evaluation metrics are illustrated in Section 2.1.

3.5.1 Ablation Study

In the following, we evaluate the impact of the employed cues such as boundary terms, optical flow, and lost object retrieval to our model on the DAVIS_{2016} dataset.

Boundary Terms

The boundary term used in Equation (3.6) is crucial to our approach. We evaluate segmentation results when [41] (*SED*), [161] (*HED*), and [109] (*COB*) are used directly and when they are combined with motion boundaries extracted from FlowNet3.0 [66] (see Figure 2.6).

Table 3.2 Our results for train and validation sets of DAVIS_{2016} are given when: 1. using FlowNet2.0 [65] instead of FlowNet3.0 [66], 2. not using lost object retrieval (**w/o** LOR), 3. employing different components of spatial, color and optical flow information.

	$F(\%)$		$J(\%)$	
Method	train	val	train	val
FlowNet2.0 [65]	71.98	68.05	75.64	71.46
w/o LOR	67.50	63.81	72.21	68.30
w/o $\mathbf{f}_{\text{mag}} + \mathbf{f}_{\text{dir}}$	69.66	58.17	68.36	59.51
w/o $\mathbf{f}_{\text{dir}}$	71.93	67.19	75.24	69.45
our full model	**73.49**	**68.39**	**77.68**	**71.58**

In this case, motion boundaries are simply summed up before non-maximum suppression. All boundary estimations are generated based on existing models, including Ilg et al. [66], who directly estimate motion boundaries along with the optical flow. We emphasize that none of these models is trained on $\text{DAVIS}_{2016,2017}$ nor SegTrack v2. In Table 3.1, we report the resulting F-measure (F) and Jaccard's index (J) values (see Section 2.1). The combination of *COB* and motion boundaries works best. All further results are based on this setting.

Flow Estimation Methods

Optical flow information is a central component of our model. It is used to 1) generate scribble points for subsequent frames, 2) compute the data terms h_i in the variational optimization, and 3) to compute motion boundaries to complement generic image boundaries (refer to Figure 2.6). Only a few optical flow methods produce motion boundary estimates directly as an additional output, which is why our setup is based on FlowNet3.0 [66]. However, motion boundaries can be computed from strong gradients of any estimated optical flow field, such as generated by Weinzaepfel et al. [159], Bailer et al. [10], Ilg et al. [65]. Thus, we compare here the performance of our full model when we replace all optical flow information from FlowNet3.0 [66] with FlowNet2.0 [65]. The results in Table 3.2 show a decrease in the segmentation accuracy. Since the difference in optical flow quality is known to be small between the two approaches, the decrease in segmentation quality indicates a rather strong impact of the better motion boundaries from FlowNet3.0.

Fig. 3.4 Exemplary results for segmentation tracking on the DAVIS_{2016} (binary) and DAVIS_{2017} (multi-label) benchmark are shown from different sequences and the ground-truth (GT). We compare different state-of-the-art methods like OSVOS-S [110], CINM [13], OSMN [164], and **ours**.

Lost Object Retrieval

In Table 3.2, we evaluate the impact of lost object retrieval (LOR) on our method. The numbers indicate a significant improvement in the segmentations due to LOR. Specifically, the results of our model improve by $3-5\%$ on the train and validation sets of DAVIS_{2016} when LOR is added.

Data Term

In Table 3.2, we provide an ablation study on the data term creation. To do so, we remove from our *full model*, the optical flow direction (**w/o** $\mathbf{f}_{\text{dir}}$) and both the optical flow direction and magnitude (**w/o** $\mathbf{f}_{\text{mag}} + \mathbf{f}_{\text{dir}}$). In this case, only color information is used. Our full model performs better than the two alternatives, thus both $\mathbf{f}_{\text{mag}}$ and $\mathbf{f}_{\text{dir}}$ provide meaningful segmentation cues.

3.5.2 Results on DAVIS

In Table 3.3, we report the mean (M), recall (R) and decay (D) values for F-measure (F) and Jaccard's index (J) over the DAVIS_{2016} dataset splits for the state-of-the-art methods OSVOS-S [110], MSK [127], VPN [71], SIAMMASK [156], CTN [72], PLM [168], OFL [153], BVS [111], FCP [129], and JMP [47], as well as for our approach. The main difference between ours and the competing methods is that we neither learn a model such as OSVOS-S, MSK, VPN, SIAMMASK, CTN, or PLM nor work at the super-pixel level and iteratively optimize the optical flow for the achieved segmentation, as is the case for OFL. We only use tracking

Table 3.3 Results on train and validation sets of $DAVIS_{2016}$ are provided. We report Mean (M), Recall (R) and Decay (D) of the evaluation metrics (F and J 2.1).

	train						
	Measure	$F(\%)$			$J(\%)$		
		M	R	D	M	R	D
CNN	MSK [127]	76.1	88.9	9.8	**80.7**	93.9	8.8
CNN	VPN [71]	**77.0**	**94.3**	13.1	78.3	**95.4**	7.2
CNN	CTN [72]	72.8	88.3	14.7	76.9	90.0	13.5
non-CNN	OFL [153]	70.9	83.1	21.9	73.2	83.0	20.2
non-CNN	BVS [111]	70.1	83.7	25.1	70.9	82.7	24.1
non-CNN	FCP [129]	58.3	67.6	**7.2**	66.2	82.0	**6.5**
non-CNN	JMP [47]	62.3	73.2	36.5	63.2	73.7	35.8
non-CNN	*ours*	**73.5**	**88.6**	13.8	**77.7**	**90.2**	12.5
	validation						
	Measure	$F(\%)$			$J(\%)$		
		M	R	D	M	R	D
CNN	OSVOS-S [110]	87.5	**95.9**	8.2	85.6	96.8	5.5
CNN	[110] + *ours*	87.6	**95.9**	8.1	**86.0**	**96.9**	5.6
CNN	CINM [13]	85.0	92.1	14.7	83.4	94.9	12.3
CNN	[13] + *ours*	**87.7**	93.0	14.3	84.2	95.6	12.1
CNN	MSK [127]	75.4	87.1	9.0	79.7	93.1	8.9
CNN	VPN [71]	65.5	69.0	14.4	70.2	82.3	12.4
CNN	SIAMMASK [156]	67.8	79.8	2.1	71.7	86.8	3.0
CNN	CTN [72]	69.3	79.6	12.9	73.5	87.4	15.6
CNN	PLM [168]	62.5	73.2	14.7	70.2	86.3	11.2
non-CNN	OFL [153]	63.4	70.4	27.2	68.0	75.6	26.4
non-CNN	BVS [111]	58.8	67.9	21.3	60.0	66.9	28.9
non-CNN	FCP [129]	49.2	49.5	**−1.1**	58.4	71.5	**−2.0**
non-CNN	JMP [47]	53.1	54.2	38.4	57.0	62.6	39.4
non-CNN	*ours*	**68.4**	**78.4**	17.8	**71.6**	**81.0**	16.8

for segmentation. Yet, our method outperforms all remaining non-deep-learning-based approaches, and CNN approaches such as PLM [168] or SIAMMASK [156]. Qualitative results are shown in Figure 3.4. Visually, the generated segmentation results are appealing, and fine details are well captured.

Table 3.4 Results on the $DAVIS_{2017}$ validation and test set are provided.

	$F(\%)$		$J(\%)$	
Method	val	test	val	test
ours	56.5	44.0	54.5	41.5
OSMN [164]	57.1	44.9	52.5	37.7
FAVOS[33]	61.8	44.2	54.6	42.9
OSVOS-S [110]	71.3	62.1	64.7	52.9
OSVOS-S [110] + *ours*	71.4	62.2	65.2	53.7
CINM [13]	74.0	70.5	67.2	64.5
CINM [13] + *ours*	**74.1**	**70.6**	**67.4**	**64.7**

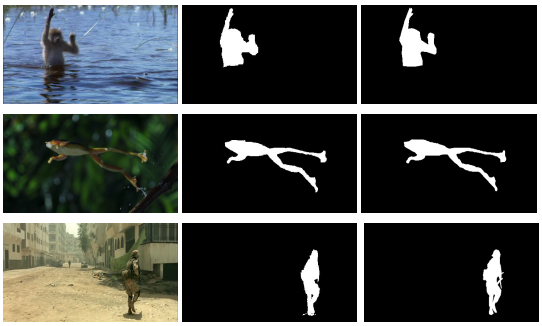


image proposed result ground-truth

Fig. 3.5 Sample results on the SegTrack v2 benchmark are shown (see Section 2.1).

Cost Terms from CNN Segmentation

Here, we evaluate the proposed method as a postprocessing step for state-of-the-art CNN predictions. If our method indeed carries complementary information to the appearance cues learned, for instance, by OSVOS-S [110], we should be able to achieve an improvement. In Table 3.3 and Table 3.4, we specifically report the results of our method when we use plain CNN predictions from OSVOS-S [110] and CINM [13] as data terms (costs) (OSVOS-S + *ours* and CINM + *ours*) in the $DAVIS_{2016}$ and $DAVIS_{2017}$ datasets, respectively. In all settings, segmentation results are improved by our method.

3.5.3 Results on SegTrack v2

In the following, we evaluate our proposed model on the SegTrack v2 dataset (see Section 2.1). In Table 3.5, we compare our results to the state-of-the-art methods JOTS [160], MSK [127] and [104]. Li et al. [104] propose two variants of their method: 1. the online version Segment

Table 3.5 A comparison of our method with several state-of-the-art methods on the SegTrack v2 dataset [104] is provided. The best results are shown with **bold** font on each sequence.

Sequence/Object	SPT+CSI	MSK	JOTS	SPT	*Ours*
Unsupervised	✓	×	×	✓	✓
Online	×	✓	✓	✓	✓
Girl	**89.2**	-	84.6	89.1	62.5
Birdfall	62.5	-	**78.7**	62.0	73.4
Parachute	93.4	-	94.4	93.2	**94.7**
Cheetah-D.	37.3	-	**66.1**	40.1	37.8
Cheetah-C.	40.9	-	35.3	41.3	**56.2**
Monkeydog-M.	71.3	-	**82.2**	58.8	14.7
Monkeydog-D.	18.9	-	21.1	17.4	**30.6**
Penguin-#1	51.5	-	**94.2**	51.4	79.9
Penguin-#2	76.5	-	**91.8**	73.2	78.5
Penguin-#3	75.2	-	**91.9**	69.6	83.1
Penguin-#4	57.8	-	**90.3**	57.6	63.5
Penguin-#5	66.7	-	**76.3**	63.4	64.8
Penguin-#6	50.2	-	88.7	48.6	**89.0**
Drift-#1	**74.8**	-	67.3	73.8	43.7
Drift-#2	60.6	-	63.7	58.4	**80.2**
Hummingb.-#1	54.4	-	**58.3**	45.4	36.8
Hummingb.-#2	72.3	-	50.7	65.2	**77.2**
Frog	72.3	-	56.3	65.8	**79.0**
Worm	**82.8**	-	79.3	75.6	82.7
Soldier	**83.8**	-	81.1	83.0	81.6
Monkey	84.8	-	86.0	84.1	**87.6**
Bird of Paradise	**94.0**	-	93.0	88.2	84.9
BMX-Person	85.4	-	**88.9**	75.1	78.7
BMX-Bike	**24.9**	-	5.70	24.6	9.9
Mean per object	65.9	67.4	71.8	62.7	65.7
Mean per seq.	71.2	-	72.2	68.0	68.6

Pool Tracking (SPT), 2. the offline version with the subsequent refinement of the segments within each frame, i.e., Composite Statistical Inference (CSI). Similar to SPT, our method operates in an online fashion and does not require dataset-specific training. However, we only use tracking information to compute segments, whereas SPT incrementally trains a global model of the appearance of objects. Yet, our approach produces results within the range of the top performing methods and improves over SPT. Several qualitative results for SegTrack v2 are given in Figure 3.5. Our segmentation is able to capture fine details of the

objects, such as the slender legs of the frog (in the *frog* sequence) and the arm and hand of the monkey (in the *monkey* sequence).

3.6 Conclusion

We have proposed a variational method for single and multiple object segmentation tracking scenarios. It leverages optical flow and image boundary estimations for the propagation of labels through video sequences, where a keyframe annotation is provided. Deep learning-based methods address video object segmentation with high computational complexity and sequence-specific training. In contrast, our approach only considers the first frame annotations and achieves competitive results without an expensive training procedure. In application scenarios that require optical flow as an input (for example, for robot navigation), the computation of video object segmentations with our method comes at very low extra costs. Our proposed method produces visually appealing segmentations and preserves fine details on the DAVIS_{2016}, DAVIS_{2017}, and SegTrack v2 datasets.

Chapter 4

Motion Segmentation - Self-Supervised Densification of Sparse Motion Segmentations

In Chapter 3, the importance of the optical flow information and the estimated image boundaries in tracking the segmentation masks is shown. Studying different optical flow and edge map generation methods emphasizes the importance of the underlying techniques for higher-level tasks, such as video object and motion segmentation. In this chapter, we elaborate on the motion segmentation task relying on optical flow estimation and boundary predictions.

Observable motion in videos can give rise to the definition of objects moving with respect to the scene. The task of segmenting such moving objects is referred to as motion segmentation. It is usually tackled either by aggregating motion information in long, sparse point trajectories or by directly producing per frame dense segmentations relying on large amounts of training data (Section 2.4). We propose a self-supervised method to learn the densification of sparse motion segmentations from single video frames. While previous approaches towards motion segmentation build upon pre-training on large surrogate datasets and use dense motion information as an essential cue for pixel-wise segmentation, our model does not require pre-training and operates at test time on single frames. It can be trained in a sequence-specific way to produce high-quality dense segmentations from sparse and noisy input. We evaluate our method on the well-known motion segmentation datasets FBMS59 and DAVIS_{2016}, (see Section 2.1), which is published in Asian Conference on Computer Vision (ACCV), 2020 [81].

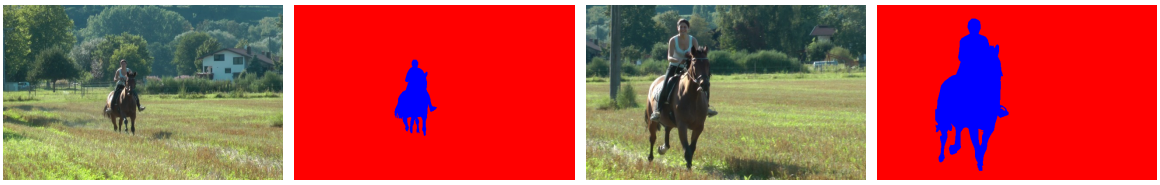


Fig. 4.1 Motion segmentation example is provided for frames 1 and 160 of the "horses01" sequence (with their ground-truth motion segmentation) in the FBMS59 [123] dataset. Due to the reason that the person and the horse are moving together and have the same motion pattern, they are assigned the same motion label (blue color).

4.1 Introduction

The importance of motion for visual learning has been emphasized in recent years. Following the Gestalt principle of common fate from Koffka [95], motion patterns within an object are often more homogeneous than its appearance and provide reliable cues for segmenting (moving) objects in a video. Motion segmentation is the task of segmenting motion patterns and differs from semantic segmentation, where one seeks to assign pixel-wise class labels in an image. Thus, for motion segmentation, we need motion information and at least two frames to be visible to distinguish between motion segments. Ideally, such motion patterns separate meaningful objects, e.g. an object moving w.r.t. the scene, refer to Figure 4.1. To illustrate the importance of motion segmentation, consider an autonomous driving scenario. A first step to classifying potential danger caused by the presence of a possibly static object, for instance, a parking car, is the knowledge about its mobility. Unknowingly waiting for the observation that the door of the parking car suddenly opens may be too late to avoid an accident. The speed of the autonomously driving vehicle must be adjusted based on the mobility and danger of other objects. While models for the direct prediction of pixel-wise motion segmentations are highly accurate, Tokmakov et al. [150, 149], they can only take very limited account of an object's motion history.

In this chapter, we propose a model to produce high quality dense segmentations from sparse and noisy input (i.e. *densification*). It can be trained in a sequence specific way using sparse motion segmentations as training data, i.e. the densification model can be trained in a self-supervised way. Our approach is based on sparse (semi-dense) motion trajectories, (see Section 2.4.1), that are extracted from videos via optical flow, Figure 4.2-middle. Point trajectory based motion segmentation algorithms have proven to be robust and fast like in Ochs et al. [123], Keuper et al. [88], Keuper [87], Brox and Malik [25], Fragkiadaki et al. [52], Shi et al. [137], Rao et al. [133] (see Section 2.4.1). By a long-term analysis of a whole video shot at once by means of such trajectories, even objects that are static for most of the

time and only move for few frames can be identified, i.e. the model would not "forget" that a car has been moving, even after it has been static for a while. The same argument allows articulated motion to be assigned to an ordinary moving object.

In our approach, we use object annotations that are generated using an established, sparse motion segmentation technique, Keuper et al. [88]. We also propose an alternative, a Gated Recurrent Unit (GRU)-based multicut model, which allows to learn the similarity between the motion of two trajectories and potentially allows for a more flexible application. In the next chapter, more than two trajectories are considered to resolve complex and non-translational motion patterns (see Section 5). In both cases, the result is a sparse segmentation of video sequences, providing labels only for points lying on the original trajectories, e.g. every 8 pixels, refer to Figure 4.2-middle. Pixel-wise segmentations from such sparse segmentations can be generated by variational methods like Ochs et al. [123]. To better leverage the consistency within the video sequences, we propose to train sequence-specific densification models using only the sparse motion segmentations as labels.

Specifically, we train a U-Net, Ronneberger et al. [135], like model to predict dense segmentations from given images, see Figure 4.2-right, while we only have sparse and potentially noisy labels. The training task can thus be interpreted as label densification. Yet, the resulting model does not need sparse labels at test time but can generalize to unseen frames.

In contrast to end-to-end motion segmentation methods such as Tokmakov et al. [149], we are not restricted to binary labels but can distinguish between different motion patterns belonging to different objects and instances per image and only require single images to be given at test time. Also, in contrast to such approaches, our model does not rely on the availability of large surrogate datasets such as ImageNet, Deng et al. [39], or FlyingChairs, Dosovitskiy et al. [42], for pre-training but can be trained directly on the sequences from the FBMS59, Ochs et al. [123], and DAVIS_{2016}, Perazzi et al. [128], datasets (see Section 2.1).

Contributions. To summarize, we make the following contributions:

- We provide an improved affinity graph for motion segmentation in the minimum cost multicut framework using a GRU model. Our model generates a sparse segmentation of motion patterns.
- We utilize the sparse and potentially noisy motion segmentation labels to train a U-Net model to produce class agnostic and dense motion segmentation. Sparse motion labels are not required during prediction.

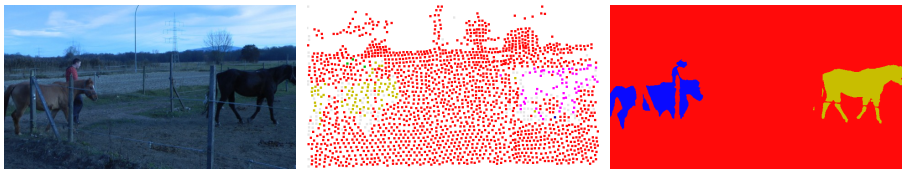


Fig. 4.2 Exemplary multi-label motion segmentation results showing (**left**) the image and its sparse (**middle**) and dense (**right**) segmentation. The sparse segmentation is produced by Keuper et al. [88] and the dense segmentation is the result of the proposed model.

- We provide competitive video object segmentation and motion segmentation results on the FBMS59 [123] and DAVIS_{2016} [128] datasets (see Section 2.1).

4.2 Related Work

Our goal is to learn to segment moving objects based on their motion pattern. For efficiency and robustness, we focus on point trajectory-based techniques as initial cues (refer to Section 2.4.1). Trained in a sequence specific way, our model can be used for label *densification*. We, therefore, consider the following related work in the *motion segmentation* and *sparse to dense labeling* areas.

4.2.1 Motion Segmentation

Our GRU approach uses the same graph construction policy as Keuper et al. [88] for motion segmentation, (Section 2.4), while the edge costs are assigned using a Siamese (also known as a twin) gated recurrent network. The Siamese networks, Chopra et al. [35], are metric learning approaches used to provide a comparison between two different inputs. Similar trajectory embeddings have been used in Bhattacharyya et al. [19] to predict a pedestrian's future walking direction. For motion segmentation, we stick to the formulation as a minimum cost multicut problem (**MP**), Keuper et al. [88], (see Section 2.5).

4.2.2 Sparse to Dense Labeling

While the trajectory-based motion segmentation methods can propagate object information through frames, they produce sparse results. Therefore, specific techniques, e.g. Ochs et al. [123], Müller et al. [116] are needed to produce dense results. A commonly used densification approach is the variational framework of Ochs et al. [123]. In this approach, the underlying

color and boundary information of the images are used for the diffusion of the sparsely segmented trajectory points, which sometimes leak the pixel labels to unwanted regions, e.g., loosely textured areas of the image. Furthermore, Fragkiadaki et al. [51] address the densification problem using Gabriel graphs as per frame superpixel maps. Gabriel edges bridge the gaps between contours using geometric reasoning. However, super-pixel maps are prone to neglect the fine structure of the underlying image and lead to low segmentation quality.

Our method benefits from trajectory-based methods for producing a sparse multi-label segmentation. A sparsely trained U-Net [135] produces dense results for each frame purely from appearance cues, potentially specific for a scene or sequence.

4.3 Proposed Self-Supervised Learning Framework

We propose a self-supervised learning framework for sparse-to-dense segmentation of the sparsely segmented point trajectories. In other words, a U-Net model is trained from sparse annotations to estimate dense segmentation maps (see Section 4.3.2). The sparse annotations can be provided either with some potentially unsupervised state-of-the-art trajectory segmentation methods as in Keuper et al. [88] or our proposed Siamese-GRU model.

4.3.1 Annotation Generation

Point trajectories are generated from optical flow estimation method of Brox and Malik [25] (Section 2.4.1). Such point trajectories are clustered by the MP approach, Andres et al. [3], (refer to Section 2.5 for the definition of MP), with respect to their underlying motion model estimated (i) from a translational motion model or (ii) from a proposed Siamese GRU network.

Translational Motion Affinities can be assigned based on motion distances of each trajectory pair with some temporal overlap, Keuper et al. [88]. For trajectories, A and B, the frame-wise motion distance at time t is computed by

$$d_t(A,B) = \frac{\|\partial_t A - \partial_t B\|}{\sigma_t}. \tag{4.1}$$

It solely measures in-plane translational motion differences, normalized by the variation of the optical flow σ_t (refer to Ochs et al. [123] for more information). The $\partial_t A$ and $\partial_t B$ represent the partial derivatives of A and B with respect to the time dimension.

The overall motion distance of a pair of trajectories A and B is computed by maximum pooling over their joint lifetime,

$$d^{motion}(A,B) = \max_t d_t(A,B) \tag{4.2}$$

Color and spatial cues are added in Keuper et al. [88] for robustness. Instead of using translational motion affinities, we propose a Siamese-GRU based model to provide associations between the trajectory pairs.

Siamese Gated Recurrent Units (GRUs). In prior work by Keuper et al. [88], the pairwise terms between trajectories were computed based on a translational motion model as in Equation (4.1), and included color and spatial cues. Here, we aim to learn these terms for pairs of trajectories. To learn encodings for these spatio-temporal curves, we use a siamese GRU network, in which each trajectory of a trajectory pair is first encoded individually using shared weights to learn to predict pairwise similarities.

The proposed network consists of two legs with shared weights, whereas in our model, each leg consists of a GRU model, which takes a motion trajectory as input. Specifically, for two trajectories A and B, the ∂A and ∂B (a partial derivative of the trajectories with respect to the time dimension) on the common life-time of the trajectories are given to each leg of the Siamese-GRU model. The partial derivatives represent their motion information, while no information about their location in image coordinates is provided. The motion cues are embedded by the bidirectional-GRU network, i.e. the hidden units are gathered for each time step (the time step N is explained in the next paragraph). Afterward, the difference between two embedded motion vectors $embed_{\partial A}$ and $embed_{\partial B}$ is computed as

$$d_{(embed_{\partial A}, embed_{\partial B})} = \sum_{i=1}^{h} (embed_{\partial A i} - embed_{\partial B i})^2, \tag{4.3}$$

where h denotes the number of hidden units for each time step. The result of the Equation (4.3) is a vector that is consequently given to two fully connected layers and the final similarity value is computed by a Sigmoid function. Therefore, for each pair of trajectories given to the Siamese [35] GRU network, it provides a measure of their likelihood of belonging to the same motion pattern.

The joint lifetime of the two trajectories could, in practice, be different from pair to pair, and the GRU network requires a fixed number of time steps as input (N). This problem is dealt with as follows:

If the joint lifetime of the two trajectories is less than N, each trajectory is padded with its respective final partial derivative value in the intersection. Otherwise, when the joint

lifetime has more than N time steps, the time step t with maximum motion distance, similar to Equation (4.2), is determined for the entire lifetime,

$$t_{A,B} = \arg\max_{t} d_t(A,B). \tag{4.4}$$

The trajectory values are extracted before and after t to reach the required number of time steps N. The reason for doing this is that the vital part of the trajectories is not lost. Consider a case where an object does not move in the first x frames and starts moving from frame $x+1$, essential information will be available around frames x and $x+1$, and it is better not to lose such information by cutting this part out.

In our approach, the frame-wise Euclidean distance of trajectory embeddings (extracted from the hidden units) of the GRU model is fed to a fully connected layer for dimensionality reduction. Consequently, the values are passed to a Sigmoid function for classification into the classes 0 (same label - pair of trajectories belong to the same motion pattern) or 1 (different label - pair of trajectories belong to different motion patterns) using a Mean Squared Error (MSE) loss.

To produce the ground-truth labeling to train the Siamese-GRU model, a subset of the edges in the produced graph $G = (V,E)$ by the method of Keuper et al. [88] are sampled (see Section 2.4.1). For each edge, which corresponds to a pair of trajectories, we look into each trajectory and its label in the provided ground-truth segmentation. We only take those trajectories which belong to precisely one motion pattern in the provided ground-truth. Some trajectories change their labels while passing through frames that are considered unreliable. Furthermore, the same amount of edges with label 0 (same motion pattern) and label 1 (different motion pattern) are sampled to have a balanced training signal. At test time, costs for each edge E in graph $G = (V,E)$ are generated by the trained Siamese-GRU network.

Trajectory Clustering yields a grouping of trajectories according to their modeled or learned motion affinities. We formalize the motion segmentation problem as MP like in Keuper et al. [88]. It aims to optimally decompose a graph $G = (V,E)$, where trajectories are represented by nodes V and their affinities define costs on the edges E. In our approach, the costs are assigned using the Siamese-GRU model.

While the MP is NP-hard, Bansal et al. [12], heuristic solvers like Kernighan and Lin [86], Keuper et al. [89], Beier et al. [15], Bailoni et al. [11] are expected to provide practical solutions. We use the Kernighan Lin [86] implementation from Keuper et al. [88] (refer to Section 2.5.2). This solver is proved to be practically successful in motion segmentation, Keuper et al. [88], Keuper [87], image decomposition and mesh segmentation, Keuper et al. [89], scenarios.

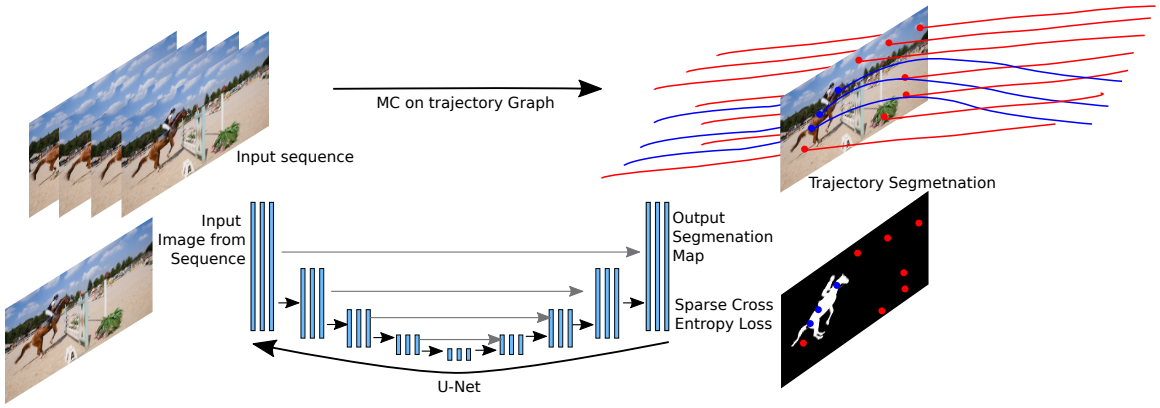


Fig. 4.3 Sparsely segmented trajectories are produced by minimum cost multicut (refered as MC on the figure) either with our Siamese-GRU model or simple motion cues as in Keuper et al. [88] (**top**). The sparsely labeled points are used to train the U-Net model (**bottom**). At test time, the U-Net model can produce dense segmentations without requiring any sparse labels as input.

4.3.2 Deep Learning Model for Sparse to Dense Segmentation

We use the sparse motion segmentation annotated video data as described in Section 4.3.1 for our deep learning-based sparse-to-dense motion segmentation model. Specifically, the training data consist of input images (video frames) or edge maps and their sparse motion segmentations, which we use as annotations. Although the loss function only applies at the sparse labels, the network learns to predict dense segmentations.

Deep Learning Model

We use a U-Net type architecture proposed by Ronneberger et al. [135] for dense segmentation, which is known for its high-quality predictions in tasks such as semantic segmentation, Siam et al. [140, 139], Fu et al. [53]. A U-Net is an encoder-decoder network with skip connections. During encoding, characteristic appearance properties of the input are extracted and are learned to be associated with objectness. In the decoding phase, the extracted properties are traced back to locations causing the observed effect, while details from the downsampling phase are taken into account to ease the localization (see Figure 4.3 for more information). The output is a dense (pixel-wise) segmentation of objects, i.e., a function $u\colon \Omega \to \{1,\dots,K\}$, where Ω is the image domain, and K is the number of classes which corresponds to the number of trajectory labels. This means that after clustering the trajectories, each cluster takes a label, and overall we have class-agnostic motion segmentation of sparse trajectories. Such labels are only used during training.

Loss Function

The U-Net is trained via the Binary Cross Entropy (BCE) and Cross-Entropy (CE) loss function for the single and multiple object cases, respectively. As labels are only present at a sparse subset of pixels, the loss function is restricted to those pixels. Intuitively, since the label locations where the loss is evaluated are unknown to the network, it is forced to predict a label at every location. (A more detailed discussion is provided below.)

Dense Predictions with Sparse Loss Functions

At first glance, a sparse loss function may not force the network to produce a meaningful dense segmentation. Since trajectories are generated according to a simple deterministic criterion, namely extreme points of the local structure tensor, Brox and Malik [25], the network could internally reproduce this generation criterion and focus on labeling such points only. We observed precisely the problematic behavior mentioned above and, therefore, suggest variants for the learning process employing either RGB images or (deterministic) Sobel edge maps, Kanopoulos et al. [75], as input. One remedy is to alleviate the local structure by smoothing the input RGB image, making it harder for the network to pick up on local dominant texture and stimulating the usage of globally extracted features associated with movable object properties.

Conditional Random Filed (CRF) Segmentation Refinement

To build the fine-grained segmentation maps from the blurred images, we employ Conditional Random Fields (CRF): We compare

- the fully connected pre-trained CRF layer (dense-CRF) [97], with parameters learned from pixel-level segmentation [31] and
- a CRFasRNN [169] model, which we train using the output of our U-Net model as unaries on our same sparse annotations. To generate a training signal even in case the U-Net output perfectly fits the sparse labels; we add Gaussian noise to the unaries.

Discussion: Sparse Trajectories vs. Dense Segmentation

The handling of sparse labels could be avoided if dense unsupervised motion segmentations were given. Although, in principle, dense trajectories can be generated by the motion segmentation algorithm, the clustering algorithm does not scale linearly with the number of trajectories, and the computational cost explodes. Instead of densely tracking pixels throughout the video, a frame-wise computationally affordable densification step, for example, based

on variational or inpainting strategies like in Ochs et al. [123], could be used. However, some sparse labels might be erroneous, an issue that can be amplified by densification. Although some erroneous labels can also be corrected by [123], especially close to object boundaries, we observed that the negative effect prevails when it comes to learning from such unsupervised annotations. Moreover, variational methods often rely on local information to steer the propagation of label information in the neighborhood. In contrast, the U-Net architecture can incorporate global information and possibly objectness properties to construct its implicit regularization.

4.4 Experiments

We evaluate the performance of the proposed models on the two datasets DAVIS_{2016} [128] and FBMS59 [123], which contain challenging video sequences with high-quality segmentation annotations of moving objects (Section 2.1).

4.4.1 Implementation Details

Our Siamese-GRU model with 2 hidden units ($h = 2$, Equation (4.3)) and experimentally selected 25 time steps ($N = 25$, for more information refer to Section 4.3.1) is trained for 3 epochs, a batch size of 256 and a learning rate of 0.001 where the trajectories are produced by large displacement optical flow (LDOF), Brox et al. [24], at 8 pixel sampling on the training set of DAVIS_{2016}, Perazzi et al. [128].

We employ two different strategies of using the sparse motion segmentations of the resulting trajectories as labels, depending on whether we face binary (DAVIS_{2016}) or multi-label (FBMS59) problems. In the case of a single label, the most frequent trajectory label *overall frames* and second most frequent label *per frame* are considered as background and foreground, respectively. For multi-label cases, the most frequent class-agnostic labels are selected, i.e., we take only those labels which are frequent enough compared to the other labels.

Our U-Net model is trained in a sequence-specific way. For instance, such a model can be used for label densification and is trained using color and edge-map data with a learning rate of 0.01 and batch size of 1 for 15 epochs. The overall train and prediction process takes around (maximally) 30 minutes per sequence on an NVIDIA Tesla V100 GPU. The CRFasRNN, Zheng et al. [169], is trained with a learning rate of 0.0001, batch size 1 and 10 epochs with 5 and 10 inference iterations at train and test time, respectively.

Table 4.1 The trajectories are segmented by 1. the method of Keuper et al. [88] and 2. our Siamese-GRU model. The densified results are generated based on 1. the method of Ochs et al. [123] and 2. the proposed U-Net model. The results are provided for the validation set of DAVIS_{2016}.

Traj. Seg. Method	Densification Method	Jaccard Index
Keuper et al. [88]	Ochs et al. [123]	55.3
Keuper et al. [88]	U-Net model (ours)	58.5
Siamese-GRU (ours)	Ochs et al. [123]	57.7
Siamese-GRU (ours)	U-Net model (ours)	**66.2**

Table 4.2 Sparse Motion Segmentation trained on DAVIS_{2016} (all sequences) and evaluated on FBMS59 (train set). We compare to Keuper et al. [88] and their variant only using motion cues.

	Precision	Recall	F-measure	#of extracted objs.
Keuper et al. [88] (motion cues)	83.88	69.97	76.30	20
Keuper et al. [88] (full)	83.69	73.17	**78.07**	**27**
Siamese-GRU (ours - transfer)	81.01	70.07	75.14	24

4.4.2 Sparse Trajectory Motion-Model

We first evaluate our GRU model for sparse motion segmentation on the validation set of DAVIS_{2016} [128]. Therefore, in the first iteration, we produce densified segmentations using the variational approach from Ochs et al. [123]. The results are given in Table 4.1 (line 3) and show an improvement over the motion model from Keuper et al. [88] by 2% in Jaccard index. In the following experiments on DAVIS_{2016}, we thus use sparse labels from our Siamese GRU approach.

4.4.3 Knowledge Transfer

Next, we investigate the generalization ability of this motion model. We evaluate the DAVIS_{2016}-trained GRU-model on the train set of FBMS59 [123]. Results are given in Table 4.2. While this model performs below the hand-tuned model of Keuper et al. [88] on this dataset, results are comparable, especially considering that our GRU model does not use color information. In further experiments on FBMS59, we use sparse labels from Keuper et al. [88].

4.4.4 Dense Segmentation of Moving Objects

Next, we evaluate our self-supervised dense segmentation framework with sequence-specific training on the color images as well as edge maps on the validation set of DAVIS_{2016} [128]. Table 4.1 shows that this model, trained on the GRU-based labels, outperforms the model trained on the motion models from Keuper et al. [88] as well as the densification of Ochs et al. [123] by a large margin. Table 4.3 (top) provides a comparison between different versions of our model, the densification model of Ochs et al. [123], and the per-frame evaluation of Tokmakov et al. [150] on DAVIS_{2016}. Tokmakov et al. [150] use large amounts of data for pre-training. Their better-performing model variants require optical flow and full sequence information to be given at test time. Our results based on RGB and edge maps are better than those solely using edge maps. We also compare the different CRF versions:

- pre-trained dense-CRF from Krähenbühl and Koltun [97],
- our trained CRFasRNN from Zheng et al. [169] model trained per sequence *(CRF-per-seq)*,
- CRFasRNN [169] trained on all frames in the train set of DAVIS_{2016} *(CRF-general)* with sparse labels

All CRF versions improve the F-measure. The CRF-general performs on par with dense-CRF by only using our sparse labels for training. See Figure 4.4 and Figure 4.5 for qualitative results of our model on DAVIS_{2016} [128] and FBMS59 [123] datasets, respectively. We show on-par performance with the layered optimization approach by Lao and Sundaramoorthi [99] in DAVIS_{2016} with Jaccard's index of 68.3 versus *ours*: 67.1.

4.4.5 Partly Trained Model

We further evaluate how well our model works for video frames for which no sparse labels are given during training. Please note that, throughout the sequences, the appearance of an object can change drastically. In Table 4.3 (bottom), we report results for the sequence-specific U-Net model + *CRF-general* trained on the first 50%, 70% and 90% of the frames and evaluated on the remaining frames. While there is some loss in Jaccard's index compared to the model evaluated on seen frames (above), the performance only drops slightly as smaller portions of the data are used for training.

Table 4.3 Evaluation of self-supervised training on sequences from $DAVIS_{2016}$ validation and comparison with other methods is provided. Effect of adding color information (RGB) to the edge maps (Sobel) is studied (*ours*) and comparison between (pre-trained) dense-CRF (*dense*), CRF-per-seq (*per-seq*) and CRF-general (*general*) is provided (for different versions of CRF refer to Section 4.4.4). We studied the effect of our best model while training it only on 50%, 70%, and 90% of the frames in the last three rows.

	% of frames	Jaccard Index	F-measure
variational [123]	100	57.7	57.1
appearance + GRU [150]	100	59.6	-
sobel + *dense*	100	62.6	54.0
sobel + RGB (*ours*)	100	61.3	49.0
ours + *dense*	100	**67.1**	60.2
ours + *per-seq*	100	66.2	60.3
ours + *general*	100	66.2	**62.1**
ours + *general*	50	59.6	50.4
ours + *general*	70	62.3	53.5
ours + *general*	90	63.4	55.4

Table 4.4 We evaluate our densification method on FBMS59 (train) using sparse motion segmentations from Keuper et al. [88]. The sparse trajectories are produced with different flow estimation methods (LDOF [24] and FlowNet2 [65]) and densified with our proposed U-Net model (using edge maps (Sobel) and color information (RGB) (*ours*)). Further, we study on different CRF methods, (pre-trained) dense-CRF (*dense*) and CRF-general (*general*). For more details about different versions of CRF refer to Section 4.4.4.

	Precision	Recall	F-measure
Ochs et al. [123]	85.31	68.70	76.11
Lao et al. [99]	90.04	65.09	76.02
LDOF + *ours* + *dense*	89.35	67.67	77.01
FlowNet2 + *ours* + *dense*	89.59	68.29	**77.56**
FlowNet2 + *ours* + *general*	89.27	68.20	77.33

4.4.6 Densification on FBMS59

Next, we evaluate our sequence-specific model for label densification on FBMS59 [123]. We study on two different variants of optical flow (FlowNet2, Ilg et al. [65], and Large Displacement Optical Flow (LDOF), Brox et al. [24]) for trajectory generation and sparse motion segmentation, Keuper et al. [88]. The results in Table 4.4 show that the proposed approach outperforms the approach of Ochs et al. [123] as well as the geometric, layered optimization approach by Lao and Sundaramoorthi [99]. Improved optical flow leads to

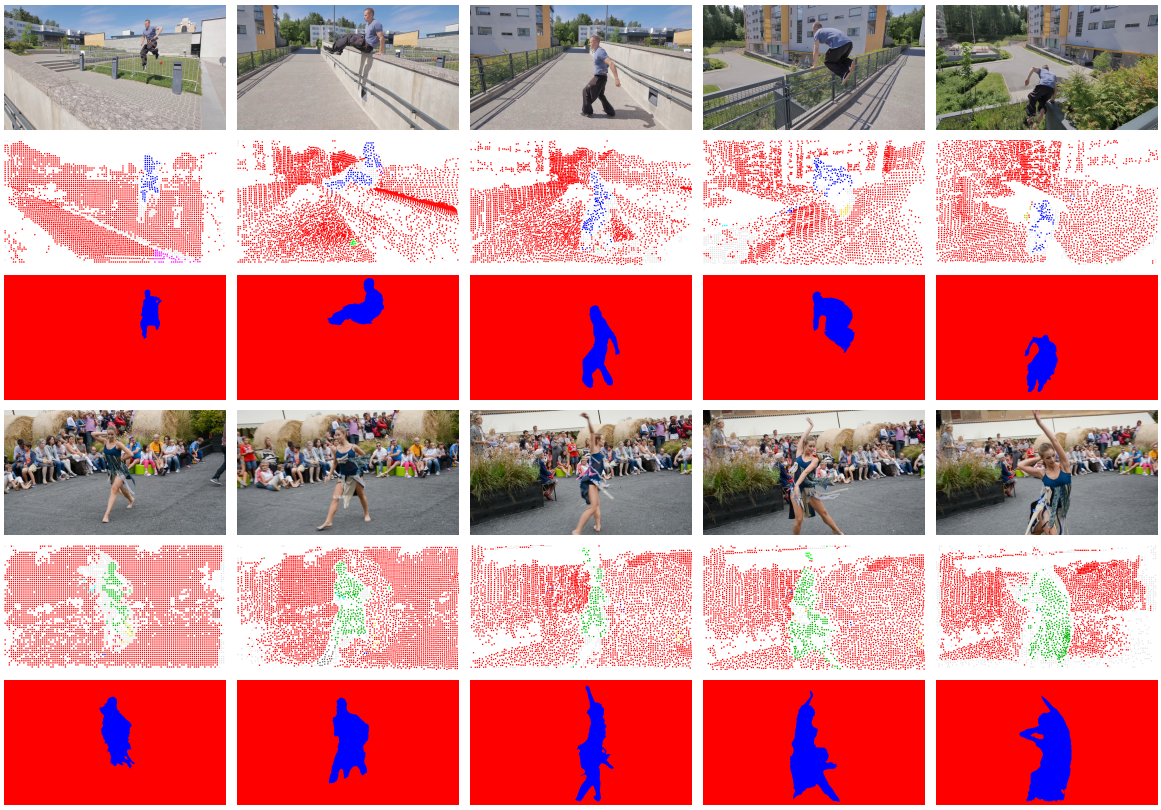


Fig. 4.4 Exemplary single-label motion segmentation results showing the five frames and their sparse and dense segmentation for two different sequences, generated using the proposed U-Net model. The images are from the sequences on the validation set of DAVIS_{2016} [128] dataset.

improved results overall. The different CRF versions do not provide significantly different results.

4.5 Conclusion

In this chapter, we have addressed the segmentation of moving objects from single frames. To that end, we proposed a GRU-based trajectory embedding to produce high-quality sparse segmentations automatically. Furthermore, we closed the gap between sparse and dense results by providing a self-supervised U-Net model trained on sparse labels and relying only on edge maps and color information. The trained model on sparse points provides single and multi-label dense segmentations. The proposed approach generalizes to unseen sequences from FBMS59 and DAVIS_{2016} and offers competitive and appealing results.

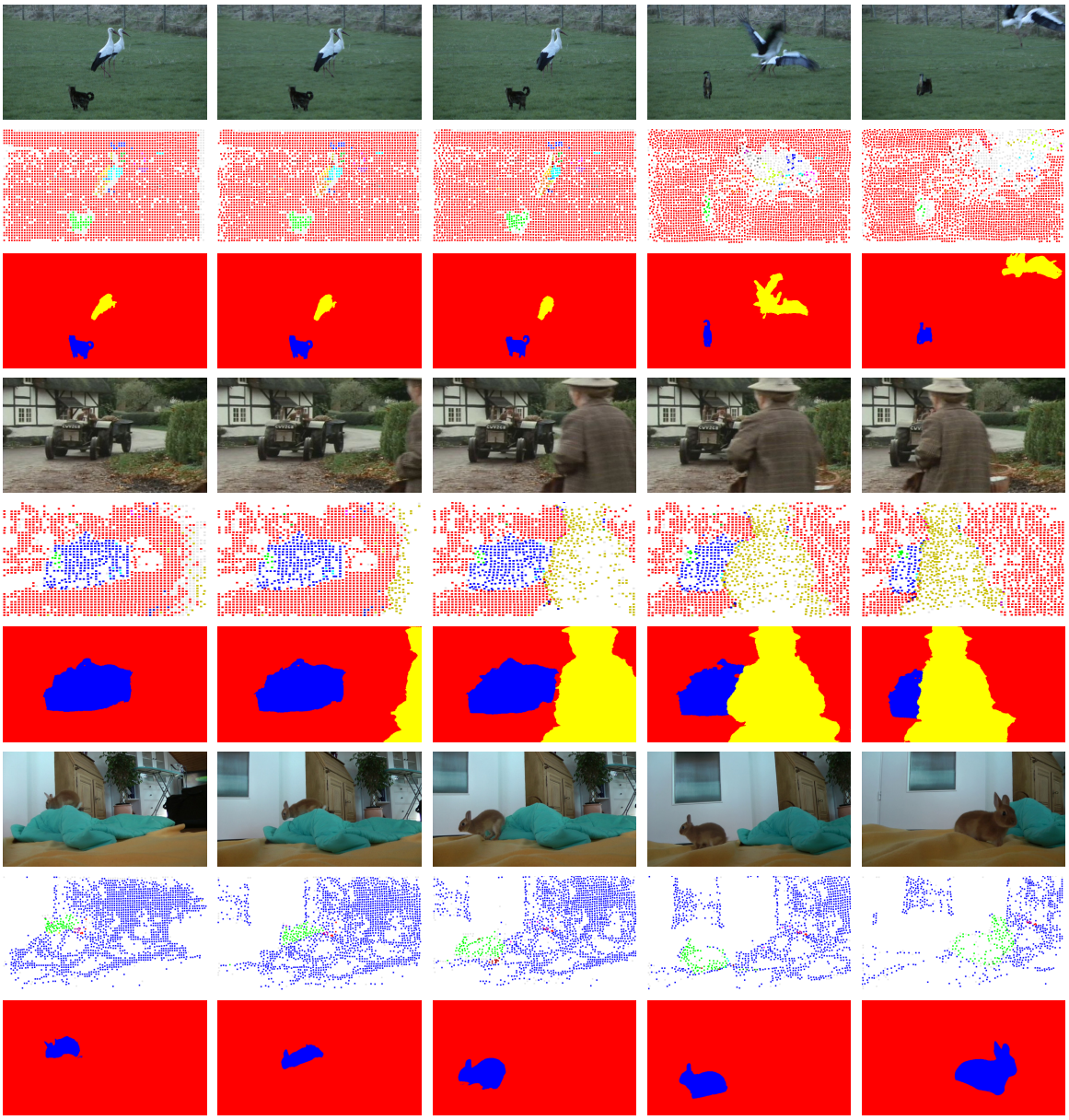


Fig. 4.5 Exemplary single- and multi-label motion segmentation results showing the image and its sparse results, as well as dense segmentation for five frames in three different sequences, generated using the proposed U-Net model. The images are from the FBMS59 [123] dataset. Segmentations with fine details are produced even when training labels are scarce; notice how scarce the labels are for "rabbit" images in the 8th row. White areas are parts without any label.

4.5.1 Relationship to the Self-Supervised Multiple Object Tracking Approach by Ho et al. [60]

In Ho et al. [60] we have proposed a self-supervised multiple object tracking approach based on visual features, and minimum cost lifted multicuts. The work has been conducted under the project lead of Kalun Ho, which is why we here only briefly discuss the relationship to the above-described approach. The method proposed in Ho et al. [60] is based on spatio-temporal cues for tracking the bounding box of multiple pedestrians. Similarly to the here presented motion segmentation setup, [60] proposes to employ a minimum cost multicut clustering to pre-group similar data points by spatio-temporal pattern. While the approach proposed in this chapter groups point trajectories, the work on multiple object tracking presented in [60] groups person detections. The pre-grouped detections are then used as a self-supervised training signal to train an auto-encoder to generate a suitable latent representation to provide robust appearance cues. Such cues can then be employed to re-identify persons over large temporal distances where there are no reliable spatio-temporal features. The goal is to learn a latent space representation to serve the match score for the same object appearing in different video frames [60]. The latent space of the same pedestrians is similar in the latent space. This is similar to our approach where we train the U-Net model by utilizing the noisy and sparse samples to generate the label for the unlabeled pixels on the sequence frames. By training the U-Net, the model learns to provide a similar latent representation for the pixels belonging to the same motion pattern.

Chapter 5

Motion Segmentation - Higher Order Minimum Cost Multicuts

In the previous chapter, we studied the motion segmentation task utilizing motion trajectories (see Section 2.4.1). To provide an informative segmentation, pairs of trajectories are used at a time to generate the costs on the edges of the multicut formulation. However, there are situations where comparing two trajectories can not resolve the complex motion patterns, such as out-of-plane rotation and scaling movements, i.e. movement of the object towards the camera. Then, higher order edges are required to address this problem by comparing more than two trajectories at a time. This chapter is based on a formulation of any-order minimum cost lifted multicuts, which allows partitioning an undirected graph with pairwise connectivity to minimize costs defined over any hyper-edges. As the proposed formulation is NP-hard and the branch-and-bound algorithm (as well as obtaining lower bounds) is too slow in practice, we propose an efficient local search algorithm for inference into resulting problems [101]. We demonstrate the versatility and effectiveness of our approach in several applications: 1) We formulate homography and motion estimation as a geometric model fitting problem where the task is to find groups of points that can be explained by the same geometrical transformation. 2) In motion segmentation our model allows going from modeling translational motion to Euclidean or affine transformations, which improves the segmentation quality in terms of the F-measure. This work is published in the Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 2022 [101] and substantially consolidates a previous paper by Keuper [87] that proposed the idea as well as an evaluation of FBMS59. The paper has shared first authors contributions with Evgeny Levinkov who implemented and evaluated the framework in the context of all geometric model fitting. Here, we omit this contribution on geometric model fitting entirely and solely show the theoretical concept from our published work [101] with a simplified reformulation of the model in Keuper [87], the

application to motion segmentation as in Keuper [87] as well as the consolidated evaluation additionally contributed in Levinkov et al. [101].

5.1 Introduction

Given the interesting properties of the minimum cost multicut (MP), Chopra and Rao [36], and its generalization, minimum cost lifted multicut (LMP), Keuper et al. [89], which are explained in Section 2.5. Kim et al. [92] proposed a higher-order multicut formulation that allows to model dependencies between more than two nodes. This chapter combines the LMP with higher order multicuts in one formulation. This generalization allows us to apply multicut formulation to motion segmentation problems, which require higher-order non-local costs and can have a variable number of objects in each problem.

On the downside, Bansal et al. [12] showed that solving the multicut problem is exactly NP-hard. This result extends to the above-mentioned multicut-based formulations. Although branch-and-bound, Andres et al. [3], algorithms, as well as Linear Programming (LP) relaxations like in Kim et al. [92], Kappes et al. [79], are feasible when applied to small problems, they do not easily scale, Levinkov et al. [102]. Instead, we propose a local search algorithm based on an efficient move-making algorithm, Keuper et al. [89]. This original heuristic by Keuper et al. [89] offers feasible solutions for the (second order) minimum cost lifted multicut problem. Here, we extend it to handle also higher-order terms and their combinations. Such heuristics do not guarantee the quality of solutions or computation time but work well in practice, Levinkov et al. [102], and provide feasible solutions at any time. Thanks to the affordable runtime of our proposed local search algorithm, we were able to apply higher order (lifted) multicuts to large problems, which we describe in detail below.

5.1.1 Motion Segmentation

The motion segmentation is illustrated in Section 2.4. In this chapter, we study motion segmentation by using motion trajectories (Section 2.4.1). The usage of two motion vectors is adequate to describe translation, rotation, and scaling transformations. Thus, for any three points, one can estimate how well their motion can be explained by one Euclidean transformation through residual errors. Costs that describe such motion differences are thus at least of order three. Affine motion differences can be estimated from four motion vectors, and to assign costs to differences in homographies, the minimum required order is five. Our model offers the flexibility to combine edges of varying order in one problem instance. The

motion segmentation benchmarks yield rather large problem sizes such that we are limited to models up to order three in this setting and prove the practicality of the results.

One additional adversity in motion segmentation is distinguishing between different objects with similar underlying motion patterns. Lifted Multicuts, Keuper et al. [89], have shown to resolve such ambiguities appropriately in the context of image segmentation. We show that higher-order graphs with third-order edges and their combination with lifted edges propose better motion segmentations and disambiguate complex motion patterns like similarly moving objects, scaling motions, and out-of-plane rotations of the objects.

Contributions. In summary, we show the practical benefit of using higher-order lifted minimum cost multicuts for motion segmentation. In contrast to previous approaches, our model allows the segmentation of noisy data into segments with respect to motion models beyond in-plane translation by combining second and third-order edges and allows for efficient optimization.

5.2 Related Work

Motion Segmentation. Our approach relies on the LMP formulation, (see Section 2.5.1), for hyper-graph decomposition. In contrast to spectral clustering, the multicut formulation does not suppose any balancing criterion. Moreover, we directly infer segmentations from the hyper-graph without projecting onto its primal graph. In contrast to Higher-Order Markov Random Field (MRF), the proposed approach allows higher-order edges to connect vertices globally, violating the Markov property. Further, MRFs and Conditional Random Fields (CRF)s aim to infer a node labeling with labels given a priori, while multicut approaches aim at figuring an edge labeling yielding an optimal number of segments.

We cast motion segmentation, (see Section 2.4), as a point trajectory, (see Section 2.4.1), grouping problem and treat it like a model fitting problem to generate segments based on different motion pattern. In a similar way, it has previously been addressed in Brox and Malik [25], Lezama et al. [103], Ochs and Brox [122], Li et al. [105], Shi et al. [137], Ochs et al. [123], Ji et al. [73], Keuper et al. [88]. From sparse motion segmentations, frame-wise dense segmentations can be computed by variational approaches like Ochs and Brox [121] or through learning, such as our proposed model in previous chapter [81].

Our approach directly estimates rigidly moving object parts in a single step, accounting for camera motion using third-order models, and does not depend on external object proposals, Levinkov et al. [101].

5.3 Higher-Order Lifted Multicut Problem

A decomposition of a graph $G = (V, E)$ can be represented by assigning to each vertex an identifier of a component it belongs to, i.e., a *vertex labeling*. The drawback of such an encoding is that a permutation of components' identifiers will result in a different vertex labeling while encoding the same decomposition. This ambiguity creates problems during optimization that are hard to deal with because the search space of feasible solutions can be factorially large. An alternative approach is to assign either 0 or 1 to each edge such that edges labeled 1 connect nodes only inside connected components, refer to Figure 2.8 (a) in Chapter 2.

Such 01-edge labeling, complying with constraints we define below, is called a *multicut* of a graph. Minimum cost multicut problem (MP), (refer to Section 2.5 for the definition of MP), allows to optimize for the 01-edge labeling, or in other words, to find an optimal decomposition of a graph. This is precisely the setting in the applications we consider in this chapter, as we do not know beforehand how many moving objects the data contains.

Below, we combine the lifted multicuts, (see Section 2.5.1), and higher-order multicuts in a joint formulation.

Definition. For a simple, connected graph $G = (V, E)$ and lifted edges F, such that $F \subseteq \binom{V}{2} \setminus E$, let $\mathscr{V} = \{U \mid U \in 2^V, |U| \geq 2\}$ denote the set of connected subsets of nodes in G. For a given cost function $c \colon \mathscr{V} \to \mathbb{R}$, written below is an instance of the higher-order minimum cost lifted multicut problem [101].

$$\min_{y \in \{0,1\}^{E \cup F}} \sum_{U \in \mathscr{V}} c_U \prod_{\{v,w\} \in \binom{U}{2} \cap \{E \cup F\}} y_{vw}\,, \tag{5.1}$$

with y subject to the following linear constraints

$$\forall C \in \text{cycles}(G), \forall e \in C\colon \quad (1 - y_e) \leq \sum_{e' \in C \setminus \{e\}} (1 - y_{e'}) \tag{5.2}$$

$$\forall f = \{v,w\} \in F, \forall P \in vw\text{-paths}(G)\colon \quad (1 - y_f) \leq \sum_{e \in P} (1 - y_e) \tag{5.3}$$

$$\forall f = \{v,w\} \in F, \forall T \in vw\text{-cuts}(G)\colon \quad y_f \leq \sum_{e \in T} y_e\,. \tag{5.4}$$

Cycle inequalities (5.2) ensure that the cut in graph G does not have holes. Path inequalities (5.3) guarantee that $\forall f = \{v,w\} \in F$, y_f can be assigned value 1, iff there exists a path P in graph G, that connects vertices v and w. Otherwise, the solver has two options: either create such a path or set $y_f = 0$. Cut inequalities (5.3) guarantee, that $\forall f = \{v,w\} \in F$, y_f can be assigned value 0, iff there exists a cut T in graph G, that separates vertices v and w. Otherwise, the solver has to either create such a cut or set $y_f = 1$.

Note that in our formulation (5.1), the set of decompositions is defined over a pairwise connected graph G. The costs, however, are determined over connected subsets $U \in 2^V$ of nodes of arbitrary cardinality larger than 1. Normally, only subsets of fixed cardinality k are used, e.g. $\mathscr{V} = \binom{V}{k}$. However, one can define a cost function over several cardinalities $K \subset \mathbb{N} \setminus \{1\}$. It is easy to see that in case $K = \{2\}$ we get the lifted multicut formulation from Keuper et al. [89]. Therefore, here we propose a strictly more general formulation.

Keuper et al. [89] showed the connection of optimization problem in Equation (5.1) with $\mathscr{V} = \binom{V}{2}$ to finding the most likely multicut in Bayesian sense: Let $p(y_{vw} = 1 \mid x_{vw})$ be a conditional probability estimate for two nodes $\{v,w\} \in \mathscr{V}$ to belong together given some features x_{vw}. If we set costs as $c_{vw} = \log \frac{1-p(y_{vw}=1|x_{vw})}{p(y_{vw}=1|x_{vw})}$, then minimizing Equation (5.1) is the same as performing Maximum A Posteriori (MAP) inference in the induced Bayesian network. The extension from 2nd-order sets to higher-order cases is straightforward. This allows us to interpret solutions of Equation (5.1) in terms of local probabilities.

If $p(\prod_{\{v,w\} \in \binom{U}{2} \cap \{E \cup F\}} y_{vw} = 1 \mid x_U)$ is greater than 0.5 for some $U \in \mathscr{V}$, then the corresponding cost c_U is less than 0. This means that all nodes in U are likely to belong together. We call such terms *attractive*. Conversely, if $p(\prod_{\{v,w\} \in \binom{U}{2} \cap \{E \cup F\}} y_{vw} = 1 \mid x_U)$ is less than 0.5, then the corresponding cost c_U is greater than 0 and acts as a penalty. We call such terms *repulsive*.

5.4 Local Search Algorithm

A branch-and-bound method in Gurobi [57] is implemented for the 3rd-order case by linearizing the objective in Equation (5.1), but a solution could not be obtained even for the smallest problem we consider in 12 hours. Toward scalable algorithms, Keuper et al. [89] define a generalization of Kernighan and Lin's primal local search algorithm for graph partitioning problems to the case of lifted multicut problem. We generalize their algorithm further to *lifted* multicut problems to include costs of arbitrary order.

Overview. The algorithm takes an instance of the higher-order lifted multicut problem and an initial decomposition of G and outputs a decomposition of G whose higher-order lifted

Algorithm 4: Kernighan-Lin Algorithm. The function UPDATE_BOUNDARY is given in Algorithm 5.

Data: weighted, undirected graph $G = (V, E)$, lifted edges F, cost function c, starting 01-edge labeling $y^0 \in \{0,1\}^{E \cup F}$
Result: 01-edge labeling y^{t-1}

```
1 t ← 1
2 while t < max_iter and y^t ≠ y^{t-1} do
3     foreach (A, B) ∈ adjacent_components(y^{t-1}) do
4         y^t ← update_boundary(G, F, c, A, B)
5     foreach A ∈ components(y^t) do
6         y^t ← update_boundary(G, F, c, A, ∅)
```

multicut has an objective value lower than or equal to that of the initial decomposition. As the original KLj-Algorithm, Keuper et al. [89], it always maintains, throughout its execution, a feasible decomposition of G. The pseudo-code is given in Algorithm 4. New components are introduced by updating a boundary of a component against an empty set $\emptyset$, as given by lines 5–6, exactly as in the 2nd-order version, Keuper et al. [89], refer to Algorithm 2 in Chapter 2.

Function UPDATE_BOUNDARY (refer to Algorithm 5) receives two components A and B and updates the cut between only them. It constructs a sequence M of elementary transformations of the components A and B greedily such that every consecutive move-operation increases the cumulative gain S maximally (or decreases it minimally). Therefore, the operation COMPUTE_GAINS computes, at the beginning of each execution of UPDATE_BOUNDARY, for every element $v \in A \cup B$ the difference in the objective function (gain) when v is moved from A to B or from B to A. These differences are updated as described in Algorithm 5, ll. 9-21. To escape local optima, we determine $i^* = \text{argmax}_i S_i$ such as to maximize the total gain of the *sequence of operations*. If the objective value can be decreased by executing either the first i^* elementary transformations or by joining the components A and B the optimal of these two operations is carried out. While components are defined with respect to the graph $G = (V, F)$, differences in objective value are computed with respect to the graph $G' = (V, E \cup F)$.

In Keuper et al. [89] as well as in our algorithm, all transformations of feasible solutions are local, resulting in changes of the objective value that are computed in linear time (in the size of the graph).

The combination of the locality of individual transformations and the non-locality of sequences of transformations has proven effective for diverse applications, Levinkov et al. [102]. As in KLj [89], the number of outer iterations of Algorithm 4 is not bounded by a

Algorithm 5: Function UPDATE_BOUNDARY greedily moves vertices from one component to the other.

Data: weighted, undirected graph $G = (V, E)$, lifted edges F, cost function c, a pair of partitions A and B
Result: 01-edge labeling y

1 $D^{A \cup B} \leftarrow$ compute_gains(G, F, c, A, B)
2 $\Omega \leftarrow$ find_boundary_nodes(V, E, A, B)
3 $\Delta_{join} \leftarrow$ compute_gain_from_joining(G, F, c, A, B)
4 $S_0 = 0$
5 $M = [\,]$ `// array to store moves`
6 **for** $i \leftarrow 1$ **to** $|\Omega|$ **do**
7 $\quad v^* \leftarrow \arg\max_{v \in \Omega} D^{A \cup B}$
$\quad$ `// w.l.o.g. let` $v^* \in A$
8 $\quad$ **foreach** $U \in \{U' \mid U' \in \mathcal{V}, v^* \in U', U' \subseteq A \cup B\}$ **do**
9 $\quad\quad U' \leftarrow U \setminus \{v^*\}$
10 $\quad\quad$ **if** $U' \subseteq A$ **then**
11 $\quad\quad\quad$ **foreach** $w \in U'$ **do**
12 $\quad\quad\quad\quad D_w \leftarrow D_w - c_U$
13 $\quad\quad$ **else if** $U' \subseteq B$ **then**
14 $\quad\quad\quad$ **foreach** $w \in U'$ **do**
15 $\quad\quad\quad\quad D_w \leftarrow D_w + c_U$
16 $\quad\quad$ **if** $|U' \cap A| = 1$ **then**
17 $\quad\quad\quad w \leftarrow U' \cap A$
18 $\quad\quad\quad D_w \leftarrow D_w - c_U$
19 $\quad\quad$ **if** $|U' \cap B| = 1$ **then**
20 $\quad\quad\quad w \leftarrow U' \cap B$
21 $\quad\quad\quad D_w \leftarrow D_w + c_U$
22 $\quad M$.push(v^*) `// move` v^* `from` A `to` B
23 $\quad S_i \leftarrow S_{i-1} + D_{v^*}$ `// cumulative gain`
24 $\quad \Omega \leftarrow$ update_boundary(v^*, E, A, B)
25 $i^* \leftarrow \text{argmax}_i S_i$ `// best number of moves`
26 **if** $\Delta_{join} > S_{i^*}$ **and** $\Delta_{join} > 0$ **then**
27 $\quad$ join_components(y, A, B)
28 undo_moves(y, A, B, i^*) `// undo moves after` i^*

polynomial, and we cannot give any guarantee for convergence. However, in practice, the algorithm converged in less than 50 iterations for the experiments described in Section 5.5.

Implementation Details. For efficiency, we pre-compute all the gains for vertices in $A \cup B$ (line 1), refer to Algorithm 5, and keep track of the vertices that currently lie on the boundary

Ω between A and B (lines 2 and 21); this dramatically improves the runtime for sparse graphs. We iteratively pick a vertex v^* with the largest gain (line 7), which can also be negative. Then, we update gains of all other vertices in A and B, that are in subsets U that contain v^* (lines 8–18). Note that in the case of 2nd-order costs, updates as given in lines 10–18 specialize to the corresponding updates in Keuper et al. [89]. In the end, we find which first i^* moves produce the most significant decrease (note, i^* can also be 0), and either merge A and B together (lines 23–24) or undo the moves after i^* (line 25).

5.4.1 Motion Segmentation

Point Trajectories. Tracking the single object point in the image plane produces a spatio-temporal curve, named point trajectory. Set of point trajectories, detailed in Section 2.4.1, provide the basis for many motion segmentation methods such as Brox and Malik [25], Fragkiadaki et al. [49, 51], Ochs et al. [123], Keuper et al. [88]. We use the precomputed optical flow from Brox and Malik [26] to allow for a direct comparison to prior work and to generate dense long-term point trajectories using the method from Brox and Malik [25]. Initially, n point trajectories p_i are produced by [25] for a video of length N. Actually, n depends on the desired sampling rate and most trajectories are shorter than the N, due to occlusions and mistakes in the optical flow estimation. Some trajectories are started after the first frame to ensure even point sampling over the sequence, Keuper [87].

5.4.2 Higher-Order Motion Models

For practical reasons, we restrict ourselves to edge potentials of orders two and three; however it is not sufficient to accurately describe object motion in a 3D environment recorded with a possibly moving camera. This lets us to measure the difference of point motions according to Euclidean motion models, i.e. from the group of transformations describing translation, rotation, and scaling in the 2D plane. Excluding the reflections, this is a subset of group of similarity transformations in the 2D plane, Keuper [87].

Furthermore, we argue that in any case, the easiest model that can explain the motion of a set of points with a single transformation should be used. We can assume the two points belong to the same object without looking at further points around them, if they move according to the same translational motion model. Looking at more complex motion models adds information, only if their motion is different according to a purely translational model. This leads us to a *motion-adaptive graph construction* strategy [87].

Algorithm 6: Motion Adaptive Graph Construction [87].

Data: set of point trajectories V with $p_k \in V$ with $k \in \{1 \dots n\}$
Result: weighted undirected higher-order graph $G = (V, E)$, cost vector c

$G \leftarrow (V, E = \emptyset)$
$c = [\,]$
foreach $(u, v) \in \binom{V}{2}$ **do**
    $c_{uv} \leftarrow$ compute_translational_motion_cost(p_u, p_v)
    $E \leftarrow E \cup (u, v)$ `// add edge`
    if $c \leq 0$ **then**
        c.push(c_{uv})
    else
        foreach $w \in V \setminus \{u, v\}$ **do**
            $c_{uvw} \leftarrow$ compute_HO_motion_cost(p_u, p_v, p_w)
            $E \leftarrow E \cup (u, w) \cup (v, w)$ `// add edge`
            c.push(c_{uvw})

Motion-Adaptive Graph Construction. We present the higher-order graph G construction from the pairwise costs computed from motion differences. The algorithm is described in Algorithm 6. We compute cost of belonging to the same translational motion model for any pair of trajectories. If this cost produce repulsive signal; i.e., positive, we look at all further points to compute for every three-tuple the cost of belonging to the same motion model for scaling, translation, and rotation. Afterwards, the respective third-order edges are inserted as well as their costs.

With this strategy, second and third-order potentials are integrated without losing model capacity. Moreover, compared to generating the full graph with higher-order potentials, it yields a significant space reduction in practice [87].

Lifted Graph Construction. For every trajectory the set of its 12 spatially nearest neighbors $\mathcal{N}$ are computed, to construct a higher-order lifted graph $G' = (V, E \cup F)$. Algorithm 6 is used to compute the edge set E of edges between direct neighbors in G'. It contains exactly all pairwise edges $e_{ij} \in E$ for which at least one of the following conditions holds: (1) $p_i \in \mathcal{N}(p_j)$, (2) $p_j \in \mathcal{N}(p_i)$ (3) the maximum spatial distance between the trajectories p_i and p_j is below 40 pixels [87].

Second-Order Costs. The pairwise differences in point trajectories are used to compute the second-order costs. Such differences are calculated only for trajectories that have at least two frames in common. We compute such differences based on color, motion, and spatial

distance cues, since it has proven successful in the work of Keuper et al. [88]. The motion distance in Equation (5.5), is exactly what we used in the last chapter in Equation (4.1). As suggested by Ochs et al. [123], we define the pairwise motion difference of two trajectories at time t as

$$d_t^{\text{motion}}(p_i, p_j) = \frac{\|\partial_t p_i - \partial_t p_j\|}{\sigma_t}. \tag{5.5}$$

Here, $\partial_t p_i$ and $\partial_t p_j$ are the partial derivatives of p_i and p_j with respect to the time dimension and σ_t, which is defined in Ochs et al. [123], is the variation of the optical flow. The motion distance of two trajectories is defined by the maximum over time [87],

$$d^{\text{motion}}(p_i, p_j) = \max_t d_t^{\text{motion}}(p_i, p_j). \tag{5.6}$$

color and spatial distances d^{color} and d^{spatial} are computed as average distances over the common lifetime of two trajectories which is proposed in Keuper et al. [88]. The costs are computed by the non-linear combination of these three cues

$$\begin{aligned} c_{ij} = \max\big(&\bar{\theta}_0 + \theta_1 d^{\text{motion}}(p_i, p_j) \\ &+ \theta_2 d^{\text{spatial}}(p_i, p_j) \\ &+ \theta_3 d^{\text{color}}(p_i, p_j), \\ &\theta_0 + \theta_1 d^{\text{motion}}(p_i, p_j)\big) \end{aligned} \tag{5.7}$$

with weights and intercept values θ as proposed in Keuper et al. [88][1].

Third-Order Costs. As proposed in Ochs and Brox [122], we compute third-order motion differences. For any two trajectories p_i and p_j, we estimate the Euclidean motion model $\mathcal{T}_{ij}(t)$, which consists of translation $v := (v_1, v_2)^\top$, rotation R_α, and scaling s as

$$\begin{aligned} \alpha &= \arccos\left(\frac{(p_i(t') - p_j(t'))^\top (p_i(t) - p_j(t))}{\|p_i(t') - p_j(t')\| \cdot \|p_i(t) - p_j(t)\|}\right) \\ s &= \frac{\|p_i(t') - p_j(t')\|}{\|p_i(t) - p_j(t)\|} \\ v &= \frac{1}{2}\left(p_i(t') + p_j(t') - sR_\alpha(p_i(t) + p_j(t))\right), \end{aligned} \tag{5.8}$$

[1] Specifically, $\bar{\theta}_0 = 6$, $\theta_0 = 2$, $\theta_1 = \theta_3 = -0.02$ and $\theta_2 = -4$.

where t and t' denote the first and last point in time where both trajectories co-exist, respectively. The distance to any third trajectory p_k existing from t to t' can then be measured by $d_{ij}^t(p_k) = \|\mathscr{T}_{ij}(t)p_k(t) - p_k(t')\|$ [87]. For numerical reasons, $d_{ij}^t(p_k)$ is normalized by

$$\gamma_{ij}^t = \frac{1}{\sigma_t}\left(\frac{1}{2}\left(\frac{\|p_i(t)-p_j(t)\|}{\|p_i(t)-p_k(t)\|} + \frac{\|p_i(t)-p_j(t)\|}{\|p_j(t)-p_k(t)\|}\right)\right)^{\frac{1}{4}}, \tag{5.9}$$

where σ_t represents the optical flow variation as in Equation (5.5). To render distances symmetric, Ochs and Brox [122] offer to utilize the maximum,

$$d_{\max}^t(i,j,k) = \max(\gamma_{ij}^t d_{ij}^t(p_k), \gamma_{ik}^t d_{ik}^t(p_j), \gamma_{jk}^t d_{jk}^t(p_i)), \tag{5.10}$$

which produces an over-estimation of the true distance. It can lead to problems in the multicut framework; however, this is unproblematic in a spectral clustering scenario, where distances define positive point affinities. Over-estimated distances generate under-estimated join probabilities and consecutively leads to switching the sign of the cost function towards repulsive terms. We compute both $d_{\max}^t(i,j,k)$ and, $d_{\min}^t(i,j,k)$, to avoid this effect. For both, we compute the maximum motion distance over the common lifetime of p_i, p_j and p_k as $d_{\max}(i,j,k) = \max_t d_{\max}^t(i,j,k)$ and $d_{\min}(i,j,k) = \max_t d_{\min}^t(i,j,k)$. We evaluate the costs $c(d_{\max}(i,j,k))$ and $c(d_{\min}(i,j,k))$ for both distances as $c(d) = \theta_0 + \theta_1 d$ and compute the final edge costs [87]

$$c_{ijk} = \begin{cases} c(d_{\min}(i,j,k)) & \text{if} \quad c(d_{\max}(i,j,k)) > 0 \\ c(d_{\max}(i,j,k)) & \text{if} \quad c(d_{\min}(i,j,k)) < 0 \\ 0 & \text{otherwise.} \end{cases} \tag{5.11}$$

Therefore, we make sure not to set any costs for edges whose underlying motion is controversial. We set $\theta_0 = 1$ and $\theta_1 = -0.08$ manually.

Implementation Details. In practice, pairwise edges e_{ij} in G and G' are inserted only if the spatial distance between p_i and p_j is below 100 pixels even for lifted edges in F. This is in analogy to Keuper et al. [88]. Due to the fact that for nearby points, the approximation of the true motion by a simplified model is usually better than for points at a considerable distance. Furthermore, since the number of pairwise edges increases quadratically with the maximal spatial distance, this heuristic decreases the computational load significantly. We introduce an edge sampling strategy for third-order edges for the same reason. We compute the maximum pairwise distance d for every triplet of points. We randomly sample $\frac{100}{d^2}\%$

Fig. 5.1 Samples of our Lifted motion-adaptive order (AOMC) segmentations densified by Ochs and Brox [121] are shown. Our segmentations show little over-segmentation even for articulated motion.

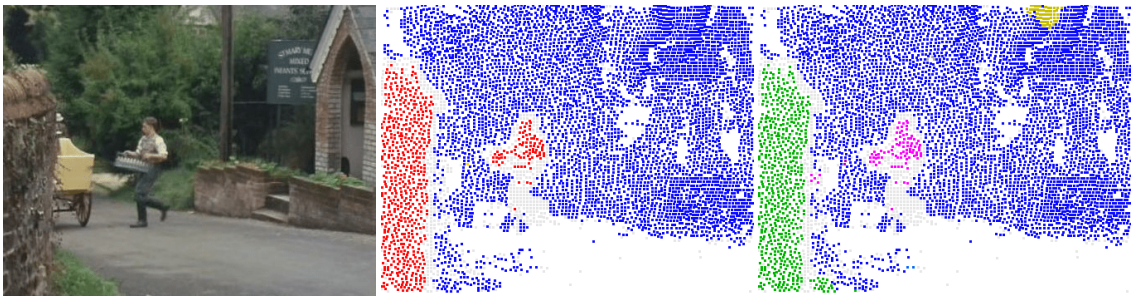


Fig. 5.2 The person and the wall are assigned to the same cluster with the non-lifted multicut approach from Keuper et al. [88] because of the camera motion. The Lifted AOMC allows for correct segmentation.

from all triplets with $20 < d < 300$, while we insert all edges e_{ijk} with $d \leq 20$. This prevents a too strong imbalance of long-range edges over short-range edges [87].

5.5 Experiments

5.5.1 Motion Segmentation

We apply the proposed higher-order lifted multicut model on the motion segmentation benchmark FBMS59 [123] (see Section 2.1). The results reported in tables 5.1 and 5.2 and in Figure 5.8 have been previously shown in Keuper [87]. In figures 5.1, 5.2, 5.3, 5.4, 5.5, and 5.6 the images in different frames from the same sequences as in Keuper [87] are shown.

The data set has been split into subsets of 29 train and 30 test sequences. Yet, the model parameters used in the evaluation have not been optimized on the training set but rather heuristically on single training sequences. First, by the time of submission, neither of the state-of-the-art methods like Ochs and Brox [122], Ochs et al. [123], Keuper et al. [88]

was fully training-based, and second, the training set of FBMS consists of only 29 sparsely annotated sequences and is therefore relatively small. To avoid confusion, we denote the original training split by *Set A* and the original test split by *Set B*.

FBMS59 [123] provides manual annotations for all moving objects in the videos for every 20th frame and *ground-truth definition* files that down weight annotated segments in some scenes. Thus, objects in some sequences that contain severe camera motion, for example, a mistakenly segmented wall in Figure 5.2, attain a lower weight in the evaluation. All objects that move in at least one frame are segmented in all annotated frames. A second set of ground-truth annotations at a similar level of sparsity has later been provided in Bideau and Learned-Miller [20] to evaluate a slightly different motion segmentation paradigm. In Bideau et al. [23], Bideau and Learned-Miller [20], it was argued that all freely moving objects in 3D space but nothing more should be segmented per frame. Specifically, this means that objects moving only in a few frames are only to be segmented in these frames. Scene geometry and camera motion yielding apparent motion in the image plane, such as the wall in the example of Figure 5.2, are not considered. Our work addresses the task originally annotated by Ochs et al. [123]. It aims to segment all objects that move in at least one frame of a sequence and does not provide full 3D motion segmentations, as we limit ourselves to third-order motion terms. Yet, we find it interesting to consider both sets of annotations during evaluation to allow for a comparison to the respective competing methods. Note that also the evaluation metrics differ slightly. Both Ochs et al. [123] and Bideau and Learned-Miller [20] measure precision, recall and F-measure, where Ochs et al. [123] evaluate precision and recall over all frames and compute the F-measure in the end, while Bideau and Learned-Miller [20] evaluate the F-measure per frame and report the mean value. In addition, Ochs et al. [123] report the number of objects O that are segmented with an F-measure above 0.75. Instead, Bideau et al. [23] propose the ΔObj metric, which measures the average absolute difference between the number of ground-truth objects in each frame and the number of segmented objects in this frame. While O should be large, ΔObj should be small.

For our evaluation, we employ the annotations provided in Ochs et al. [123] when considering the metric proposed therein. We use the annotations from Bideau and Learned-Miller [20] when evaluating using the metric proposed in Bideau et al. [23] to allow for direct comparison to their work. We start with an evaluation using annotations and metrics from Ochs et al. [123].

Evaluation. First, we evaluate a purely higher-order non-lifted version of our model to assess the capacity of our model components. All pairwise costs are removed in this model, and all edges are connectivity-defining. We compare this simple model to the purely motion-

Table 5.1 Segmentation results on the FBMS59 dataset on Set A (**top**) and Set B (**bottom**). We report **P**: average precision in %, **R**: average recall in %, **F**: F-measure in % and **O**: extracted objects with F $\geq$ 75%. All results are computed for sparse trajectory sampling at an 8-pixel distance. Our result **HO MC** is computed on the non-lifted purely higher-order model to allow for a direct comparison to the listed competing methods.

Set A (29 sequences)	P [%, ↑]	R [%, ↑]	F [%, ↑]	O [↑]
SC [123]	85.10	62.40	72.0	17/65
Higher-Order SC [122]	81.55	59.33	68.68	16/65
MC [88]	84.94	71.22	77.48	23/65
HO MC (ours)	83.20	74.34	**78.52**	**29/65**
Set B (30 sequences)	P [%, ↑]	R [%, ↑]	F [%, ↑]	O [↑]
SC [123]	79.61	60.91	69.02	24/69
Higher-Order SC [122]	82.11	64.67	72.35	**27/69**
MC [88]	82.87	69.89	**75.83**	**27/69**
HO MC (ours)	82.92	68.82	75.22	**27/69**

based version of Keuper et al. [88] and Ochs et al. [123], Ochs and Brox [122]. The affinities in Ochs and Brox [122] are defined most similarly to our proposed higher-order costs, while Ochs et al. [123] and Keuper et al. [88] only consider translational motion. While Ochs et al. [123] and Ochs and Brox [122] follow a spectral clustering approach, as the proposed approach, Keuper et al. [88] formulate a multicut problem. The results are given in Table 5.1 in terms of precision, recall, F-measure, and the number of extracted objects. From Table 5.1, it is observable that our higher-order lifted multicut model outperforms the higher-order spectral clustering method from Ochs and Brox [122] by about 10% on Set A and 3.5% on Set B. The imbalance is remarkable, while there is a clear improvement on both sets. A similarly remarkable imbalance is observable when comparing the performance between the two spectral clustering methods Ochs et al. [123] and Ochs and Brox [122]. The higher-order model [122] leads to a lower F-measure on Set A compared to [123]. Yet it outperforms Ochs et al. [123] by about 3% on Set B. This shows an indication that the motion statistics in both splits are significantly different. We can observe an improvement in Set A when we compare our higher-order model to the pairwise minimum cost multicut model from Keuper et al. [88]. Both models perform almost equally on Set B.

We show the evaluation of our higher-order multicut model with the motion-adaptive order, denoted AOMC (refer to Algorithm 6) in Table 5.2. This model has access to similar pairwise cues as the color and motion-based version from Keuper et al. [88], denoted by MCe. It also has access to the higher-order motion cues from Equation (5.11), Keuper [87].

Table 5.2 Segmentation Results on FBMS59 on Set A (top) and Set B (bottom) are provided. We report **P**: average precision, **R**: average recall, **F**: F-measure and **O**: extracted objects with F $\geq$ 75%. All results are computed for sparse trajectory sampling at an 8-pixel distance. The proposed approach **Lifted AOMC** performs best.

Set A (29 sequences)	P [%, ↑]	R [%, ↑]	F [%, ↑]	O [↑]
MCe [88]	86.73	73.08	79.32	31/65
HOPMC	87.66	74.15	80.34	31/65
AOMC	82.29	76.17	79.11	32/65
Lifted HOPMC	87.07	70.84	78.12	28/65
Lifted AOMC	86.20	78.35	**82.08**	**34/65**
Set B (30 sequences)	P [%, ↑]	R [%, ↑]	F [%, ↑]	O [↑]
MCe [88]	87.88	67.7	76.48	25/69
HOPMC	85.00	67.75	75.40	25/69
AOMC	84.48	73.08	78.37	**27/69**
Lifted HOPMC	87.07	70.84	78.12	28/69
Lifted AOMC	87.82	71.45	**78.79**	24/69

We also generate graphs that simply contain all pairwise costs c_{ij} and all third-order edges with costs c_{ijk} without any adaptation with respect to the costs as a sanity check for the motion-adaptive graph construction. This additive model is denoted by HOPMC (higher-order + pairwise multicut). All three approaches produce similar results on Set A, whereas the proposed AOMC shows especially good performance on the test set with about 2% improvement over MCe, proposed by Keuper et al. [88], in F-measure [87].

A slightly further improvement is acquired on Set B by lifted versions of both types of problems (HOPMC and AOMC). However, on Set A, the segmentation quality of Lifted HOPMC is below the one of MCe by around 1%. In contrast, all competing methods and baselines are outperformed by the proposed Lifted AOMC.

In Table 5.3, we evaluate the effect of the quality of the point trajectories when they are computed from different models for optical flow estimation like Brox and Malik [26], Ilg et al. [65, 66]. The proposed approach, Lifted AOMC, consistently outperforms the pairwise MCe [88] on comparable optical flow estimations. The overall differences are small; however, the most recent FlowNet [66] performs best.

In Figure 5.1, several examples of pixel-segmentations computed from our sparse segmentation using Ochs and Brox [121] are shown. The densified segmentation results look reasonable. In the bear example, still the over-segmented results appear due to the articulated motion of the leg. One of the horses is missed in the *horses05* sequence. However, the small objects such as the phone in the *marple13* sequence can be correctly segmented. Such densi-

Table 5.3 Segmentation results are provided for the proposed model Lifted AOMC on the FBMS59 dataset on Set A (**top**) and Set B (**bottom**) for different optical flow methods. We report **P**: average precision, **R**: average recall, **F**: F-measure and **O**: extracted objects with $F \geq 75\%$. All results are computed for sparse trajectory sampling at an 8-pixel distance.

Set A	Flow	P [%, ↑]	R [%, ↑]	F [%, ↑]	O [↑]
MCe [88]	LDOF [26]	86.73	73.08	79.32	31/65
Lifted AOMC		86.20	78.35	82.08	34/65
MCe [88]	FlowNet [65]	89.63	73.38	80.69	29/65
Lifted AOMC		88.22	77.34	82.42	33/65
MCe [88]	FlowNet [66]	89.77	75.78	82.19	34/65
Lifted AOMC		89.19	79.35	83.98	38/65
Set B	Flow	P [%, ↑]	R [%, ↑]	F [%, ↑]	O [↑]
MCe [88]	LDOF [26]	87.88	67.7	76.48	25/69
Lifted AOMC		87.82	71.45	78.79	24/69
MCe [88]	FlowNet [65]	86.73	68.77	76.71	26/69
Lifted AOMC		86.89	69.81	77.42	24/69
MCe [88]	FlowNet [66]	84.59	7019	76.72	27/69
Lifted AOMC		88.39	72.12	79.43	26/69

fied segmentations, computed from sparse results using FlowNet [66], can be evaluated by the metrics from Bideau et al. [23] with their matching annotations from Bideau and Learned-Miller [20]. Table 5.4 shows our results in the setting by Bideau et al. [23], i.e. considering a frame-wise evaluation on all freely moving 3D objects. While the F-measure of our model is similar to the one reached by Keuper et al. [88], the ΔObj metric is significantly improved, i.e. lower, and almost on par with the results from Bideau et al. [23], which are dedicated to this setting. Evaluating binarized segmentations (Table 5.4) allows for a comparison to learning based encoder-decoder models such as Tokmakov et al. [149]. Without learning any prior on object saliency, our F-measure for binary segmentation on FBMS59 is 76.16% and thus slightly better than the learned model from plain motion cues in Tokmakov et al. [149] with 74.79%, but inferior to Tokmakov et al. [150], who learn an additional appearance stream and reach 86.96% [23].

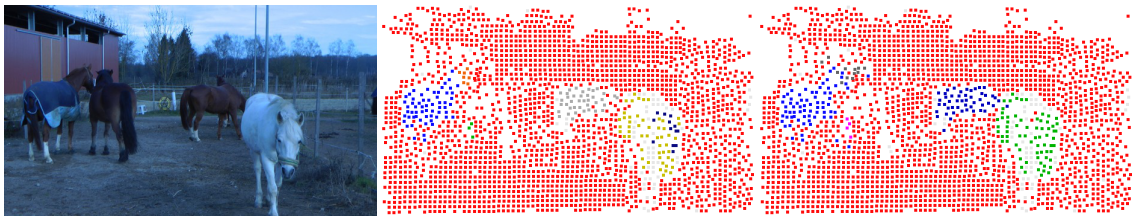
In the following, we discuss several example segmentations in detail [87].

Figure 5.3 shows the trajectory segmentation quality under scaling. In the *horses05* sequence, the motion of the white horse towards the camera shows the scaling. This creates over-segmentation in the competing method MCe [88], which can not solve higher-order motion models. The segmentation can be improved with the proposed Lifted AOMC.

Both figures 5.4 and 5.2 show examples where the same label is assigned to distinct objects that move similarly. In the *cars2* sequence in Figure 5.4, this effect is due to similar

Table 5.4 Results for densified segmentations on FBMS59 using annotations and metrics as in Bideau et al. [23] for freely moving 3D objects. For ΔObj, lower is better. Results for Keuper et al. [88], Taylor et al. [148], Tokmakov et al. [149], Tokmakov et al. [150] and Bideau et al. [23] are taken from Bideau et al. [23].

		P [%, ↑]	R [%, ↑]	F [%, ↑]	ΔObj [↓]
	MCe [88]	74.64	62.03	63.59	7.7
	Taylor at al. [148]	72.69	54.36	56.32	11.7
	Bideau et al. [23]	74.23	63.07	64.97	4
	Lifted AOMC	73.29	60.26	63.58	4.41
binary	Tokmakov at al. [149]	87.29	72.19	74.79	-
binary	Tokmakov et al. [150]	92.40	85.07	86.96	-
binary	Lifted AOMC	79.98	76.97	76.16	-



MCe [88] ours

Fig. 5.3 The scaling motion of the white horse moving towards the camera causes over-segmentation with a simple motion model from Keuper et al. [88]. With the proposed Lifted AOMC, this can be avoided.

real-world object motion, whereas, in the *marple10* sequence, the effect is caused because of the camera motion and the scene geometry. In both scenarios, the formulation of the Lifted AOMC problem lets us tell the distinct objects apart. However, in the *marple10* sequence (Figure 5.2), we can see a spurious segment in the background, which is probably caused by non-exact flow.

In Figure 5.5, an example of the *goats01* sequence is shown. Here, the pairwise method from Keuper et al. [88] causes the head and body of the goat in front are segmented into distinct components, because of the obvious articulated motion. Although our proposed third-order model can not explicitly handle articulation, the over-segmentation can be fixed in this case.

A failure case of the proposed method is shown in Figure 5.6. Due to the dominant camera motion in a scene with complex geometry, the Euclidean motion model particularly badly fits. Therefore, our model leads to the segmentation of the scene into its depth layers and, consequently, to strong over-segmentation [87].

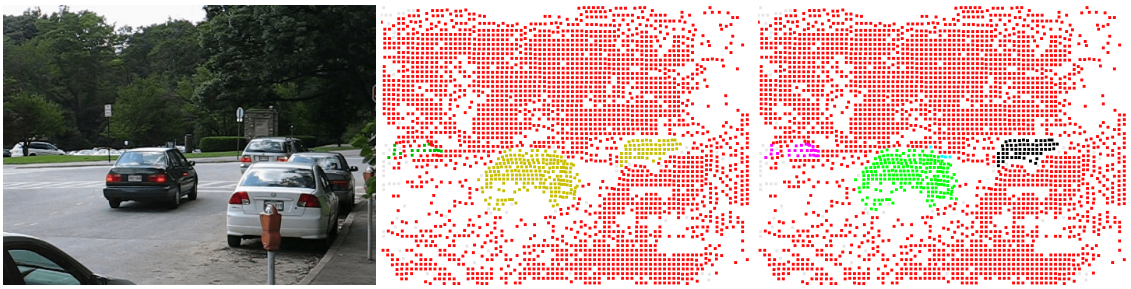


Fig. 5.4 The two cars in the front move in the same direction. This leads to the same cluster assignment with the non-lifted multicut approach [88]. The Lifted AOMC can assign the distinct motion labels to the different cars.

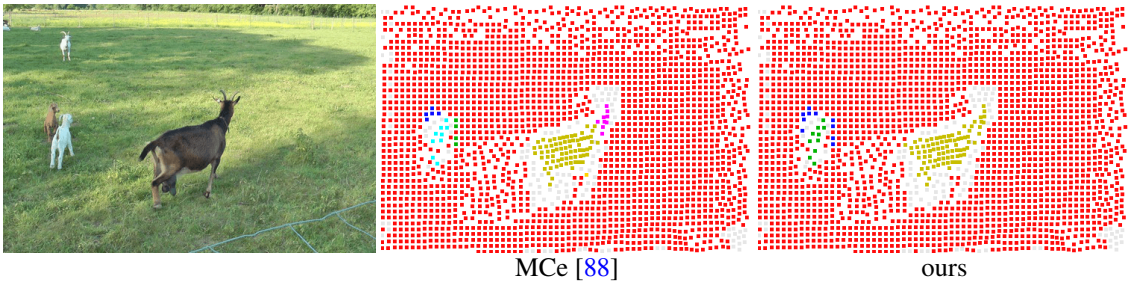
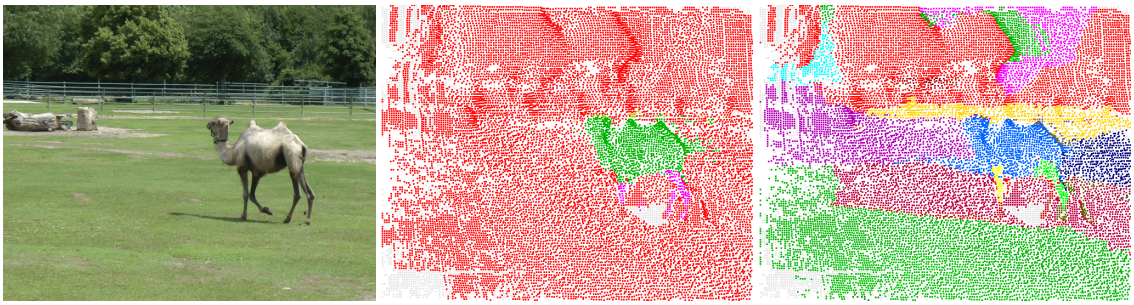


Fig. 5.5 The articulated motion leads to over-segmentation in [88]. Showing that the Lifted AOMC performs better.

Next, we evaluate our approach on two additional datasets widely considered for motion segmentation, the DAVIS_{2016} dataset [128] and the VSB100 dataset [55, 144]. Both have originally been designed for different purposes. DAVIS_{2016} is a dataset, (see Section 2.1 for the information about the datasets), for binary video object segmentation which has been used to learn appearance and motion patterns of salient objects in videos, e.g. in Tokmakov et al. [150]. While the sequences may contain several moving objects, the task is to track the segmentation of the dominant object throughout the sequence. Complementary to this, the VSB100 dataset is originally proposed as a video segmentation dataset where the task is to mimic human boundary level annotations, i.e., the segments do not necessarily have a notion of objectness. This general purpose multi-label video segmentation dataset has a motion subtask which can be used to evaluate motion segmentation approaches before, e.g. as in Keuper et al. [88]. Table 5.5 shows our results on the DAVIS_{2016} dataset in terms of the Jaccard index (J) and the F-measure (F), which measures the boundary fidelity of the segmentation. We compare the results by Keuper et al. [88] which is, like ours, a method for multi-label motion segmentation. The proposed approach improves significantly over those results. Yet, note that dedicated video object segmentation approaches recently proposed



MCe [88] ours

Fig. 5.6 The failure cases are shown. With the proposed method, the dominant camera motion causes strong over-segmentation. The proposed third-order model can not appropriately model the motion.

Table 5.5 Evaluation on DAVIS_{2016} is provided. The more complex motion model in Lifted AOMC is beneficial for this dataset of binary object segmentation. Results marked with * are taken from Tokmakov et al. [149].

	MCe [88]			Lifted AOCM		
	train	val	trainval	train	val	trainval
J mean [↑]	53.4	55.2*/54.59	53.9	**62.7**	**57.79**	**60.74**
J recall [↑]	59.5	57.5*/58.41	59.04	**74.18**	**64.72**	**70.40**
J decay [↓]	**-1.57**	**2.2***/ 4	**0.66**	3.8	3.75	3.78
F mean [↑]	51.86	55.2*/52.35	52.05	**61.55**	**57.55**	**59.94**
F recall [↑]	56.39	61.0*/54.60	55.67	**72.36**	**67.63**	**70.47**
F decay [↓]	**2.5**	**3.4*** / 4.25	**3.23**	7.05	5.39	6.38

in Yang et al. [165, 162] yield higher numbers on the DAVIS benchmark with a mean Jaccard index of up to 80% on the validation set.

The evaluation of our model on the motion subtask of VSB100 is given in Figure 5.7 in terms of boundary precision and recall (BPR) and the region metric volume precision and recall (VPR). It can be seen that the proposed higher-order model with adaptive edge order outperforms the previous models on this task. As expected, the differences in the BPR are rather small while they are more significant in VPR. The dashed lines indicate the results of our model using FlowNet [66] to compute optical flow, while the solid lines are based on Brox and Malik [26] to ensure fair comparison to Keuper et al. [88] and Ochs et al. [123]. The improved optical flow has a slightly larger impact on the BPR values, indicating that the issues addressed by more robust optical flow estimation and the issues addressed by our more complex motion model are complementary.

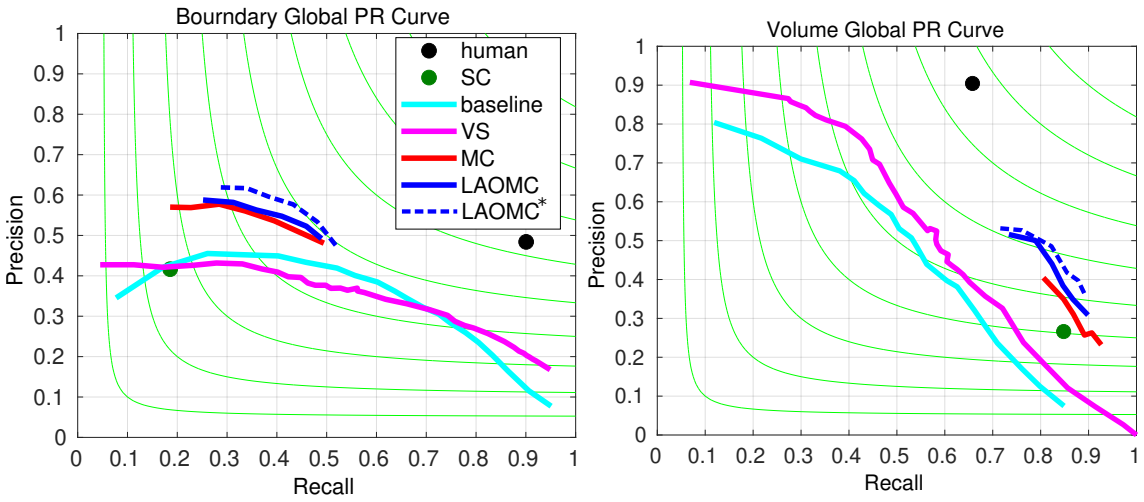


Fig. 5.7 Evaluation on the motion subtask of the VSB100 dataset [55, 144] is provided. We compare our results to SC, Ochs et al. [123], the video segmentation approach VS, Galasso et al. [54], the superpixel tracking baseline from Galasso et al. [55], and the multicut models with pairwise terms MCe, Keuper et al. [88]. The proposed lifted adaptive order model (LAOMC) outperforms the pairwise terms consistently. LAOMC* shows results based on FlowNet [66], while LAOMC is computed on flows from Brox and Malik [26] for fair comparison to Ochs et al. [123].

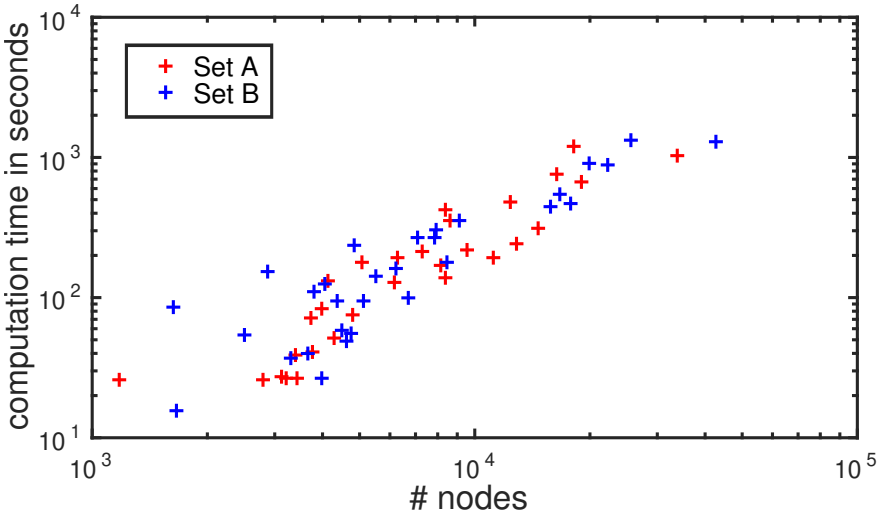


Fig. 5.8 Computation times in the log-scale of the problem instances from Set A and B of FBMS59 with respect to the number of point trajectories are provided (following Keuper [87]).

Scalability Analysis on FBMS59. Last, we evaluate our proposed heuristic in terms of computation times for the higher-order minimum cost lifted multicut problems (compare Algorithm 4), following Keuper [87]. Figure 5.8 shows the computation times of our full

pipeline on FBMS59 with respect to the number of point trajectories. The runtime distribution represents linear runtime behavior and shows that for most instances heuristic solutions can be generated in a few minutes. Yet, the number of large problem instances is very few to make any claim [87].

5.6 Conclusion

We presented a multicut-based approach that can be applied to computer vision tasks such as motion segmentation. To do so, we proposed a pseudo-boolean formulation that allows defining costs on subsets of vertices of arbitrary cardinality and includes lifted edges. In motion segmentation, higher than simple pair-wise costs allow modeling object motion more precisely (Euclidean instead of in-plane translational motion). Since the emerging higher-order multicut problem is NP-hard to solve exactly, we proposed an efficient local search algorithm for inference. Our approach yields either competitive or state-of-the-art results, is highly flexible and is easy to apply.

Chapter 6

Minimum Cost Multicuts – Efficient Solvers

In the last chapters, we studied the application of the video object and motion segmentation using the minimum cost *lifted* multicut (**LMP**) framework. This framework is further used in the multiple-person tracking method proposed by Ho et al. [60]. Due to the NP-hard nature of the problem, the available solvers provide the results at a high computational cost. This hinders the generation of the results on the larger problem instances. While the currently available solvers for this problem provide high-quality solutions, they can have long computation times for more difficult problem instances. Here, we propose two variants of a heuristic solver (primal feasible heuristic), which greedily generate solutions within a bounded amount of time. Evaluations of image and mesh segmentation benchmarks show the high quality of these solutions. This work is published in the Asian Conference on Computer Vision (ACCV), 2018 [82].

6.1 Introduction

This chapter tackles the fast, heuristic optimization of a graph decomposition problem, the LMP (for the definition of the LMP refer to Section 2.5.1). More specifically, we propose a modification of a very simple primal feasible heuristic, the greedy additive edge contraction (GAEC) algorithm from Keuper et al. [89] (refer to Algorithm 1 in Chapter 2). The proposed algorithm produces results that are close to the ones generated by KLj [89] with a computation time similar to the one from GAEC [89] and a guaranteed worst-case complexity. We evaluate the proposed heuristic on instances of (1) image segmentation problems on the BSDS500 dataset from Arbeláez et al. [5], (2) mesh segmentation problems on the Princeton Shape

Segmentation benchmark from Chen et al. [32], and (3), the challenging 3D segmentation instance from the ISBI 2012 challenge on segmentation of neuronal structures from electron microscopy images used in Beier et al. [15, 18] (see Section 2.1). In all scenarios, the proposed algorithm can generate high-quality results at a significantly reduced computation time.

6.2 Related Work

There are many solvers proposed for the LMP (Section 2.5.2). The two solvers, KLj from Keuper et al. [89] and Fusion Moves from Beier et al. [15] can be applied to the LMP, which is of interest in this chapter. Both have proven to provide good solutions in practice, generating results on par with the state-of-the-art in image segmentation in terms of the BSDS500 benchmark from Arbeláez et al. [5], the PSB dataset for shape segmentation from Chen et al. [32], and the ISBI2012 challenge on segmentation of neuronal structures in image stacks from Cardona et al. [27], Carreras et al. [28]. For problem instances, for example, on the image segmentation task, these heuristics still need several minutes to converge, while Fusion Moves [15] has an edge on KLj in terms of computation time at the price of a greedy data pre-clustering.

The proposed approach is built on the entirely greedy approach of GAEC, which has been proposed as an initialization procedure for KLj in Keuper et al. [89]. As GAEC, KLj, and Fusion Moves, the proposed approach can not provide any bounds for the proposed solution. However, in contrast to KLj and Fusion Moves, it can provide a guaranteed worst-case complexity. In practice, it generates results with a computation time slightly higher than GAEC, while the resulting objective values can be close to the ones from KLj.

6.3 Optimization Problem

In this chapter, we propose a heuristic solver for the LMP [89] (see Section 2.5.1 for the definition of LMP). This problem is of interest because its feasible solutions relate one-to-one to decompositions of a graph and because it provides a principled way to define a more general cost function decompositions than the minimum cost multicut problem (**MP**). A decomposition of the graph $G = (V, E)$ is any partition Π of the vertices V such that every component $V' \in \Pi$ induces a connected subgraph of G.

6.4 Objectives

To generate fast solutions to LMP, we investigate the GAEC solver proposed in Keuper et al. [89] (refer to Algorithm 1 in Chapter 2). While other heuristic solvers such as the one proposed by Beier et al. [15] generate better solutions in terms of the resulting energy, GAEC is one of the fastest currently available, although it operates directly on the nodes of the original graph without any local pre-clustering. GAEC provides deterministic solutions created in a greedy procedure of edge contractions and thus provides low computation times even for large and difficult problem instances [89]. While GAEC provides relatively low energy solutions on MP, it has major difficulties finding acceptable solutions to LMP. Where does this discrepancy come from? To analyze this effect, we visualize intermediate GAEC solutions in Figure 6.1-top. We observe that large clusters are built pretty quickly. They tend to grow by adding small clusters or isolated nodes. In this scenario, long-range repulsive edges are not in the scope of the greedy optimization until almost all nodes have been merged. Thus, the "difficult parts" of the problem, i.e., those parts possibly causing conflicts with respect to the objective, are only considered when most nodes have already been irreversibly merged.

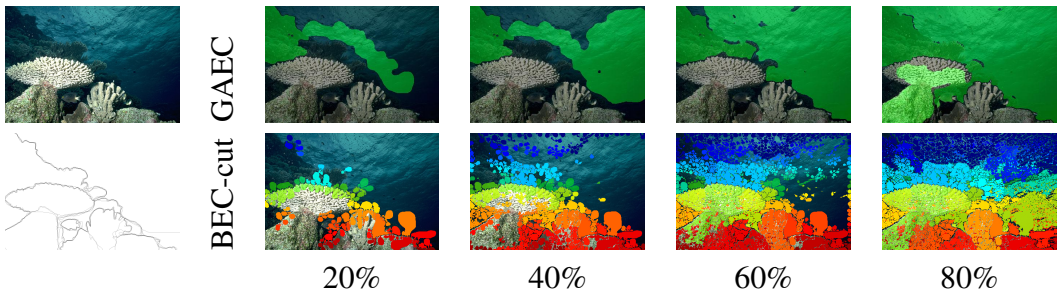


Fig. 6.1 For the BSDS500 [5] image on the left (average ground-truth annotations are depicted below it), we show intermediate states during the execution of the GAEC solver [89] (**top**) and the proposed BEC-cut solver (**bottom**) after 20%, 40%, 60%, and 80% of the total merges have been executed. GAEC tends to generate large segments that merge points across object boundaries. These merges can not be "repaired". In contrast, BEC-cut generates and grows many segments simultaneously. It starts generating these segments in the vicinity of the object boundaries.

We conclude that the GAEC algorithm would benefit from two aspects: (1) avoid creating very unbalanced clusters early on, such that repulsive edges are in the scope of intermediate regions early on, and (2) use information about repulsive edges directly as a criterion for merges.

6.5 Proposed Approach

We want to base our heuristic on the fast GAEC solver from Keuper et al. [89] and follow a similar, greedy merging scheme such that running times remain affordable for large and difficult problem instances. To tackle aspects (1) avoid creating very unbalanced clusters early on and (2) use information about repulsive edges in the merge criterion, we define an improved ranking scheme for the next best edge to be contracted. In the basic version of our proposed heuristic, we only tackle aspect (1) by presenting a greedy "Balanced Edge Contraction" (BEC) scheme. Instead of using the plain edge potentials as a merge criterion as in GAEC (see Algorithm 7, line 6), we propose to normalize these potentials by the size of the components to be merged (Algorithm 8, line 6). Note that this normalization only affects the order in which attractive edges are merged, not the actual edge potentials or the cut objective.

To tackle aspect (2), we formulate a secondary merge criterion. Ultimately, we want to find a cut that minimizes the total energy of the cut (not maximize the energy of the joined components). We are motivated to encourage the solver to pursue this objective during the greedy optimization: We want to make the join decisions that minimize the outgoing potentials of resulting components, i.e., the intermediate cut.

To formalize this, lets look at the graph $G_d = (V, E_d)$ defined on G', and assign to every vertex v in V a unary potential $d_v = \sum_{\{b|e_{vb} \in E'\}} c_{vb}$, i.e. the degree of v. The outgoing potentials of a component in G resulting from joining two vertices u and v can be computed as $d_u + d_v - 2c_{u,v}$. Then, we define for edges $uv \in E_d$, edge weights $c^d_{u,v} = d_u + d_v - 2c_{u,v}$. Minimizing these "dual" edge weights is used as a secondary join criterion in Algorithm 9, BEC-cut.

This modification of the order in which segments are agglomerated affects the intermediate stages of the solution during the optimization. Figure 6.1-bottom shows such intermediate optimization stages after 20%, 40%, 60%, and 80% of the total merges have been computed by the proposed algorithm. Unlike GAEC, it generates many small node agglomerations along both sides of the boundaries. Thus, repulsive terms on the boundary are in scope early on.

6.5.1 Algorithms

Algorithms 8 and 9 are adaptations of greedy agglomeration, more specifically, greedy additive edge contraction (Algorithm 7). Both take as input an instance of the LMP, (see Section 2.5.1), defined by $G = (V, E)$, F and $c : E \cup F \to \mathbb{R}$ (refer to Equations (2.9) - (2.12) in Chapter 2) and construct as output a decomposition of the graph G. Both algorithms maintain

Algorithm 7: GAEC [89]

$\mathscr{E} := E$, $\mathscr{E}' := E'$
$\mathscr{V} := V$
foreach $ab \in \mathscr{E}'$ **do**
    $\chi_{ab} := c_{ab}$
while $\mathscr{E} \neq \emptyset$ **do**
    $ab := \underset{a'b' \in \mathscr{E}}{\operatorname{argmax}} \chi_{a'b'}$
    if $\chi_{ab} < 0$ **then**
        break
    contract ab in $\mathscr{G}$ and $\mathscr{G}'$
    foreach $ab \neq ab' \in \mathscr{E}'$ **do**
        $\chi_{ab'} := \chi_{ab'} + \chi_{bb'}$

Algorithm 8: BEC

$\mathscr{E} := E$, $\mathscr{E}' := E'$
$\mathscr{V} := V$
foreach $ab \in \mathscr{E}'$ **do**
    $\chi_{ab} := c_{ab}$
while $\mathscr{E} \neq \emptyset$ **do**
    $ab := \underset{a'b' \in \mathscr{E}}{\operatorname{argmax}} \frac{\chi_{a'b'}}{|a'|+|b'|}$
    if $\chi_{ab} < 0$ **then**
        break
    contract ab in $\mathscr{G}$ and $\mathscr{G}'$
    foreach $ab \neq ab' \in \mathscr{E}'$ **do**
        $\chi_{ab'} := \chi_{ab'} + \chi_{bb'}$

a decomposition of G, represented by graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ whose nodes $a \in \mathscr{V}$ are components of G and whose edges $ab \in \mathscr{E}$ connect any components a and b of G which are neighbors in G. Objective values are computed with respect to the larger graph $G' = (V, E \cup F)$ and c.

Balanced Edge Contraction (BEC)

Starting from the decomposition into single nodes, in every iteration, a pair of neighboring components is joined, for which the join decreases the objective value. While for the basic GAEC algorithm, an edge is picked such that the objective value decreases maximally, we propose to weight the prospective gain by the size of the components. This weighting encourages components of similar size to merge earlier than components of different sizes if both merges are advantageous. Intuitively, this should lead to several balanced clusters (as opposed to one large cluster and many single nodes) at intermediate optimization states. If no join strictly decreases the objective value, the algorithm terminates.

BEC-cut

Starting from the decomposition into single nodes, at every iteration, a pair of neighboring components is joined, for which the join decreases the objective value. If a unique pair of neighboring components exists whose join strictly decreases the *size-weighted* objective value maximally, this join is executed. From the set of all possible joins that decrease the weighted objective value maximally, the join is executed, which minimizes the sum of outgoing costs, i.e. the prospective value ζ of the resulting component. Refer to Figure 6.2

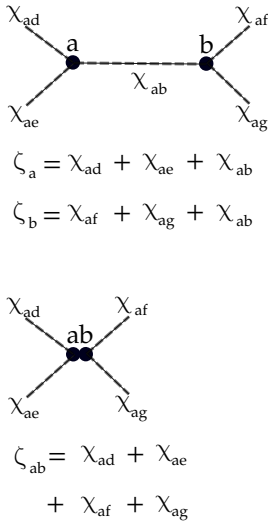


Fig. 6.2 Exemplary computation of ζ from χ before (**top**) and after (**bottom**) contraction of components a and b. χ encodes the sum of outgoing costs of a component.

Algorithm 9: BEC-cut

1 $\mathscr{E} := E$, $\mathscr{E}' := E'$
2 $\mathscr{V} := V$
3 **foreach** $ab \in \mathscr{E}$ **do**
4 $\quad \chi_{ab} := c_{ab}$
5 **foreach** $a \in \mathscr{V}$ **do**
6 $\quad \zeta_a := \sum_{\{b|e_{ab}\in E'\}} c_{ab}$
7 **while** $\mathscr{E} \neq \emptyset$ **do**
8 $\quad \mathscr{S} := \operatorname{argmax}_{a'b'\in\mathscr{E}} \frac{\chi_{a'b'}}{|a'|+|b'|}$
9 $\quad ab := \operatorname{argmin}_{a'b'\in\mathscr{S}} \frac{\zeta_{a'}+\zeta_{b'}-2\chi_{a'b'}}{|a'|+|b'|}$
10 $\quad$ **if** $\chi_{ab} < 0$ **then**
11 $\quad\quad$ **break**
12 $\quad$ **contract** $ab =: \bar{a}$ in $\mathscr{G}$ and $\mathscr{G}'$
13 $\quad \zeta_{\bar{a}} = \zeta_a + \zeta_b - 2\chi_{ab}$
14 $\quad$ **foreach** $ab \neq ab' \in \mathscr{E}'$ **do**
15 $\quad\quad \chi_{ab'} := \chi_{ab'} + \chi_{bb'}$

for an illustration of the computation of ζ. If no join strictly decreases the objective value, the algorithm terminates.

Implementation

Our implementation is built upon GAEC [89] (Section 2.5.2). As GAEC, it uses ordered adjacency lists for the graph $\mathscr{G}$ and for a graph $\mathscr{G}' = (\mathscr{V}, \mathscr{E}')$ whose edges $ab \in \mathscr{E}'$ connect any components a and b of G for which there is an edge $vw \in E \cup F$ with $v \in a$ and $w \in b$. It uses a *disjoint set data structure* for the partition of V and a *priority queue* for an ordered sequence of costs $\chi : \mathscr{E} \to \mathbb{R}$ of feasible joins, with a secondary sorting criterion $\bar{\chi} : \mathscr{E} \to \mathbb{R}, \bar{\chi}_{ab} = \frac{\zeta_a+\zeta_b-2\chi_{ab}}{|a|+|b'|}$ defined on ζ for Algorithm 9 (compare line 9). As GAEC, its worst-case time complexity $O(|V|^2 \log |V|)$ is due to a sequence of at most $|V|$ contractions, in each of which at most $\deg \mathscr{G}' \leq |V|$ edges are removed, each in time $O(\log \deg \mathscr{G}') \in O(\log |V|)$.

6.6 Experiments

We evaluate the proposed heuristics on types of problem instances and with respect to three different tasks: lifted pixel-grid graphs from Keuper et al. [89] defined on the image segmentation problems of the BSDS500 [5] benchmark, lifted graphs on 3D shape meshes [89] defined on the Princeton Shape Segmentation benchmark [32] and the lifted superpixel adjacency graph from Beier et al. [18] defined on the ISBI 2012 challenge data, a volumetric electron microscopic recording of neuronal structures [27, 28].

6.6.1 Image Decomposition

On the image segmentation task posed by the BSDS500 benchmark [5], we apply the proposed solvers BEC (Algorithm 8) and BEC-cut (Algorithm 9) to the LMP instances proposed in Keuper et al. [89]. These problem instances represent pixel grid graphs. Lifted "long-range" edges are inserted to connect all pixels within a pre-defined pixel radius to provide more "global" information to the optimization problem. Lifting radii of 20 pixels showed the best performance in Keuper et al. [89].

Figure 6.3 shows the results of the proposed solvers in comparison to GAEC and KLj initialized with GAEC in terms of the variation of information (VI), Meilă [115], (see Figure 6.3-left, lower is better) and the boundary precision and recall (see Figure 6.3-right, higher is better). To evaluate these metrics, results at different levels of granularity need to be generated. This can be done by modifying the prior probability of a cut in the lifted multicut problem instances. Every point in the plots of Figure 6.3 shows, for one cut prior and algorithm, the average over all the test images in the benchmark. While BEC and BEC-cut show almost equivalent performance, the improvement over GAEC [89] especially on the region metric VI, is evident. Figure 6.4 shows scatter plots of the computation times and resulting objective values for the algorithms GAEC and KLj [89] and the proposed solvers (BEC-cut and BEC). Concerning the objective value, both BEC-cut and BEC yield a clear improvement over GAEC while significantly faster than KLj.

Some qualitative results are shown in Figure 6.5 for the proposed BEC-cut solver and lifting radius 10. Although it is a greedy merging procedure, it can produce nicely closed contours on these problem instances. Here we provide exemplary results for the BSDS500 benchmark to compare GAEC and our proposed solver BEC-cut.

In Figure 6.6, some of the results are shown, which are acquired by the GAEC [89] solver and our proposed approach BEC-cut. Our proposed approach provides an image decomposition in comparable runtime as GAEC while the quality of the segmentations is significantly better. GAEC [89] fails to generate closed contours. Figure 6.6-right shows

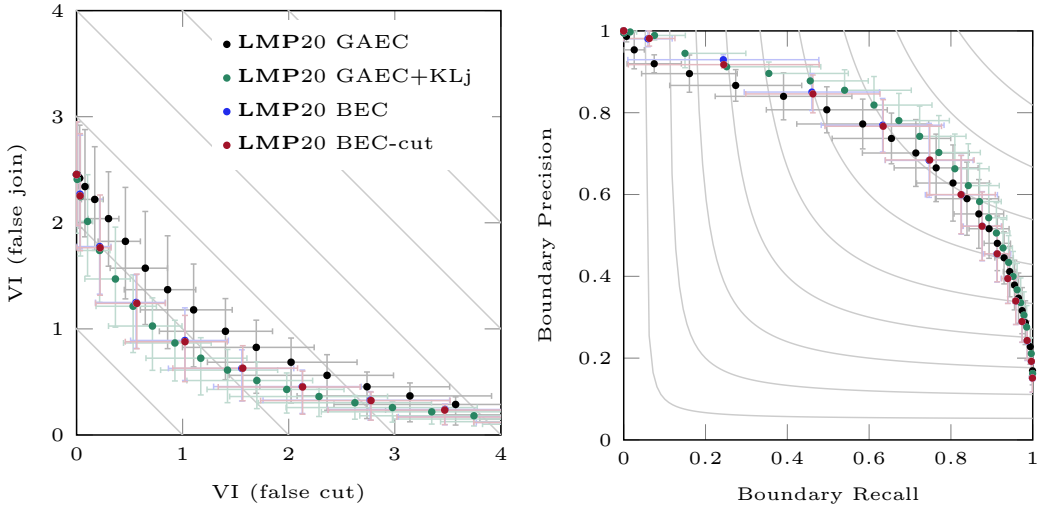


Fig. 6.3 Depicted above is an evaluation of the heuristics GAEC, (Algorithm 7), KLj [89] and the proposed BEC (Algorithm 8) and BEC-cut (Algorithm 9) on the large and difficult lifted multicut problem instances from [89] with lifting radius 20 (LMP20). These instances address the image decomposition problem posed by the BSDS500 benchmark [5]. On the **left**, the variation of information (VI), split additively into a distance due to false cuts and a distance due to false joins, is depicted (lower is better); on the **right**, the accuracy of boundary detection, split into recall and precision is shown (higher is better). Error bars depict the 0.25 and 0.75-quantile. The result of the proposed heuristics figure in between those of the solvers GAEC and KLj in both metrics. In the region metric VI, the results of the proposed solvers BEC and BEC-cut are very close to the results of KLj.

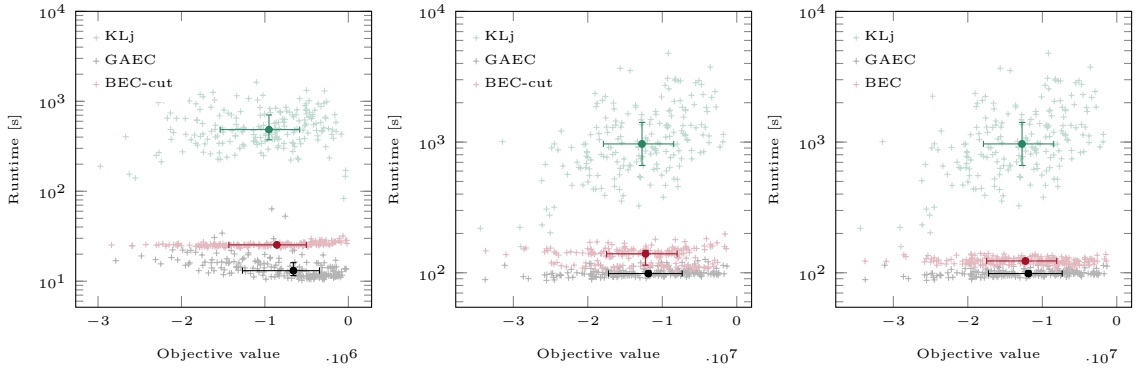


Fig. 6.4 Depicted above is a comparison of Algorithm 9 (BEC-cut) and Algorithm 8 (BEC) with GAEC and KLj [89]. Every point corresponds to one instance of the lifted multicut problem [89] defined with respect to one test image in the BSDS500 benchmark [5] with lifting radius 10 (**left**) and 20 (**middle** and **right**). The computation times of the proposed algorithms BEC-cut (**middle**) and BEC (**right**) are close to the ones from GAEC while the resulting energy is improved.

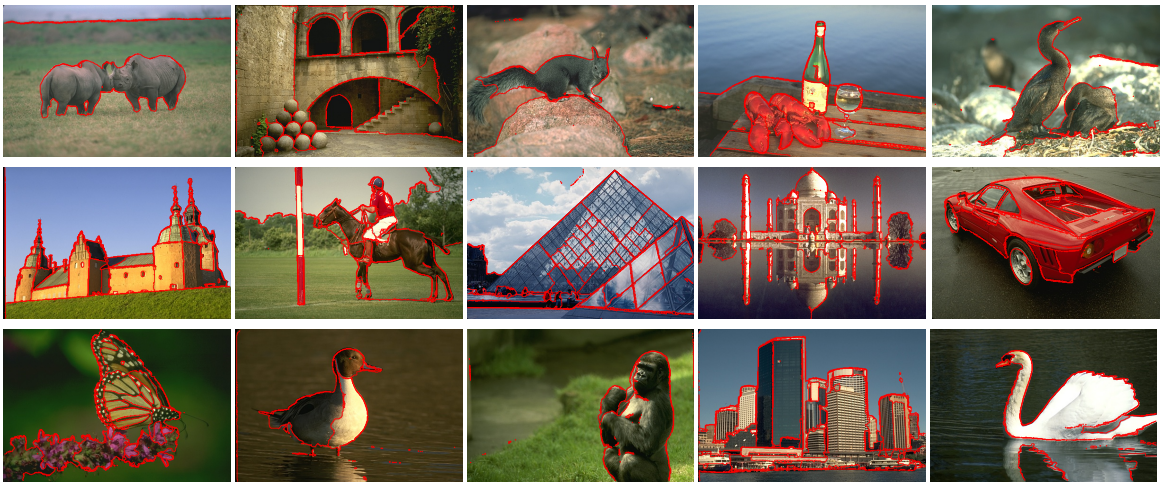


Fig. 6.5 Qualitative segmentation results of the proposed BEC-cut solver on instances from the BSDS500 benchmark (Section 2.1.3). The lifted multicut problems are computed with lifted edges between all points at a distance less than 10.

the final results from the proposed BEC-cut and the GAEC [89] solver on the example from Figure 6.1. GAEC [89] only segments isolated pixels on the boundary while BEC-cut provides a meaningful segmentation into closed segments. In Table 6.1, we report quantitative results for additional BSDS500 [5] metrics. On BSDS500, both versions of the proposed solvers perform similarly well, while BEC-cut is slightly better in terms of the objective value and has a marginally longer computation time.

6.6.2 Mesh Segmentation

Next, we evaluate on the Princeton Shape Segmentation benchmark [32] (Section 2.1.4). In this dataset, we apply the LMP formulation defined in Keuper et al. [89] in their experiments on mesh segmentation. The local edge costs for $e \in E$ are computed using curvatures and dihedral angle features. Lifted edges are then inserted up to a fixed node distance of 70 (defined on the graph) and for cut prior of 0.55 as proposed in [89] using their code [1]. Consequently, the created problem instances are solved by the proposed solvers. Qualitative results including failure cases are shown in Figure 6.9.

Table 6.2 compares the resulting objective functions and the consumed optimization time of the different approaches. On this dataset, the optimization by KLj initialized with GAEC or BEC converges surprisingly fast. This might be an indication of a meaningful initial segmentation in these cases. Among the greedy heuristics, BEC yields the lowest objective

[1] https://www.mpi-inf.mpg.de/fileadmin/inf/d2/levinkov/iccv-2015/code.tar.gz

Fig. 6.6 Results from our proposed solver BEC-cut are provided as well as the GAEC solver. The results from the solvers on the example from Figure 6.1 are shown in the third column. GAEC only segments isolated pixels on the boundary, while BEC-cut provides a meaningful segmentation into closed segments.

values. Initializing KLj with the proposed BEC produces a better objective value than KLj initialized with GAEC, Keuper et al. [89].

Table 6.3 provides a comparison between different solvers in terms of Rand's Index (RI) (higher is better) and Variation of Information (VI) (lower is better). Additionally, to report the plain results of our proposed solvers, we evaluate the quality of solutions from KLj, initialized with our solutions, i.e., KLj-BEC. The iterative, local move making heuristic KLj [89] is known to benefit from good initializations.

The evaluation shows that both proposed heuristics outperform GAEC. Specifically, BEC yields an improvement of 0.05 on the RI and a decrease of 0.08 in the VI. On this dataset, BEC performs better than BEC-cut. In fact, it yields a RI value close to the one from KLj-GAEC with the same lifting parameters. KLj, initialized with BEC, yields slightly better results than KLj initialized with GAEC in terms of the VI. The last column shows results from Keuper et al. [89], which could be obtained by KLj-GAEC when the lifting radius and cut prior were optimized per instance. KLj, initialized by BEC, almost meets these results.

Some of the results for the BEC and KLj-BEC solvers are provided in Figure 6.10. The results generated from our proposed solver BEC are comparable with the KLj-BEC, while BEC demands less computation time. A scatter plot showing the resulting computation times and objective values per instance is given in Figure 6.7.

Table 6.1 Written below are boundary and volume metrics measuring the distance between the man-made decompositions of the BSDS500 benchmark [5] and the decompositions defined by lifted multicuts (LMP) and top-performing competing methods Arbeláez et al. [5], Arbelaez et al. [6], Dollár and Zitnick [41]. Parameters are fixed for the entire data set (ODS).

	Boundary	Volume		
	F-measure	Covering	RI	VI [115]
gPb-owt-ucm [5]	0.73	0.59	0.83	1.69
SE+MS+SH [41]+ucm	0.73	0.59	0.83	.71
SE+multi+ucm [6]	0.75	0.61	0.83	1.57
LMP10 GAEC	0.71	0.51	0.80	2.33
LMP10 **BEC**	0.71	0.56	0.82	1.85
LMP10 **BEC-cut**	0.71	0.56	0.82	1.84
LMP10 GAEC-KLj	0.73	0.58	0.82	1.76
LMP20 GAEC	0.71	0.52	0.80	2.22
LMP20 **BEC**	0.71	0.56	0.82	1.81
LMP20 **BEC-cut**	0.71	0.56	0.82	1.81
LMP20 GAEC-KLj	0.73	0.58	0.82	1.74

Table 6.2 Average computation time and objective value of the different solvers over the Princeton Shape Segmentation Benchmark are provided.

	GAEC [89]	KLj-GAEC [89]	BEC	BEC-cut	KLj-BEC-cut	KLj-BEC
Avg. Comp. time [s]	576	755	589	574	797	755
Avg. Objective Value (lower is better)	-17840450	-18988930	-18484140	-18057480	-18991920	-18989332

6.6.3 ISBI 2012 Challenge

In the ISBI 2012 challenge, a stack of electron microscopy images of neural structures [27, 28] is provided for segmentation (see Section 2.1.5).

We applied the proposed BEC, and BEC-cut solver to the instance of the LMP defined in Beier et al. [18]. The instance is based on a superpixel adjacency graph, i.e., every node corresponds to a superpixel and is connected to its direct neighbors. An additional *lifted* edge set encodes long-range information. All edge weights are learned from training data using two random forest classifiers; one is trained to predict whether two adjacent superpixels should be assigned to the same cluster, and the other one learns this prediction for non-adjacent superpixels. The trained classifiers are used to assign weights to the edges

Table 6.3 Resulting RI and VI score for the 3D mesh segmentation instances of the Princeton Segmentation Benchmark [32]. We evaluate the proposed solvers BEC and BEC-cut and compare them to GAEC [89] and KLj [89] initialized by GAEC. We also evaluate KLj, initialized with our results from BEC.

	GAEC [89]		BEC (proposed)		BEC-cut (proposed)		KLj-GAEC [89]		KLj-BEC		KLj-GAEC-opt [89]	
	RI	VI	RI	VI	RI	VI	RI	VI	RI	VI	RI	VI
Human	0.77	1.90	0.84	1.77	0.85	1.74	0.86	1.63	0.86	1.60	0.87	1.79
Cup	0.83	0.53	0.88	0.44	0.86	0.45	0.88	0.40	0.89	0.37	0.90	0.39
Glasses	0.61	1.38	0.82	0.85	0.78	1.04	0.84	0.78	0.84	0.77	0.90	0.68
Airplane	0.89	0.93	0.91	0.92	0.91	0.86	0.91	0.84	0.91	0.85	0.92	0.83
Ant	0.93	0.70	0.97	0.48	0.96	0.52	0.97	0.43	0.97	0.44	0.98	0.42
Chair	0.85	0.85	0.92	0.67	0.89	0.84	0.92	0.58	0.92	0.58	0.93	0.55
Octopus	0.96	0.41	0.97	0.41	0.97	0.44	0.97	0.37	0.97	0.37	0.98	0.33
Table	0.87	0.48	0.91	0.40	0.83	0.63	0.93	0.29	0.93	0.29	0.94	0.29
Teddy	0.92	0.76	0.93	0.76	0.92	0.78	0.95	0.55	0.95	0.55	0.96	0.50
Hand	0.67	1.75	0.80	1.47	0.76	1.72	0.84	1.23	0.84	1.23	0.85	1.32
Plier	0.72	1.38	0.84	1.12	0.81	1.19	0.88	0.93	0.91	0.85	0.93	0.84
Fish	0.73	1.20	0.75	1.26	0.76	1.30	0.80	1.12	0.80	1.11	0.80	1.09
Bird	0.90	1.01	0.92	0.98	0.91	1.01	0.92	0.91	0.92	0.91	0.93	0.99
Armadillo	0.88	1.88	0.91	1.75	0.90	1.87	0.91	1.63	0.91	1.62	0.92	1.48
Bust	0.57	2.12	0.66	2.32	0.63	2.46	0.68	2.29	0.68	2.29	0.69	2.25
Mech	0.80	0.66	0.81	0.72	0.80	0.66	0.84	0.60	0.84	0.59	0.84	0.59
Bearing	0.79	0.76	0.82	0.75	0.83	0.70	0.84	0.69	0.84	0.69	0.84	0.69
Vase	0.77	1.02	0.80	1.06	0.77	1.13	0.84	0.87	0.84	0.87	0.84	0.87
FourLeg	0.81	1.94	0.82	1.98	0.81	2.04	0.83	1.87	0.83	1.89	0.84	1.72
Average	0.81	1.14	0.86	1.06	0.84	1.13	0.88	0.95	0.88	0.94	0.89	0.93

Table 6.4 Objective value and run time of the proposed solvers, and fusion move algorithm with the randomized proposal generator (FM-R), Beier et al. [15], for the LMP for ISBI 2012 Challenge [27, 28] are provided.

Algorithm	Comp. time [s]	Objective Value (lower is better)	V^{Rand}	V^{Info}
FM-R	10.43	-1.948524	0.9822	0.9884
BEC	2.53	-1.944767	-	-
BEC-cut	2.59	-1.944964	0.9811	0.9883

and lifted edges, respectively. Table 6.4 shows the results we achieve on this data with our heuristic solvers BEC and BEC-cut. At a significantly reduced computation time, we can get results close to the state-of-the-art in terms of objective value and segmentation quality. Figure 6.8 represents the first frame in the stack of the test data from ISBI 2012 with the corresponding segmentation boundaries resulting from the proposed BEC-cut solver.

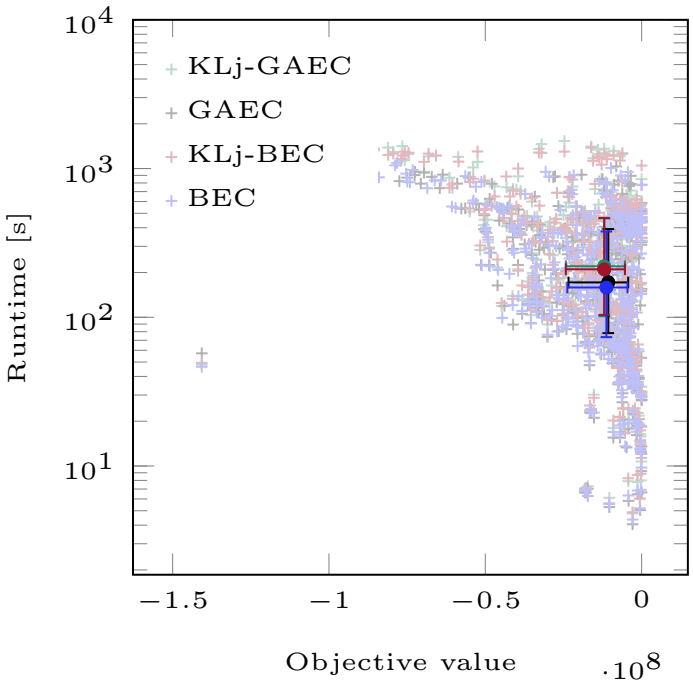


Fig. 6.7 Depicted above is a comparison of BEC with GAEC and KLj [89], initialized with both BEC and GAEC, in terms of computation time over the resulting objective value on the Princeton Shape Segmentation Benchmark.

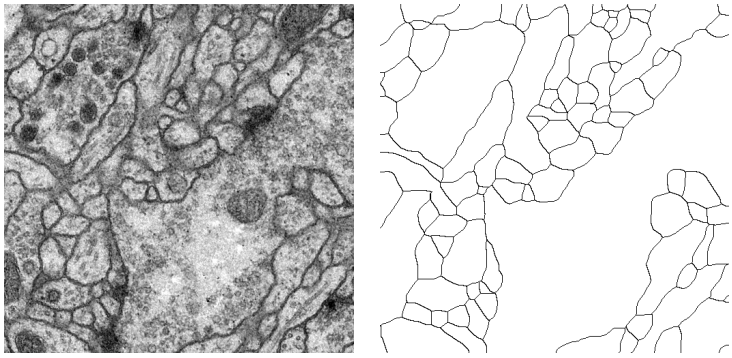


Fig. 6.8 The first frame of the stack in the test data of ISBI 2012 and the corresponding segmentation boundaries are shown. The results for this dataset are acquired with a BEC-cut solver.

6.7 Conclusion

In this chapter, we propose two heuristic solvers, BEC and BEC-cut. Both are modifications of the greedy agglomerative heuristic GAEC and provide a bounded worst-case complexity. We evaluate our solvers on three application scenarios: image decomposition, 3D mesh

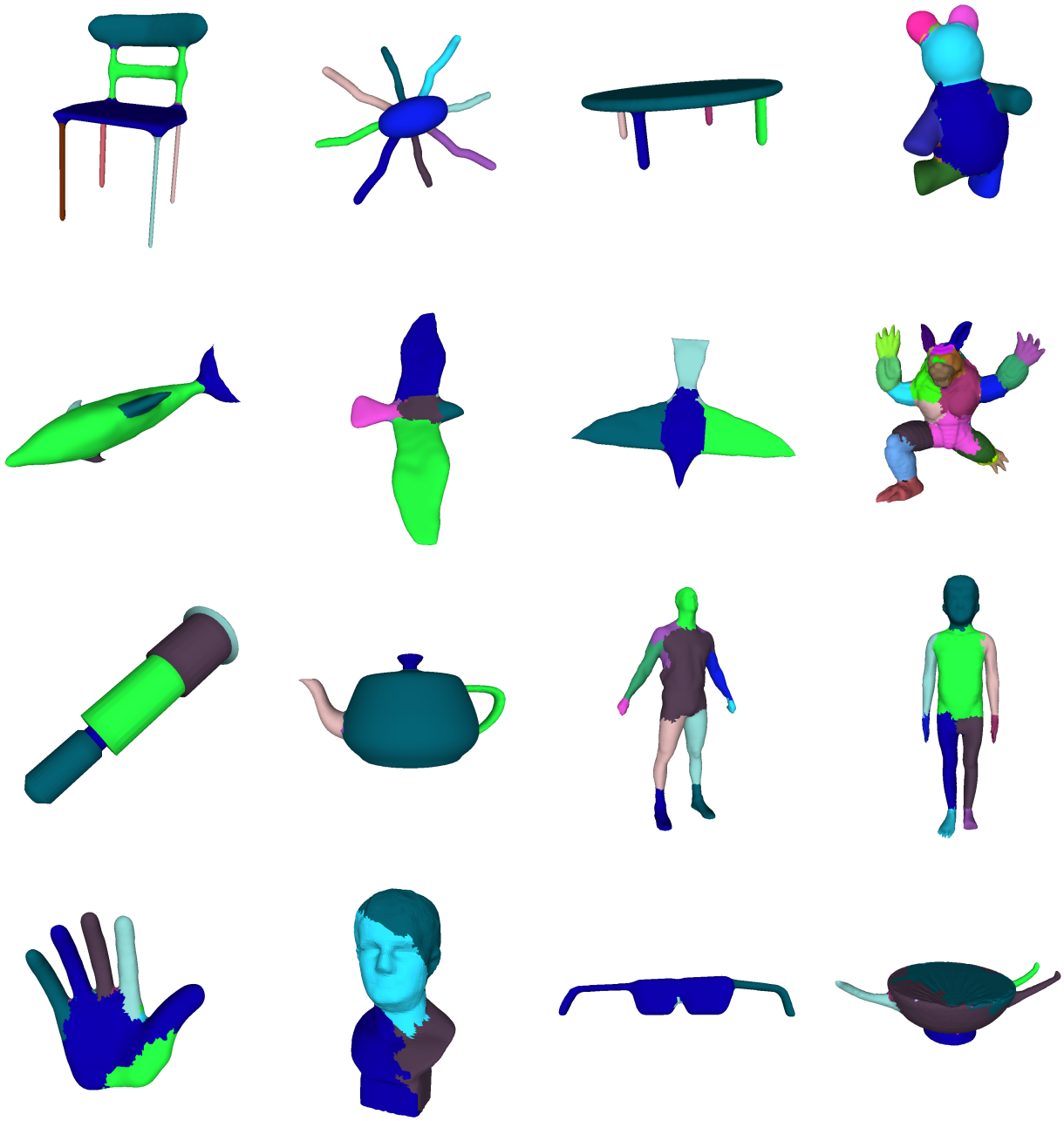


Fig. 6.9 Some results generated by the proposed BEC-cut heuristic are shown for the Princeton Shape Segmentation Benchmark. The last row shows failure cases.

segmentation, and segmentation of neuronal structures from volumetric electron microscopic data. In all scenarios, the proposed solvers yield results close to the ones from more elaborate

solvers KLj from Keuper et al. [89] or Fusion Moves from Beier et al. [15], while the computation time is significantly lower.

While we here proposed a purely heuristic solver based on a primal feasible heuristic, we acknowledge that end-to-end learnable weights in the context of MP are desirable. I contributed to one such approach as a co-author in the context of a supervised master thesis in the work briefly summarized in the following paragraph.

6.7.1 End-to-End Multicut Graph Decomposition

Song et al. [143] proposed an end-to-end trainable network for the human-pose estimation where the cycle inequality constraints of the multicut are used as higher-order potentials in a Conditional Random Field (CRF), as they always consist of three edge variables. The learned edge weights are sub-optimal because they are built on the loose relaxation of the multicut problem, Jung et al. [74]. In work based on the master thesis of Sebastian Ziegler and consolidated further by Steffen Jung [74], we have proposed an adaptive cooling-based CRF formulation to enforce more binary edge weights (close to 0 or 1). This leads to less violated cycle constraints. In this approach, all the triangles in the multicut graph are transformed into binary cubic problems. The model sets a low cost on the valid edge label configuration (e.g., (join, join, join), (join, cut, cut), and (cut, cut, cut)) and a high cost for invalid edge label configuration (e.g., (cut, join, cut)) [74]. The model pushes the edge values larger or equal to 0.5 to get closer to 1 and edge values below 0.5 to get closer to 0. The approach is evaluated on the BSDS500 [5] and ISBI challenge [15, 18], where it shows improvements over the approach by Song et al. [143].

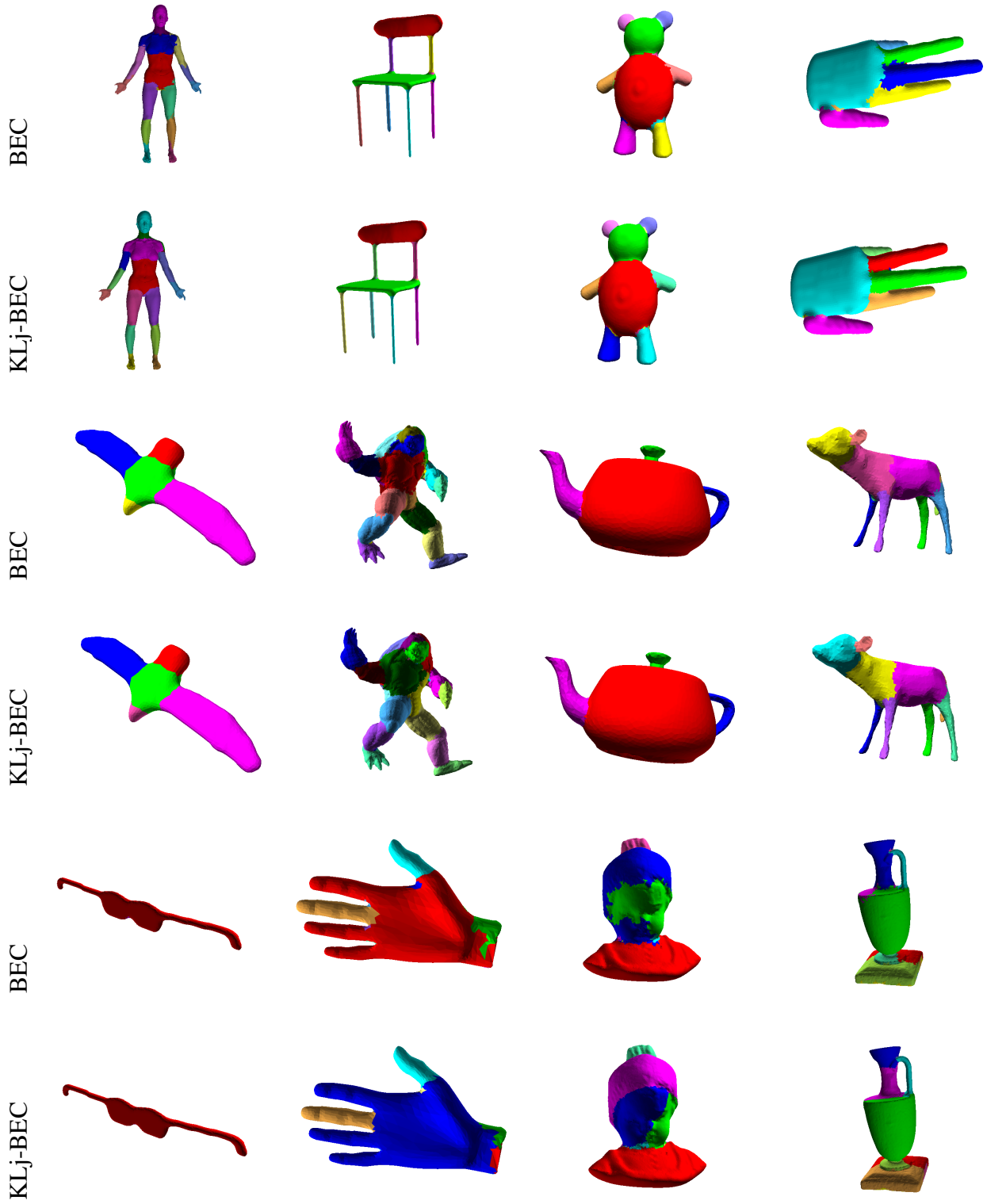


Fig. 6.10 Some of the results for the Princeton shape segmentation benchmark are provided for two heuristic solvers, BEC and KLj-BEC. The first four rows show the successful segmentation, and the last two rows represent the failure cases. The results corresponding to the solvers are shown in pairs of rows for better comparison purposes. The results from BEC are acquired with less computation time than KLj-BEC, while the segmentation qualities are similar.

Chapter 7

Uncertainty Prediction in Minimum Cost Multicuts

In the last chapter, we proposed two solvers for the minimum cost (*lifted*) multicut (**LMP**) framework. It addresses the problems in a graph-based model, where real-valued costs are assigned to the edges between entities such that the minimum cut decomposes the graph into an optimal number of segments. Driven by a probabilistic formulation of minimum cost multicuts (**MP**), in this chapter, we provide a measure for the uncertainties of the decisions made during the optimization. We argue that access to such uncertainties is crucial for many practical applications and conduct an evaluation by means of sparsifications on three different, widely used datasets in the context of image decomposition (BSDS500, Arbeláez et al. [5]) and motion segmentation (DAVIS_{2016}, Perazzi et al. [128] and FBMS59, Ochs et al. [123]) in terms of variation of information (VI) and Rand index (RI), (for the information about the datasets refer to Section 2.1). This work is published in the Uncertainty in Artificial Intelligence (UAI) conference, 2021 [83].

7.1 Introduction

Andres et al. [3], relate the minimum cost multicut to a probabilistic model, which was further extended in Keuper et al. [89] to *lifted* multicuts. They show that if the edge costs are set according to logits of the cut probability between nodes, the minimum cost multicut provides the Maximum A Posteriori probability (MAP) estimate. Motivated by this finding, edge cost definitions based on cut probability estimates have become common practice, even in settings where the final solution is estimated using primal feasible heuristics such as Keuper et al. [88], Beier et al. [16] to account for the NP-hardness of the problem, Bansal et al. [12].

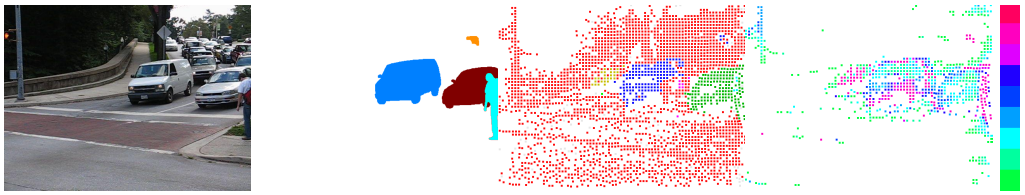


image ground-truth multicut uncertainty

Fig. 7.1 Motion segmentation and the proposed uncertainty measure on a street scene. The uncertainty is high on incorrectly segmented points, specifically the missed person.

In this chapter, we argue that considering the (approximate) MAP solution solely is not satisfying in all scenarios. For instance, in an automotive setting, it might be required to assess the uncertainty in predicting the number of moving objects and their shapes. Therefore, we propose to employ the probabilistic model from Andres et al. [3] and derive, for a proposed solution, a measure for the uncertainty of each node-to-label assignment. An example is provided in Figure 7.1 for sparse motion segmentation. Based on sparse point trajectories, we apply the motion model derived by Ochs et al. [123] and provide the minimum cost multicut solution, Keuper et al. [88]. Depending on the exact parameters employed, the person standing nearby the street can not be correctly segmented. Yet, the uncertainty indicates potential mistakes in the prediction.

The proposed uncertainty measure is directly derived from the probabilistic multicut formulation and can be applied to *any* given decomposition. We evaluate our approach in the context of MP for sparse motion segmentation using the model from Keuper et al. [88] on the datasets FBMS59 [123] and DAVIS$_{2016}$ [128] (see Section 2.1). Further, we investigate the potential benefits of the predicted uncertainties for generating dense motion segmentations from sparse ones. Such densifications can be computed using convolutional neural networks trained in a self-supervised way, for example, using our proposed approach in [81] which is explained in Chapter 4. Last, we evaluate the proposed uncertainty measure in the context of LMP for image decomposition on BSDS500 [5]. In both applications, motion and image segmentation, we show via sparsification plots that subsequently removing uncertain predictions from the solution improves segmentation metrics. The proposed measure is thus a robust indicator of the uncertainty of a given solution.

7.2 Related Work

In the following, we summarize related work with respect to the uncertainty estimation and the prior work on multicuts in the considered application domains.

Our proposed uncertainty measure is defined on the formulation of the minimum cost (*lifted*) multicut problem (see Section 2.5.1 for the definition) and can be applied to any given solution. We perform a thorough evaluation in the context of the widely used KLj and GAEC heuristics from Keuper et al. [89]. For all of the applications of the minimum cost (*lifted*) multicut, (see Section 2.5.3), one can easily imagine use-cases that benefit from a measure of uncertainty. We evaluate the proposed approach on image and motion segmentation tasks.

Uncertainty Estimation. Kohli and Torr [96] proposed a measure of uncertainty for the graph cut solutions using the min-marginals associated with the label assignments in a Markov Random Field (MRF). Concerning minimum cost multicuts, Kappes et al. [76] measure the uncertainty of image partitions by an approximate marginal distribution using cost perturbations and induced *edge* label flips. The proposed approach is complementary to this method as it does not measure the uncertainty of a binary edge labeling but assesses the uncertainty of the induced *node* labeling. In Tomczak et al. [151], the Perturb and MAP (PM) approach is used for learning a restricted Boltzmann machine. In this method, each of the observable and hidden variables is flipped to check the change in the energy function. We propose a measure of uncertainty on the cut/join decisions of the graph in minimum cost (*lifted*) multicut formulation. To show the applicability of our uncertainty measure in real-world scenarios, we evaluate on two *motion segmentation* and one *image decomposition* benchmarks.

More specifically, we study the proposed uncertainty measure on motion segmentation (Section 2.4) in the datasets FBMS59 [123] and DAVIS_{2016} [128] (Section 2.1). Additionally, we use uncertainties to improve the training of the densification model of ours in [81] (refer to Chapter 4).

Image Decomposition For image decomposition, the MP is defined over sets of pixels or superpixels, for example, in Arbeláez et al. [5], Keuper et al. [89], Kappes et al. [79, 78], Andres et al. [2]. In this scenario, the pixels act as nodes in the graph, and their connectivity is defined by edges [2], i.e., directly neighboring pixels are connected. The extension of this problem using *lifted* mulitucts is defined in Keuper et al. [89], where lifted edges are used to encode long-range information while the connectivity of the original graph is preserved. This can lead to an improved pixel-level clustering behavior. Kappes et al. [79] use a higher order multicut model on superpixels for the task. Orbanz and Buhmann [124] provide a non-parametric Bayesian model for histogram clustering to determine the number of image segments, utilizing the Dirichlet process mixture model. Cutting plane and integer programming techniques are used in Kappes et al. [78] for image decomposition. Further, they proposed an approximate solution by solving polynomial-size linear programming.

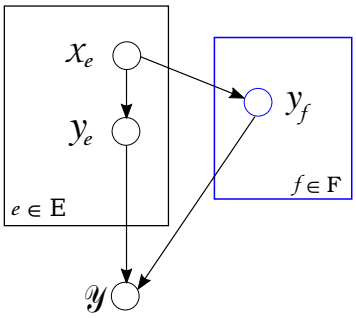


Fig. 7.2 Bayesian Network from Keuper et al. [89], defining a set of probability measures on multicuts (MP) (black) and *lifted* multicuts (LMP) (blue) are shown.

7.3 Uncertainties in Minimum Cost (Lifted) Multicuts

7.3.1 Probability Measures

Andres et al. [3] show that minimum cost multicuts (MP) can be derived from a Bayesian model (see Figure 7.2), and their solutions correspond to the maximum a posteriori estimates. In the following, we summarize this probabilistic model and its extension to lifted multicuts (LMP) from Keuper et al. [89], upon which we build the proposed uncertainty estimation.

Probability Measures on MP and LMP. For a graph $G = (V, E)$, Andres et al. [3] assume that likelihoods $p_{\mathcal{X}_e|\mathcal{Y}_e}$ are computed based on an affinity definition of the nodes in graph G. Further, the costs assigned to the edges are assumed to be independent of each other and the graph's topology G. Moreover, the prior $p_{(\mathcal{Y}_e)}$ is assumed to be identical for all edges $e \in E$ and is specified by a value $\beta \in (0,1)$, so that $p_{\mathcal{Y}_e=1} = \beta$ and $p_{\mathcal{Y}_e=0} = 1-\beta$. In this setting, they show that the MP maximizes the posterior probability $p_{\mathcal{Y}|\mathcal{X},\mathscr{y}}$ of a joint labeling $y \in \{0,1\}^{|E|}$. By definition, the Maximum a Posteriori (MAP) is then

$$p_{\mathcal{Y}|\mathcal{X},\mathscr{y}} \propto p_{\mathscr{y}|\mathcal{Y}} \cdot \prod_{e \in E} p_{\mathcal{Y}_e|\mathcal{X}_e} \cdot p_{\mathcal{Y}_e} \quad . \tag{7.1}$$

for the MP, Figure 7.2 (the black part) [3], and it can be extended to the LMP, Figure 7.2 (the blue part) as follows [89]

$$p_{\mathcal{Y}|\mathcal{X},\mathscr{y}} \propto p_{\mathscr{y}|\mathcal{Y}} \cdot \prod_{e \in E} p_{\mathcal{Y}_e|\mathcal{X}_e} \cdot p_{\mathcal{Y}_e} \cdot \prod_{f \in F} p_{\mathcal{Y}_f|\mathcal{X}_E} \cdot p_{\mathcal{Y}_f} \quad . \tag{7.2}$$

In both cases, $p_{\mathscr{Y}|\mathcal{Y}}$ indicates the feasibility of a solution, i.e.

$$p_{\mathscr{Y}|\mathcal{Y}}(Y_{E'}, y) \propto \begin{cases} 1 & \text{if } y \in Y_{E'} \\ 0 & \text{otherwise} \end{cases} . \tag{7.3}$$

where $E' = E$ for the MP and $E' = E \cup F$ for the LMP.

Equations (7.1) and (7.2) can be maximized by minimizing their negative log-likelihoods. This leads to the definition of instances of the MP (see Equation (2.8)) and LMP (see Equation (2.9)), by setting edge costs according to

$$\forall e \in E: \quad c_e = \log \frac{p_{\mathcal{Y}_e|\mathcal{X}_e}(0, x_e)}{p_{\mathcal{Y}_e|\mathcal{X}_e}(1, x_e)} + \log \frac{1-\beta}{\beta} . \tag{7.4}$$

The value of the scalar β is the cut prior and assumed to be 0.5 for a bias-free case.

7.3.2 Uncertainty Estimation Model

Based on the probabilistic formulation of the MP (7.1) and LMP (7.2), we can study the uncertainty of given feasible solutions. More specifically, we aim to assign to every node in V confidence reflecting the certainty of the assigned label. The simplest attempt to this goal would be to directly assess the posterior probability of the solution. In the following, we briefly sketch this baseline approach before introducing the proposed measure.

Baseline Approach. For every node $v_i \in V$ we propose, as a baseline approach, to draw an uncertainty measure from the proxy to the posterior probability, for example, in Equation (7.1). Thus, the confidence of the given label A of node v_i is

$$\prod_{e=(v_i,v_j), v_j \in A} p_{\mathcal{Y}_e|\mathcal{X}_e}(0, x_e) . \prod_{e=(v_i,v_j), v_j \in V \setminus A} p_{\mathcal{Y}_e|\mathcal{X}_e}(1, x_e) . \tag{7.5}$$

This measure intuitively aggregates the local join probabilities of all edges adjacent to v_i and joined (set to 0) in the current solution, and the local cut probabilities of all edges adjacent to v_i and cut (set to 1) in the current solution. Accordingly, Equation (7.5) yields a low value if the local probabilities for the given solution are low.

A potential issue with this simple approach from Equation (7.5) is that it under-estimates the confidence in many practical scenarios, especially when the local probabilities are

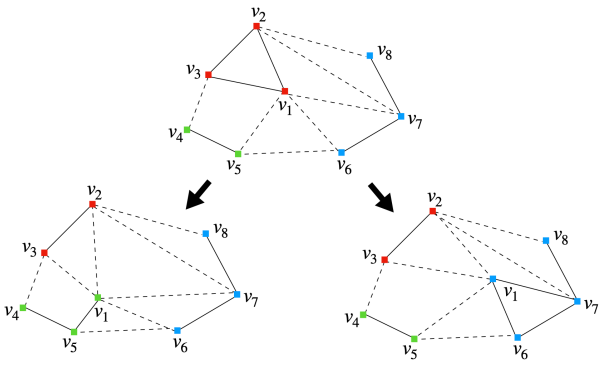


Fig. 7.3 In the current decomposition of the exemplary graph $G = (V, E)$ (top figure), we study the node uncertainties as represented in Equation (7.11). For instance, v_1 is moved from one partition (red label) to the new possible partitions (blue and green labels), and the cost change is estimated. The γ_α represents the cost that minimizes the cost among these moves.

uncertain and the connectivity is dense. Then, the product of local probabilities will issue a low value even if all local cues agree. Therefore, we describe in the following the derivation of the proposed, calibrated uncertainty measure.

Uncertainty Estimation. Given an instance of the LMP and its solution, we employ the probability measures in equations (7.1) (MP) and (7.2) (LMP). We iterate through nodes $v_i \in \{1, \dots, |V|\}$ in vicinity of a cut, i.e. $\exists e \in \mathcal{N}_E(v_i)$ with $e \in E$ and $y_e = 1$. Assuming that v_i belongs to segment A and its neighbour v_j according to E belongs to the segment B, the amount of cost change γ_B is computed in the linear cost function (defined in (2.8) and (2.9)) by moving v_i from cluster A to cluster B as

$$\gamma_B = \sum_{v_j \in \mathcal{N}_{E'}(v_i) \cap A} c_{(v_i, v_j)} - \sum_{v_j \in \mathcal{N}_{E'}(v_i) \cap B} c_{(v_i, v_j)}. \tag{7.6}$$

Thus, in γ_B, we accumulate all costs of edges from v_i that are not cut in the current decomposition and subtract all costs of edges that are cut in the current decomposition but would not be cut if v_i is moved from A to B. Note that, while the cost change is computed over all edges in E' for lifted graphs, only the uncertainty of nodes with an adjacent cut edge in E can be considered to preserve the feasibility of the solution. For each node v_i, the number of possible moves depends on the labels of its neighbors $\mathcal{N}_E(v_i)$, and Equation (7.6) allows us to assign a cost to any such node-label change. Altogether, we assess the uncertainty of a given node label by the cheapest, i.e., the most likely, possible move

$$\gamma_i = \min_B \gamma_B. \tag{7.7}$$

and set γ_i to ∞ if no move is possible. The minimization in Equation (7.7) corresponds to considering the local move of v_i which maximizes

$$\prod_{e=(v_i,v_j),v_j\in A} \frac{p_{\mathcal{Y}_e|\mathcal{X}_e}(1,x_e)}{p_{\mathcal{Y}_e|\mathcal{X}_e}(0,x_e)} \cdot \prod_{e=(v_i,v_j),v_j\in B} \frac{p_{\mathcal{Y}_e|\mathcal{X}_e}(0,x_e)}{p_{\mathcal{Y}_e|\mathcal{X}_e}(1,x_e)}. \tag{7.8}$$

To produce an uncertainty measure for each node in the graph, we apply the logistic function on Equation (7.7)

$$\text{uncertainty} = \frac{1}{1+\exp(-\gamma_i)} \tag{7.9}$$

as it is the inverse of the logit function used in the cost computation in Equation (7.4). In the following, we show the expansion of Equation (7.9) by injecting the corresponding values for γ_i. To determine γ_i, the minimum change in the cost is computed among all the possible transitions between the clusters for each node $v_i \in V$ (refer to Figure 7.3 for an example), such that

$$\text{uncertainty} = \frac{1}{1+\exp(-\min_B \gamma_B)} \tag{7.10}$$

where

$$\gamma_B = \sum_{v_j\in\mathcal{N}_{E'}(v_i)\cap A} c_{(v_i,v_j)} - \sum_{v_j\in\mathcal{N}_{E'}(v_i)\cap B} c_{(v_i,v_j)}. \tag{7.11}$$

The resulting uncertainty measure is thus of the form

$$\text{uncertainty} = \frac{1}{1+\exp\left(\sum_{v_j \in \mathcal{N}_{E'}(v_i)\cap B} c_{(v_i,v_j)} - \sum_{v_j \in \mathcal{N}_{E'}(v_i)\cap A} c_{(v_i,v_j)}\right)}. \tag{7.12}$$

According to the Bayesian model and the findings in Andres et al. [3], the costs $c_{(v_i,v_j)}$ for each $e := (v_i, v_j) \in E$ are computed via Equation (7.4),

$$\forall e \in E: \quad c_e = \log \frac{p_{\mathcal{Y}_e|\mathcal{X}_e}(0,x_e)}{p_{\mathcal{Y}_e|\mathcal{X}_e}(1,x_e)} + \log \frac{1-\beta}{\beta} \quad . \tag{7.13}$$

For simplicity and to be compatible with our experiments, we set the value of $\beta = 0.5$ (i.e. we assume an unbiased decomposition), which makes $\log \frac{1-\beta}{\beta} = 0$.

We insert $c(v_i, v_j) = \log \frac{p_{\mathcal{Y}_e|\mathcal{X}_e}(0,x_e)}{p_{\mathcal{Y}_e|\mathcal{X}_e}(1,x_e)}$, $(e = (v_i, v_j))$ into Equation (7.12) and denote $v_j \in \mathcal{N}_{E'}(v_i) \cap B$ by e,B where $e = (v_i, v_j), v_j \in B$ and $v_j \in \mathcal{N}_{E'}(v_i) \cap A$ by e,A where $e = (v_i, v_j), v_j \in A$, and get

$$\text{uncertainty} = \frac{1}{1+\exp\left(\sum_{e,B} \log \frac{p_{\mathcal{Y}_e|\mathcal{X}_e}(0,x_e)}{p_{\mathcal{Y}_e|\mathcal{X}_e}(1,x_e)} - \sum_{e,A} \log \frac{p_{\mathcal{Y}_e|\mathcal{X}_e}(0,x_e)}{p_{\mathcal{Y}_e|\mathcal{X}_e}(1,x_e)}\right)}$$

$$= \frac{1}{1+\exp\left(\log \prod_{e,B} \frac{p_{\mathcal{Y}_e|\mathcal{X}_e}(0,x_e)}{p_{\mathcal{Y}_e|\mathcal{X}_e}(1,x_e)} - \log \prod_{e,A} \frac{p_{\mathcal{Y}_e|\mathcal{X}_e}(0,x_e)}{p_{\mathcal{Y}_e|\mathcal{X}_e}(1,x_e)}\right)}$$

$$= \frac{1}{1+\exp\left(\log \prod_{e,B} \frac{p_{\mathcal{Y}_e|\mathcal{X}_e}(0,x_e)}{p_{\mathcal{Y}_e|\mathcal{X}_e}(1,x_e)} + \log \prod_{e,A} \frac{p_{\mathcal{Y}_e|\mathcal{X}_e}(1,x_e)}{p_{\mathcal{Y}_e|\mathcal{X}_e}(0,x_e)}\right)}$$

$$
\begin{aligned}
&= \frac{1}{1+\exp\left(\log\left(\prod_{e,B}\frac{p_{\mathcal{Y}_e|\mathcal{X}_e}(0,x_e)}{p_{\mathcal{Y}_e|\mathcal{X}_e}(1,x_e)}\cdot\prod_{e,A}\frac{p_{\mathcal{Y}_e|\mathcal{X}_e}(1,x_e)}{p_{\mathcal{Y}_e|\mathcal{X}_e}(0,x_e)}\right)\right)} \\
&= \frac{1}{1+\left(\prod_{e,B}\frac{p_{\mathcal{Y}_e|\mathcal{X}_e}(0,x_e)}{p_{\mathcal{Y}_e|\mathcal{X}_e}(1,x_e)}\cdot\prod_{e,A}\frac{p_{\mathcal{Y}_e|\mathcal{X}_e}(1,x_e)}{p_{\mathcal{Y}_e|\mathcal{X}_e}(0,x_e)}\right)} \\
&= \frac{1}{1+\left(\frac{\prod_{e,B}p_{\mathcal{Y}_e|\mathcal{X}_e}(0,x_e)}{\prod_{e,B}p_{\mathcal{Y}_e|\mathcal{X}_e}(1,x_e)}\cdot\frac{\prod_{e,A}p_{\mathcal{Y}_e|\mathcal{X}_e}(1,x_e)}{\prod_{e,A}p_{\mathcal{Y}_e|\mathcal{X}_e}(0,x_e)}\right)}
\end{aligned}
\tag{7.14}
$$

Note that in the denominator, we have exactly 1 + the term from Equation (7.8). With a slight reformulation, we get

$$
= \frac{\prod_{e,A}p_{\mathcal{Y}_e|\mathcal{X}_e}(0,x_e)\cdot\prod_{e,B}p_{\mathcal{Y}_e|\mathcal{X}_e}(1,x_e)}{\prod_{e,A}p_{\mathcal{Y}_e|\mathcal{X}_e}(0,x_e)\cdot\prod_{e,B}p_{\mathcal{Y}_e|\mathcal{X}_e}(1,x_e)+\prod_{e,B}p_{\mathcal{Y}_e|\mathcal{X}_e}(0,x_e)\cdot\prod_{e,A}p_{\mathcal{Y}_e|\mathcal{X}_e}(1,x_e)} \tag{7.15}
$$

or by a simplified notation $p_{\mathcal{Y}_e|\mathcal{X}_e}(1,x_e) =: p_c$ for *cut* probabilities and $p_{\mathcal{Y}_e|\mathcal{X}_e}(0,x_e) =: p_j$ for *join* probabilities.

$$
\frac{\prod_{e,A}p_{\mathcal{Y}_e|\mathcal{X}_e}(0,x_e)\cdot\prod_{e,B}p_{\mathcal{Y}_e|\mathcal{X}_e}(1,x_e)}{\prod_{e,A}p_{\mathcal{Y}_e|\mathcal{X}_e}(0,x_e)\cdot\prod_{e,B}p_{\mathcal{Y}_e|\mathcal{X}_e}(1,x_e)+\prod_{e,B}p_{\mathcal{Y}_e|\mathcal{X}_e}(0,x_e)\cdot\prod_{e,A}p_{\mathcal{Y}_e|\mathcal{X}_e}(1,x_e)} \tag{7.16}
$$

where the nominator is the product of the local posterior probabilities of the observed solution A at node v_i (compare Equation (7.1) for $p_{\mathcal{Y}_e}$ const.) and is proportional to the posterior of the chosen node label if v_i has at most two labels in the local neighborhood. The denominator sums trivially to one in the case of $|\mathcal{N}_{E'}(v_i)| = 1$. For the more common case of $|\mathcal{N}_{E'}(v_i)| \geq 1$, expression (7.16) can be interpreted as a "calibrated" probability. It normalizes the posterior probability of the MAP solution with the sum of posteriors of this solution and the second most likely one. Intuitively, if all solutions have a low absolute posterior probability, the relatively best one can still have a high certainty.

7.4 Evaluation and Results

We evaluate the proposed uncertainty measure in two common application scenarios of minimum cost multicuts: motion segmentation and image segmentation. For motion segmentation, we conduct experiments on the datasets FBMS59, Ochs et al. [123], and DAVIS_{2016}, Perazzi et al. [128], (see Section 2.1). On both datasets, we compute point trajectories, as well as the segmentation using the method from Keuper et al. [88], which computes edges between point trajectories as pseudo-probabilities from a simple motion model (refer to Section 2.4). Additionally, we highlight the potential benefit of a robust uncertainty measure: It not only allows us to produce several hypotheses of solutions. We can also employ the estimated uncertainties in our densification framework in [81] (see Chapter 4) to compute improved, dense motion segmentations from sparse ones. Last, we evaluate the proposed uncertainty measure on the image segmentation task posed by the BSDS500 dataset by Arbeláez et al. [5], employing the LMP instances from Keuper et al. [89].

Parameter Setting. In the MP and LMP, the value β (as in Equation (7.4)) is set to 0.5 (unbiased) to be comparable with the baseline methods in the evaluated datasets. The LMP on images requires the value τ to be set, corresponding to the radius until which to insert lifted edges. We set $\tau = 20$ as in Keuper et al. [89].

Metrics. In both applications, we evaluate our uncertainty measure based on the variation of information (VI), Meilă [115], and Rand index (RI) [115]. We study the effect of measured uncertainties by means of sparsification. The most uncertain nodes are removed subsequently, and in each step, VI and RI are measured. Ideally, VI should drop, and RI should increase monotonically as more uncertain nodes are removed. In Meilă [115] it is shown that VI is a more reliable indicator than the RI concerning the density of the results since the RI depends heavily on the granularity of the decomposition. Specifically, the more pixels or motion trajectories are removed during sparsification, the less reliable the RI becomes, which is why both metrics are reported in all our experiments. It is essential to note that the high uncertainty of a node indicates that the label assigned to the node after the termination of the solver tends to flip. Therefore, removing those nodes from the decomposition should improve the quality of the decomposition in terms of VI and RI. We compared our approach with the proposed baseline approaches.

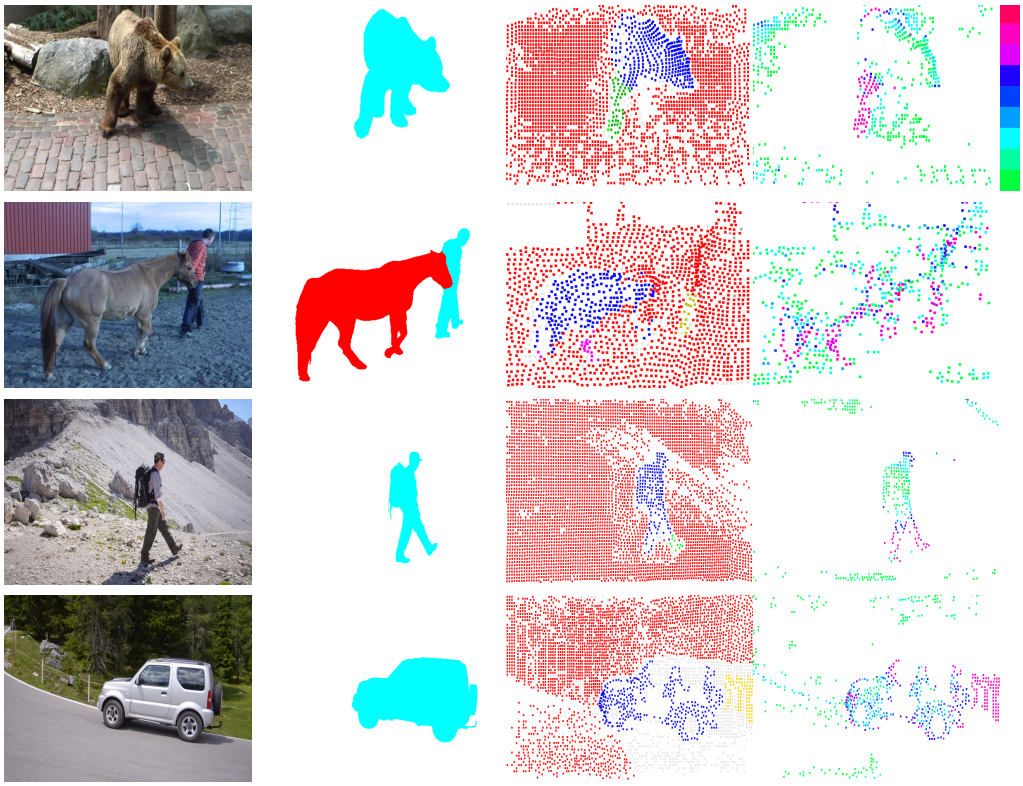
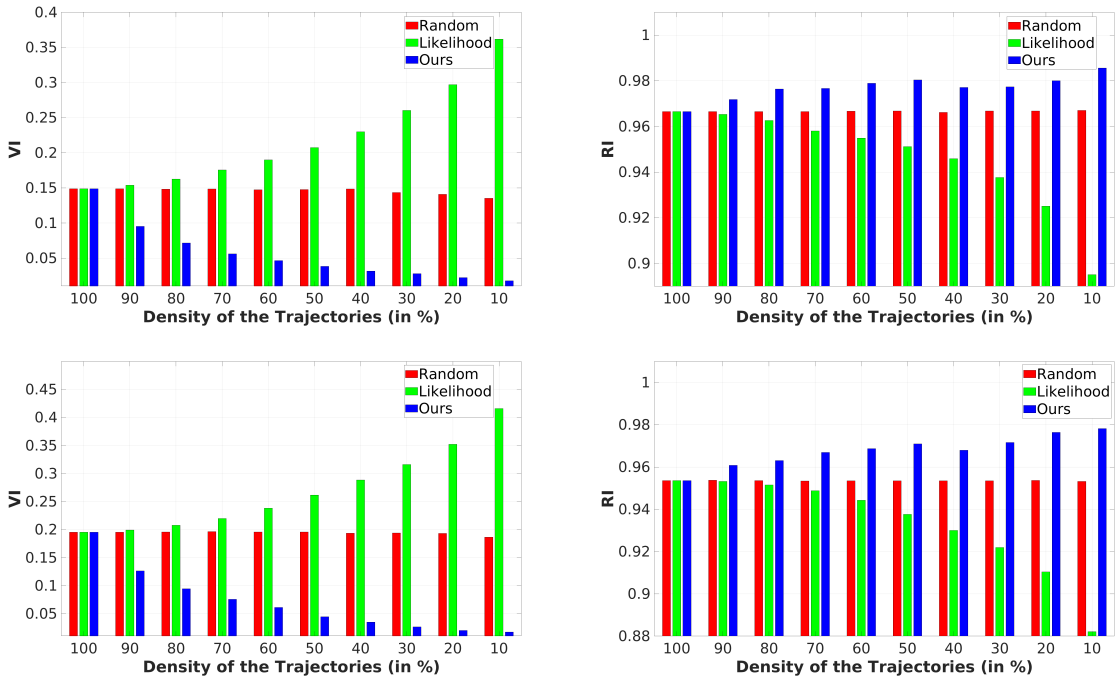


Fig. 7.4 Uncertainty measures on point trajectories for the two sequences from FBMS59 (**first two rows**) and DAVIS$_{2016}$ (**last two rows**) are shown. In each row, from left to right, we provide an image, its ground-truth, the segmentation, and our uncertainty estimation. The uncertainty values are discretized and color-coded for visualization purposes. White areas correspond to the trajectories with high certainty. The uncertainty on thin, articulated object parts is high.

7.4.1 Motion Segmentation Results

To study the proposed uncertainty measure, we first compute minimum cost multicuts on the motion segmentation instances of FBMS59 and DAVIS$_{2016}$. On these decompositions, we compute the proposed uncertainties. In Figure 7.4, we depict example images from sequences of FBMS59 and DAVIS$_{2016}$, their ground-truth segmentation, trajectory segmentation and the proposed uncertainty measures. The uncertainty measures are in the range of 0 (lowest uncertainty) to 1 (highest uncertainty). As can be seen, the sparse segmentations show good overall accuracy but tend to have issues on dis-occlusion areas, for example, behind the driving car in the last row, and on object parts under articulated motion, such as the legs of the horse in the second row and the legs of the person in the third row. In all these regions, the estimated uncertainty is high. Next, we assess the proposed uncertainties in terms of sparsification plots as done in Ilg et al. [64], i.e., by subsequently removing nodes in the order

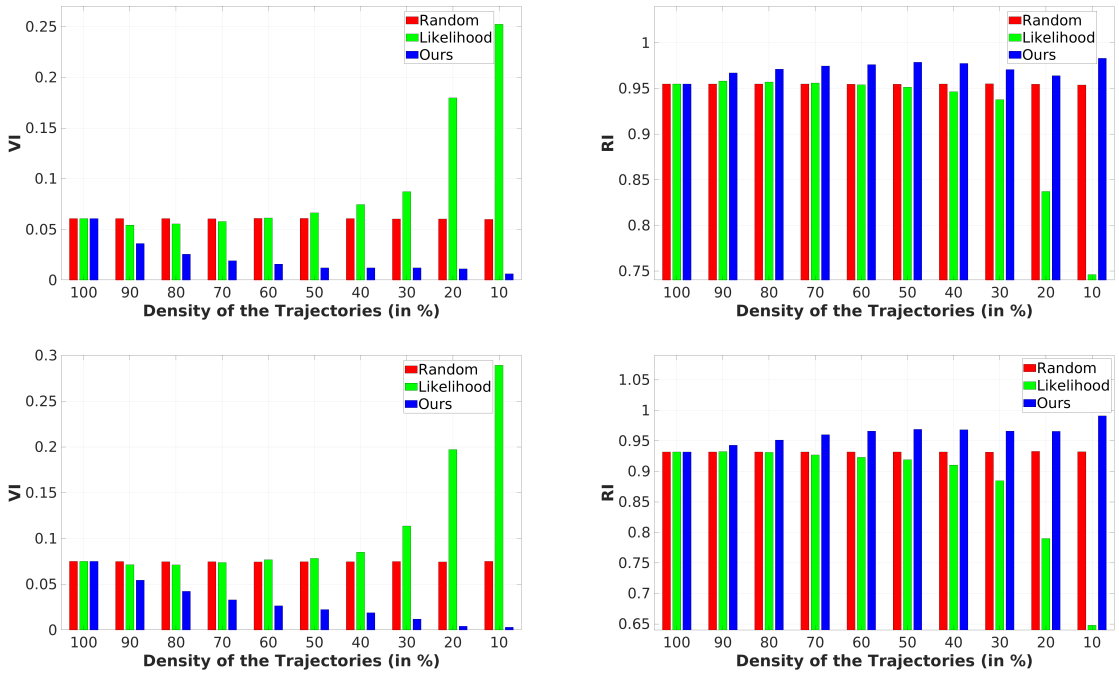


VI (lower is better) RI (higher is better)

Fig. 7.5 Study on motion trajectory uncertainties on VI (**left**) and RI (**right**) on the train set (**top row**) and on the test set (**bottom row**) of FBMS59. The metrics improve by removing trajectories according to the proposed uncertainty measure. Notice that removing uncertain trajectories according to the likelihood baseline deteriorates both VI and RI.

of their decreasing uncertainty from the solution. We compare our results to the *Likelihood* baseline defined in Equation (7.5) as well as to a random baseline. In the random baseline, the trajectories are removed subsequently at *random*. In this case, neither improvement nor decay of the segmentation metric should be expected.

FBMS59. The Variation of Information (VI) and the Rand index (RI) are provided in Figure 7.5 for the FBMS59-train and FBMS59-test. The most uncertain trajectories are removed until reaching 10% of the density of the trajectories. Interestingly, removing highly uncertain trajectories accounts for the reduction in VI (lower is better) and improves the RI (higher is better), indicating the effectiveness of the proposed uncertainty measure. The VI continuously improves as more uncertain trajectories are removed. Yet, the RI becomes unsteady when getting closer to the lowest density. Recall that the RI depends on the granularity of the solution (see Meilă [115]), so this behavior is to be expected as more and more ground-truth segments are not considered in the sparse solution.



VI (lower is better) RI (higher is better)

Fig. 7.6 Study on the motion trajectory uncertainties on VI (**left**) and RI (**right**) on train set (**top row**) and validation set (**bottom row**) of DAVIS$_{2016}$ [128]. Our results improve significantly over the baselines.

DAVIS$_{2016}$. The evaluation on the DAVIS$_{2016}$ train and validation sets in Figure 7.6 indicates a similar behaviour as seen on FBMS59 and shows the effect of our approach. Notice that our approach provides better uncertainty measures than the baselines (*Random Baseline* and *Likelihood Approach*, refer to Section 7.3.2. Again, the likelihood baseline even shows an increase in VI during sparsification, proving the importance of using the denominator in Equation (7.16).

7.4.2 Multimodal Motion Segmentation

In the following, we briefly sketch how the proposed approach can be leveraged to generate multiple, likely solutions. After the termination of the heuristic solver, the nodes in the graph $G = (V, E)$ are moved between different clusters to compute the cost differences in Equation (7.6). Given the N clusters, the node from cluster A is moved to other $N-1$ clusters, and the cost change is computed. Therefore, a vector with the number of different partitions minus one $(N-1)$ is computed for each node. This leads to $\frac{N \times (N-1)}{2}$ vectors, each with

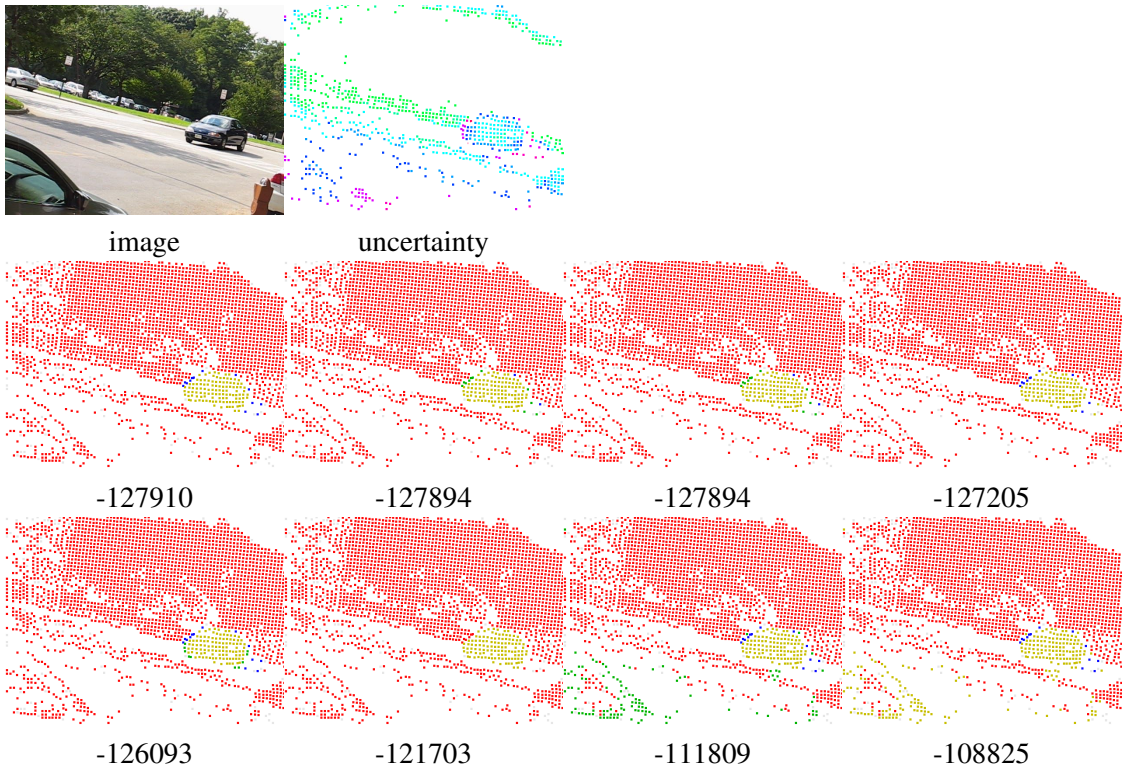


Fig. 7.7 Visualization of eight different likely solutions and their energies (refer to Equation (2.9) in Chapter 2, lower is better), as they can be generated by the proposed method. The different solution candidates vary mainly along object boundaries. The best segmentation with respect to the ground-truth corresponds to the second image (from left) in the last row.

$|V|$ values. To produce multiple segmentations, such vectors are ordered based on their magnitude, i.e., their associated energy increase. Each such vector holds the costs for moving nodes from cluster A to cluster B. The label of the node v is changed from cluster A to cluster B if $\Delta cost(A,B)_v \approx 0$. With this approach, it is possible to generate n best clusterings, which differ in the labels of uncertain points. In Figure 7.7, we visualize an example of eight such potential solutions and their energies. The variance is strongest on the object boundaries and the car in the foreground. It coincides with the estimated uncertainties. In Figure 7.8, we allow the evaluation script of FBMS59 to choose the best among the n most likely solutions for $n \in \{1,...,10\}$. By taking more solutions into account, the F-measure improves.

7.4.3 Densified Motion Segmentation

To create actual pixel-level segmentations from sparse segmentations a separate model needs to be learned. For example Ochs et al. [123] proposes a variational model while in [81] we

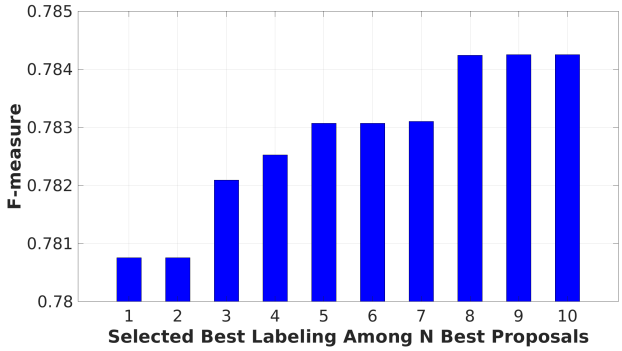


Fig. 7.8 F-measure on the train set of FBMS59 when selecting n best segmentation proposals. The F-measure improves as the number of segmentation candidates increases.

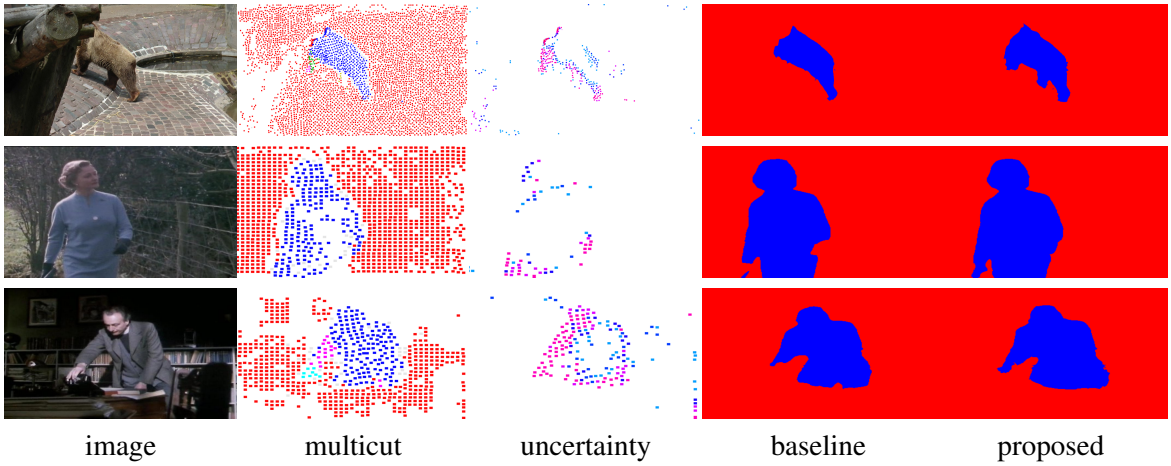


Fig. 7.9 Densification of sparse segmentations using uncertainties. We compare the result from our densification model in [81] (baseline) with the proposed method, which uses uncertainties in the model training. Improvements can be observed especially on thin structures such as limbs.

Table 7.1 Densification of the sparse trajectory segmentations on the FBMS59 train set. We compare our model [81] in Chapter 4 to the variant trained using uncertainties.

	Precision	Recall	F-measure
[81]	89.35	67.67	77.01
Ours (uncertainty-aware)	88.17	68.96	77.40

employ a U-Net [135] based model, which is trained in a self-supervised manner, using the sparsely segmented trajectories as labels, (see Chapter 4). In the following, we employ our uncertainties to enhance the self-supervised training signal in this model in the FBMB59

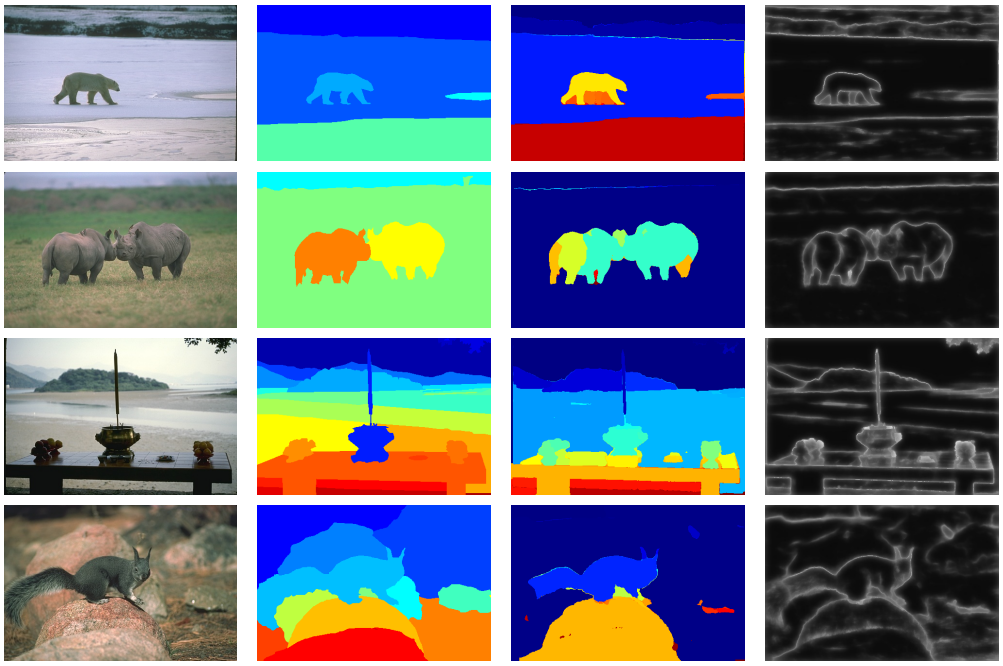


Fig. 7.10 Exemplary images and segmentation uncertainties on the BSDS500 [5] dataset. In each row, from left to right, the original images, ground-truth segmentation, the resulting minimum cost lifted multicut segmentation, and the proposed uncertainties are given. Bright areas in the uncertainty images represent uncertain pixels.

dataset. As a pre-processing, our denisification model in [81] remove small segments from the training data such that the remaining uncertain points are mostly on thin, articulated object parts. For each segment A in the decomposition of graph G, the mean uncertainty is computed as μ_A. The highly uncertain nodes, where $u_v > \mu_A$, in the highly uncertain segments $\mu_A > \phi$ are assumed to be hard examples for the network and are therefore used more often than other points during training (with factor 1000). In Table 7.1, we report the Precision, Recall, and F-measure of the proposed training and the original model in [81] which is explained in Chapter 4. Using this simple trick, the F-Measure can be improved by 0.4%. Qualitative results are provided in Figure 7.9.

7.4.4 Image Decomposition

As the last application, we evaluate the proposed uncertainty measure on the image segmentation task in the context of minimum cost lifted multicuts. The evaluation is computed on the BSDS500 dataset using the problem instances and solutions from Keuper et al. [89]. Figure 7.10 visualizes the node (pixel) level uncertainty on several example images from the

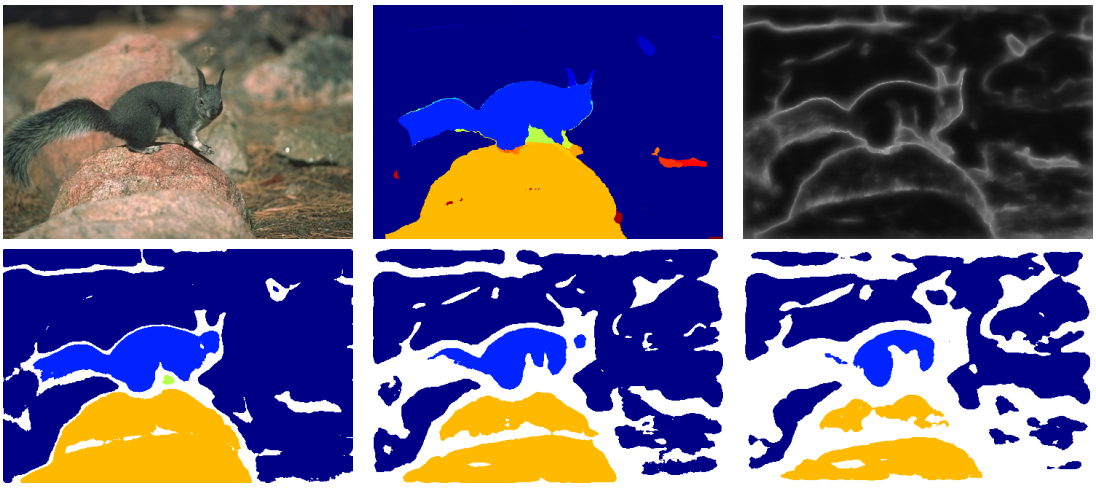
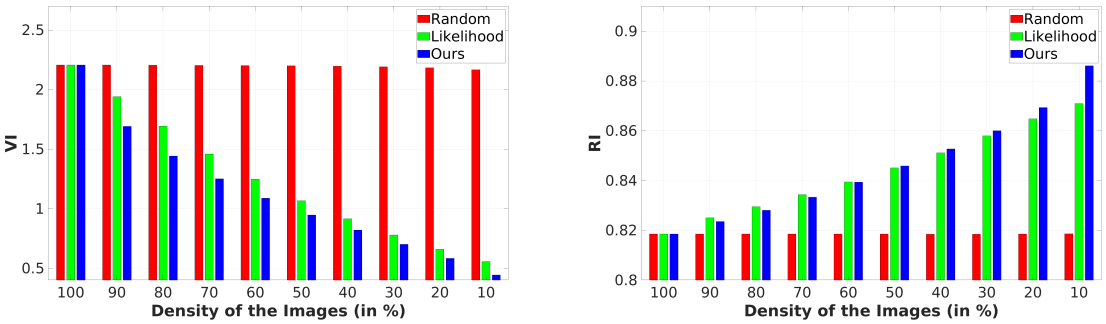


Fig. 7.11 Visualization of removing uncertain pixels in BSDS500 images [5]. Notice that removing uncertain pixels corresponds to removing pixels along the object boundaries. The original image (**left**), its multicut solution (**middle**), and the uncertainty measure (**right**) based on our model are shown in the first row. The second row visualizes (from left to right) removing 10, 30, and 50 % of the most uncertain pixels.

BSDS500 [5] dataset (see Section 2.1.3). As expected, the node level uncertainty is higher along the object boundaries, where label changes are likely to happen.

Removing the most uncertain pixels corresponds to removing object boundaries, as depicted in Figure 7.11. As the segmentation becomes sparser, entire object parts, such as the squirrel's tail, will be removed from the solution. In Figure 7.12, the VI and RI metrics are provided concerning the pixel densities. As expected, removing highly uncertain pixels increases the RI and decreases VI. Thus, an improvement can be observed in both metrics. Again, we compare our approach to the probability-driven baseline and a random baseline. Unlike before, the probability-driven baseline performs reasonably in this setting. Yet, the proposed measure can achieve a faster decrease in VI and a higher increase in RI, thus showing more robust behavior. The relatively good performance of the baseline in comparison to its results on motion segmentation can be explained by the way local cut probabilities are computed in this setting. Specifically, they are derived by Keuper et al. [89] from edge maps and thus have consistently low values in homogeneous regions. Therefore, the normalization by the denominator in the proposed measure (refer to Equation (7.16)) becomes less important.



VI (lower is better) RI (higher is better)

Fig. 7.12 Sparsification analysis in VI (**left**) and RI (**right**) on the BSDS500 [5] test data. The proposed method shows a faster decrease in VI than the baseline and reaches a higher RI.

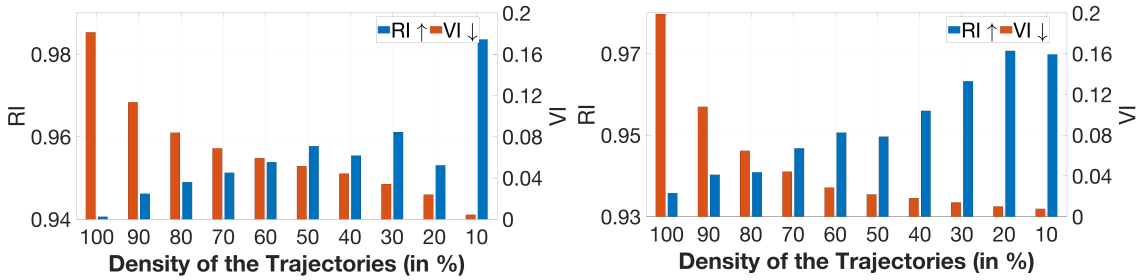


Fig. 7.13 Study on the trajectory uncertainty on the GAEC [89] solver is provided. The experiment relates to the Variation of Information (VI) and Rand Index (RI) on the train (**left**) and test (**right**) set of FBMS59 [123].

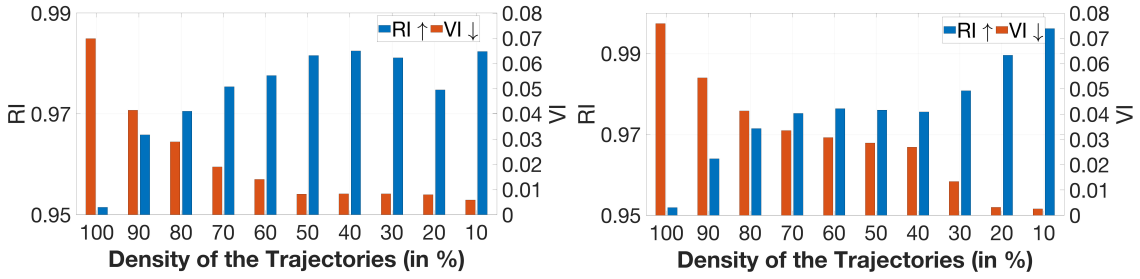


Fig. 7.14 Study on the trajectory uncertainty on the GAEC [89] solver is provided. The experiment relates to the Variation of Information (VI) and Rand Index (RI) on the train (**left**) and validation (**right**) set of DAVIS_{2016} [128].

7.4.5 Uncertainty on Minimum Cost Multicut Solutions from GAEC

In Figures 7.13 and 7.14, we provide an additional evaluation of the proposed uncertainty measure in the motion segmentation setting. Specifically, we compute solutions for the motion segmentation problem instances on the FBMS59 and DAVIS_{2016} datasets. While we evaluated using the widely employed high-quality solutions from the KLj heuristic, here we additionally assess uncertainties on a faster, lower quality solver, GAEC [89]. It can be seen that the sparsification plots behave as expected. The VI decreases as the segmentation becomes sparser and the RI increases. However, it can be seen that for the poorer segmentation results, the RI does not increase as monotonically as for KLj. Specifically, when considering the high sparsity regime, the RI metric becomes brittle, indicating that entire labels might have been removed from the solution.

7.5 Conclusion

The minimum cost *lifted* multicut problem has been widely studied in different computer vision and data analysis applications such as motion segmentation and image decomposition. Due to the probabilistic behavior of the multicut problem, the final decomposition of the nodes in the proposed graph can be seen as relaxed decisions rather than hard decisions on label assignment to the nodes of the graph. We study this probabilistic behavior and provide an informative uncertainty measure in the node uncertainties. We showed that removing uncertain nodes according to the proposed measure improves the variation of information (VI) and the Rand index (RI) in two applications of motion segmentation and image decomposition. Further, we showed the application of such uncertainties to train a self-supervised model for motion segmentation. The proposed uncertainty measure can be combined with any minimum cost multicut-based formulation.

Chapter 8

Future Work and Conclusion

In this Chapter, we discuss the limitations of our proposed approaches. We then illustrate possible future works which could improve the results. Finally, we finish this dissertation with a conclusion.

8.1 Limitations

In the video instance segmentation task, which is discussed in Chapter 3, the *optical flow* and *edge maps* are used to track the segmentation of the objects to the subsequent frames and provide the instance segmentation via the variational framework. The performance of the approach depends on the quality of these underlying information.

In this dissertation, the clustering tasks are projected into the graph structure and decomposed via the minimum cost *lifted* multicut (LMP) framework proposed in Keuper et al. [89]. Further, in the work described in Section 6.7.1, we have proposed an adaptive cooling-based Conditional Random Field (CRF) formulation in the end-to-end multicut graph decomposition to enforce more binary edge weights. This model injects the set of multicut constraints into the objective function. The generation of such constraints, more specifically, the valid and invalid cycle inequalities, is produced beforehand to optimize the model over such constraints. Moreover, this model depends on the post-processing of the produced edge maps to get the Intervening Contour Cues (ICC), Leung and Malik [100]. Such data are processed offline and before processing through the graph decomposition model.

In the motion segmentation task, in Chapters 4 and 5, the sparse motion segmentation is proposed. To generate the dense motion segmentation, the sparsely segmented results need to pass through a post-processing stage. Although, in principle, dense trajectories can be generated by the motion segmentation algorithm, the clustering algorithm does not scale linearly with the number of trajectories, and the computational cost explodes. In the

higher-order motion segmentation approach, in Chapter 5, the combination of the second and third-order costs is used to handle the complex motion patterns. The better results can be generated using higher order edges, considering such edges make the result generation more intense regarding the time and memory consumption.

8.2 Future Work

As for any proposed methods, there is always some room for improvement; here, we describe future works that ideally could enhance either the performance or the quality of the final results. In the case of video instance segmentation, where the objects are tracked from the first frame mask [84], explained in Chapter 3, instead of the variational framework, one approach for tracking the segments is the usage of optical flow information and creating the multicut problem where the nodes correspond to the tracked segment labels through the frames; similar to the motion segmentation except that instead of motion trajectories the point trajectories are used. This will be similar to the work of Keuper et al. [89], while instead of an image (one frame) the sequence of frames is used with the usage of optical flow information.

The end-to-end trainable model proposed by Song et al. [143] for graph decomposition via injecting the multicut constraints into the objective function of the real-world multi-person pose estimation problem is addressed. We presented an adaptive CRF that allows us to progressively consider more violated constraints and, consequently, to issue solutions with higher validity in work led by Steffen Jung [74]. Therefore, we confirm the efficiency of our approach in the image segmentation task. The model requires the offline processing of the edge maps with the ICC [100] for production of the set of cycle inequality constraints (see Section 6.7.1). One possibility is to generate such constraints online without pre-processing the edge maps. This model is proposed for the multicut problem; one option is to extend this to the *lifted* multicut problem, which requires more constraints to be considered. Moreover, the improvement of this model can lead to the end-to-end trainable model for multi-label dense motion segmentation. The graph production of the multicut problem is generated offline as well; for instance, in the work of motion segmentation, Chapters 4 and 5. This means the whole sequence of frames must be seen before providing the motion segmentation. This makes the solution generation in an offline manner.

We proposed Gated Recurrent Unit (GRU) based model to provide the affinity costs between the pair of trajectories in [81], see Chapter 4, for the edges on the produced graph. We only used the position of the pixels to provide the similarities between the trajectories. One possibility is to include appearance features by considering the patches around each

sub-pixel position. Moreover, vision transformers like Touvron et al. [152] can be replaced with the GRU model to get the results faster and more accurately.

8.3 Conclusion

The clustering problem, its frameworks, and its application in different fields such as image, mesh data, video, and motion segmentation are studied in this dissertation. We used a graph-based framework to project the applications to the nodes and edges on the graph. One well-known framework that does not require prior knowledge, such as the number of clusters, and does not favor balanced or specific properties of the clusters, is the minimum cost (lifted) multicut framework. Due to the NP-hard nature of the problem, different heuristics are proposed; in this dissertation, we proposed two variants of a heuristic solver (primal feasible heuristic), which greedily generate solutions within a bounded amount of time and evaluated on image and mesh segmentation benchmarks. Relying on the probabilistic formulation of minimum cost multicuts, we proposed a measure for the uncertainties of the decisions made during the optimization. Our approaches ranged from the applications of the multicut framework, such as in mesh data, image, motion, and video instance segmentation, to the methods for decomposing the problem instances (multicut graphs) and the uncertainty estimation.

Bibliography

[1] Alush, A. and Goldberger, J. (2012). Ensemble segmentation using efficient integer linear programming. *TPAMI*. 30

[2] Andres, B., Kappes, J. H., Beier, T., Köthe, U., and Hamprecht, F. A. (2011). Probabilistic image segmentation with closedness constraints. In *ICCV*. 26, 30, 107

[3] Andres, B., Kröger, T., Briggman, K. L., Denk, W., Korogod, N., Knott, G., Köthe, U., and Hamprecht, F. A. (2012). Globally optimal closed-surface segmentation for connectomics. In *ECCV*. 3, 9, 26, 28, 30, 55, 68, 105, 106, 108, 112

[4] Andres, B., Yarkony, J., Manjunath, B. S., Kirchhoff, S., Turetken, E., Fowlkes, C., and Pfister, H. (2013). Segmenting planar superpixel adjacency graphs w.r.t. non-planar superpixel affinity graphs. In *EMMCVPR*. 30

[5] Arbeláez, P., Maire, M., Fowlkes, C., and Malik, J. (2011). Contour detection and hierarchical image segmentation. *TPAMI*. xviii, xxi, xxiii, xxiv, xxvii, 16, 17, 30, 89, 90, 91, 95, 96, 97, 99, 103, 105, 106, 107, 114, 120, 121, 122

[6] Arbelaez, P., Pont-Tuset, J., Barron, J., Marques, F., and Malik, J. (2014). Multiscale combinatorial grouping. In *CVPR*. xxvii, 99

[7] Badrinarayanan, V., Galasso, F., and Cipolla, R. (2010). Label propagation in video sequences. In *CVPR*. 20

[8] Bae, E., Lellmann, J., and Tai, X.-C. (2013). Convex relaxations for a generalized Chan-Vese model. In *EMMCVPR*. 3, 21, 37

[9] Bagon, S. and Galun, M. (2011). Large scale correlation clustering optimization. *CoRR*. 30

[10] Bailer, C., Taetz, B., and Stricker, D. (2015). Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation. In *ICCV*. 45

[11] Bailoni, A., Pape, C., Wolf, S., Beier, T., Kreshuk, A., and Hamprecht, F. A. (2019). A generalized framework for agglomerative clustering of signed graphs applied to instance segmentation. *CoRR*. 57

[12] Bansal, N., Blum, A., and Chawla, S. (2004). Correlation clustering. *Machine Learning*. 26, 27, 57, 68, 105

[13] Bao, L., Wu, B., and Liu, W. (2018). CNN in MRF: Video object segmentation via inference in a CNN-based higher-order spatio-temporal MRF. In *CVPR*. xix, 21, 38, 46, 47, 48

[14] Bardes, A., Ponce, J., and LeCun, Y. (2021). Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *CoRR*. 7

[15] Beier, T., Andres, B., Köthe, U., and Hamprecht, F. A. (2016). An efficient fusion move algorithm for the minimum cost lifted multicut problem. In *ECCV*. xxvii, 27, 57, 90, 91, 100, 103

[16] Beier, T., Hamprecht, F. A., and Kappes, J. H. (2015). Fusion moves for correlation clustering. In *CVPR*. 26, 29, 30, 105

[17] Beier, T., Kroeger, T., Kappes, J. H., Köthe, U., and Hamprecht, F. A. (2014). Cut, glue, & cut: A fast, approximate solver for multicut partitioning. In *CVPR*. 26, 29, 30

[18] Beier, T., Pape, C., Rahaman, N., Prange, T., Stuart, B., Bock, D., Cardona, A., Knott, G. W., Plaza, S. M., Scheffer, L. K., Ullrich, K., Kreshuk, A., and Hamprecht, F. A. (2017). Multicut brings automated neurite segmentation closer to human performance. *Nature Methods*. 27, 30, 90, 95, 99, 103

[19] Bhattacharyya, A., Fritz, M., and Schiele, B. (2018). Long-term on-board prediction of people in traffic scenes under uncertainty. In *CVPR*. 54

[20] Bideau, P. and Learned-Miller, E. (2016). A detailed rubric for motion segmentation. 79, 82

[21] Bideau, P. and Learned-Miller, E. G. (2016). It's moving! A probabilistic model for causal motion segmentation in moving camera videos. *CoRR*. 23

[22] Bideau, P., Menon, R. R., and Learned-Miller, E. (2018a). Moa-net: Self-supervised motion segmentation. In *ECCV workshop*. 23

[23] Bideau, P., RoyChowdhury, A., Menon, R. R., and Learned-Miller, E. (2018b). The best of both worlds: Combining cnns and geometric constraints for hierarchical motion segmentation. In *CVPR*. xxvi, 23, 79, 82, 83

[24] Brox, T., Bregler, C., and Malik, J. (2009). Large displacement optical flow. In *CVPR*. xxvi, 60, 63

[25] Brox, T. and Malik, J. (2010). Object segmentation by long term analysis of point trajectories. In *ECCV*. 2, 15, 25, 52, 55, 59, 69, 74

[26] Brox, T. and Malik, J. (2011). Large displacement optical flow: descriptor matching in variational motion estimation. *TPAMI*. xx, 74, 81, 82, 85, 86

[27] Cardona, A., Saalfeld, S., Preibisch, S., Schmid, B., Cheng, A., Pulokas, J., Tomancak, P., and Hartenstein, V. (2010). An integrated micro- and macroarchitectural analysis of the drosophila brain by computer-assisted serial section electron microscopy. *PLOS Biology*. xxvii, 18, 21, 90, 95, 99, 100

[28] Carreras, I. A., Turaga, S. C., Berger, D. R., San, D. C., Giusti, A., Gambardella, L. M., Schmidhuber, J., Laptev, D., Dwivedi, S., Buhmann, J. M., Liu, T., Seyedhosseini, M., Tasdizen, T., Kamentsky, L., Burget, R., Uher, V., Tan, X., Sun, C., Pham, T. D., Bas, E., Uzunbas, M. G., Albert, C., Schindelin, J., and Seung, H. S. (2015). Crowdsourcing the creation of image segmentation algorithms for connectomics. *Frontiers in Neuroanatomy*. xxvii, 18, 21, 90, 95, 99, 100

[29] Chambolle, A., Cremers, D., and Pock, T. (2012). A convex approach to minimal partitions. *SIAM Journal on Applied Mathematics*. 40, 44

[30] Chen, L., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2018). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*. 23

[31] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2015). Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*. 59

[32] Chen, X., Golovinskiy, A., and Funkhouser, T. (2009). A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics (Proc. SIGGRAPH)*. xxvii, 17, 90, 95, 97, 100

[33] Cheng, J., Tsai, Y.-H., Hung, W.-C., Wang, S., and Yang, M.-H. (2018). Fast and accurate online video object segmentation via tracking parts. In *CVPR*. 48

[34] Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 23

[35] Chopra, S., Hadsell, R., and LeCun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification. In *CVPR*. 54, 56

[36] Chopra, S. and Rao, M. R. (1993). The partition problem. *Mathematical Programming*. 3, 5, 25, 26, 27, 28, 68

[37] Cremers, D. (2003). A variational framework for image segmentation combining motion estimation and shape regularization. In *CVPR*. 22, 23

[38] Demaine, E. D., Emanuel, D., Fiat, A., and Immorlica, N. (2006). Correlation clustering in general weighted graphs. *Theoretical Computer Science*. 26, 27, 28

[39] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *CVPR*. 22, 53

[40] Deza, M. M., Laurent, M., and Weismantel, R. (1997). Geometry of cuts and metrics. *Mathematical Methods of Operations Research-ZOR*. 3, 5, 26, 27

[41] Dollár, P. and Zitnick, C. L. (2013). Structured forests for fast edge detection. In *ICCV*. xxvii, 20, 37, 42, 44, 99

[42] Dosovitskiy, A., Fischer, P., Ilg, E., Häusser, P., Hazırbaş, C., Golkov, V., v.d. Smagt, P., Cremers, D., and Brox, T. (2015). Flownet: Learning optical flow with convolutional networks. In *ICCV*. 19, 22, 53

[43] Dosovitskiy, A., Springenberg, J. T., Riedmiller, M., and Brox, T. (2014). Discriminative unsupervised feature learning with convolutional neural networks. In *Advances in Neural Information Processing Systems*. 7

[44] Drayer, B. and Brox, T. (2016). Object detection, tracking, and motion segmentation for object-level video segmentation. *CoRR*. 20

[45] Elhamifar, E. and Vidal, R. (2013). Sparse subspace clustering. In *ICCV*. 32

[46] Everingham, M., Gool, L. V., Williams, C. K. I., Winn, J., and Zisserman, A. (2012). The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. 23

[47] Fan, Q., Zhong, F., Lischinski, D., Cohen-Or, D., and Chen, B. (2015). JumpCut: Non-successive mask transfer and interpolation for video cutout. *ACM Transactions on Graphics*. 46, 47

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

[49] Fragkiadaki, K., Arbelaez, P., Felsen, P., and Malik, J. (2015). Learning to segment moving objects in videos. In *CVPR*. 22, 25, 74

[50] Fragkiadaki, K. and Shi, J. (2011). Detection free tracking: Exploiting motion and topology for segmenting and tracking under entanglement. In *CVPR*. 32

[51] Fragkiadaki, K., Zhang, G., and Shi, J. (2012a). Video segmentation by tracing discontinuities in a trajectory embedding. In *CVPR*. 25, 55, 74

[52] Fragkiadaki, K., Zhang, W., Zhang, G., and Shi, J. (2012b). Two-granularity tracking: Mediating trajectory and detection graphs for tracking under occlusions. In *ECCV*. 25, 52

[53] Fu, J., Liu, J., Wang, Y., Zhou, J., Wang, C., and Lu, H. (2019). Stacked deconvolutional network for semantic segmentation. *IEEE Transactions on Image Processing*. 58

[54] Galasso, F., Keuper, M., Brox, T., and Schiele, B. (2014). Spectral graph reduction for efficient image and streaming video segmentation. In *CVPR*. xx, 86

[55] Galasso, F., Nagaraja, N., Cardenas, T., Brox, T., and B.Schiele (2013). A unified video segmentation benchmark: Annotation, metrics and analysis. In *ICCV*. xx, 17, 33, 84, 86

[56] Galushin, S. and Kudinov, P. (2015). Scenario grouping and classification methodology for postprocessing of data generated by integrated deterministic-probabilistic safety analysis. *Science and Technology of Nuclear Installations*. 1

[57] Gurobi Optimization, LLC (2021). Gurobi Optimizer Reference Manual. 71

[58] Hasegawa, S., Wada, K., Kitagawa, S., Uchimi, Y., Okada, K., and Inaba, M. (2019). Graspfusion: Realizing complex motion by learning and fusing grasp modalities with instance segmentation. In *ICRA*. 19, 35, 36

[59] He, Y., Chiu, W., Keuper, M., and Fritz, M. (2017). Std2p: RGBD semantic segmentation using spatio-temporal data driven pooling. In *CVPR*. 20, 38

[60] Ho, K., Kardoost, A., Pfreundt, F.-J., Keuper, J., and Keuper, M. (2020). A two-stage minimum cost multicut approach to self-supervised multiple person tracking. In *ACCV*. xv, 5, 6, 11, 12, 66, 89

[61] Hornáková, A., Henschel, R., Rosenhahn, B., and Swoboda, P. (2020). Lifted disjoint paths with application in multiple object tracking. In *ICML*. 30

[62] Horňáková, A., Lange, J. H., and Andres, B. (2017). Analysis and optimization of graph decompositions by lifted multicuts. In *ICML*. 28, 30

[63] Hu, Y. T., Huang, J. B., and Schwing, A. G. (2018). Unsupervised Video Object Segmentation Using Motion Saliency-Guided Spatio-Temporal Propagation. In *ECCV*. 22

[64] Ilg, E., Çiçek, Ö., Galesso, S., Klein, A., Makansi, O., Hutter, F., and Brox, T. (2018a). Uncertainty estimates and multi-hypotheses networks for optical flow. In *ECCV*. 115

[65] Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., and Brox, T. (2017). Flownet 2.0: Evolution of optical flow estimation with deep networks. In *CVPR*. xxv, xxvi, 1, 19, 20, 36, 37, 45, 63, 81, 82

[66] Ilg, E., Saikia, T., Keuper, M., and Brox, T. (2018b). Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation. In *ECCV*. xx, xxv, 19, 20, 36, 37, 42, 44, 45, 81, 82, 85, 86

[67] Irani, M. and Anandan, P. (1998). A unified approach to moving object detection in 2d and 3d scenes. *TPAMI*. 24

[68] J. Shi and Malik, J. (2000). Normalized cuts and image segmentation. *TPAMI*. 24, 26

[69] Jain, S., Xiong, B., and Grauman, K. (2017). Fusionseg: Learning to combine motion and appearance for fully automatic segmentation of generic objects in videos. *arXiv preprint arXiv:1701.05384*. 23

[70] Jain, S. D. and Grauman, K. (2014). Supervoxel-consistent foreground propagation in video. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *ECCV*. 20

[71] Jampani, V., Gadde, R., and Gehler, P. V. (2017). Video propagation networks. In *ICCV*. 38, 46, 47

[72] Jang, W. D. and Chang-Su, K. (2017). Online video object segmentation via convolutional trident network. In *CVPR*. Institute of Electrical and Electronics Engineers Inc. 46, 47

[73] Ji, P., Li, H., Salzmann, M., and Dai, Y. (2014). Robust motion segmentation with unknown correspondences. In *ECCV*. 69

[74] Jung, S., Ziegler, S., Kardoost, A., and Keuper, M. (2022). Optimizing edge detection for image segmentation with multicut penalties. In *GCPR*. 5, 11, 12, 103, 126

[75] Kanopoulos, N., Vasanthavada, N., and Baker, R. L. (1988). Design of an image edge detection filter using the sobel operator. *IEEE Journal of solid-state circuits*. 59

[76] Kappes, J., Swoboda, P., Savchynskyy, B., Hazan, T., and Schnörr, C. (2016a). Multicuts and perturb & map for probabilistic graph clustering. *Journal of Mathematical Imaging and Vision*. 107

[77] Kappes, J. H., Andres, B., Hamprecht, F. A., Schnörr, C., Nowozin, S., Batra, D., Kim, S., Kausler, B. X., Kröger, T., Lellmann, J., Komodakis, N., Savchynskyy, B., and Rother, C. (2015a). A comparative study of modern inference techniques for structured discrete energy minimization problems. *IJCV*. 28, 29

[78] Kappes, J. H., Speth, M., Andres, B., Reinelt, G., and Schnörr, C. (2011). Globally optimal image partitioning by multicuts. In *EMMCVPR*. 28, 30, 107

[79] Kappes, J. H., Speth, M., Reinelt, G., and Schnörr, C. (2016b). Higher-order segmentation via multicuts. *Computer Vision and Image Understanding*. 3, 26, 28, 30, 68, 107

[80] Kappes, J. H., Swoboda, P., Savchynskyy, B., Hazan, T., and Schnörr, C. (2015b). Probabilistic correlation clustering and image partitioning using perturbed multicuts. In *SSVM*. 30

[81] Kardoost, A., Ho, K., Ochs, P., and Keuper, M. (2020). Self-supervised sparse to dense motion segmentation. In *ACCV*. xvii, xxiii, xxvii, 4, 5, 6, 9, 12, 24, 32, 33, 51, 69, 106, 107, 114, 118, 119, 120, 126

[82] Kardoost, A. and Keuper, M. (2018). Solving minimum cost lifted multicut problems by node agglomeration. In *ACCV*. 3, 10, 12, 89

[83] Kardoost, A. and Keuper, M. (2021). Uncertainty in minimum cost multicuts for image and motion segmentation. In *UAI*. 4, 6, 10, 12, 33, 105

[84] Kardoost, A., Müller, S., Weickert, J., and Keuper, M. (2021). Object segmentation tracking from generic video cues. In *ICPR*. 2, 8, 12, 35, 126

[85] Kendoul, F., Fantoni, I., and Nonami, K. (2009). Optic flow-based vision system for autonomous 3d localization and control of small aerial vehicles. *Robotics and Autonomous Systems*. 19, 36

[86] Kernighan, B. W. and Lin, S. (1970). An efficient heuristic procedure for partitioning graphs. *Bell Systems Technical Journal*. 29, 57

[87] Keuper, M. (2017). Higher-order minimum cost lifted multicuts for motion segmentation. In *ICCV*. xx, 10, 21, 25, 27, 30, 31, 32, 33, 38, 52, 57, 67, 68, 74, 75, 76, 77, 78, 80, 81, 82, 83, 86, 87

[88] Keuper, M., Andres, B., and Brox, T. (2015a). Motion trajectory segmentation via minimum cost multicuts. In *ICCV*. xvii, xix, xx, xxv, xxvi, 4, 21, 24, 25, 30, 31, 32, 33, 38, 52, 53, 54, 55, 56, 57, 58, 61, 62, 63, 69, 74, 76, 77, 78, 80, 81, 82, 83, 84, 85, 86, 105, 106, 114

[89] Keuper, M., Levinkov, E., Bonneel, N., Lavoué, G., Brox, T., and Andres, B. (2015b). Efficient decomposition of image and mesh graphs by lifted multicuts. *ICCV*. xxi, xxii, xxiv, xxvii, 3, 4, 6, 8, 26, 27, 29, 30, 57, 68, 69, 71, 72, 74, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 103, 105, 107, 108, 114, 120, 121, 122, 123, 125, 126

[90] Keuper, M., Tang, S., Andres, B., Brox, T., and Schiele, B. (2020). Motion segmentation multiple object tracking by correlation co-clustering. *TPAMI*. 6

[91] Khoreva, A., Benenson, R., Ilg, E., Brox, T., and Schiele, B. (2017). Lucid data dreaming for object tracking. In *The 2017 DAVIS Challenge on Video Object Segmentation - CVPR Workshops*. 20

[92] Kim, S., Nowozin, S., Kohli, P., and Chang, D. Y. (2011). Higher-order correlation clustering for image segmentation. In *NeurIPS*. 3, 26, 28, 30, 68

[93] Kim, S., Yoo, C. D., and Nowozin, S. (2014). Image segmentation using higher-order correlation clustering. *TPAMI*. 28, 30

[94] Kirillov, A., Levinkov, E., Andres, B., Savchynskyy, B., and Rother, C. (2017). Instancecut: From edges to instances with multicut. In *CVPR*. 30

[95] Koffka, K. (1935). *Principles of Gestalt Psychology*. Hartcourt Brace Jovanovich, NewYork. 2, 22, 52

[96] Kohli, P. and Torr, P. H. S. (2006). Measuring uncertainty in graph cut solutions – efficiently computing min-marginal energies using dynamic graph cuts. In *ECCV*. 107

[97] Krähenbühl, P. and Koltun, V. (2011). Efficient inference in fully connected crfs with gaussian edge potentials. In *Advances in neural information processing systems*. 59, 62

[98] Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*. 16

[99] Lao, D. and Sundaramoorthi, G. (2018). Extending layered models to 3d motion. In *ECCV*. 22, 62, 63

[100] Leung, T. and Malik, J. (1998). Contour continuity in region based image segmentation. In *ECCV*. 125, 126

[101] Levinkov, E., Kardoost, A., Andres, B., and Keuper, M. (2022). Higher-order multicuts for geometric model fitting and motion segmentation. *TPAMI*. xviii, 4, 9, 10, 12, 14, 16, 17, 24, 28, 29, 33, 67, 68, 69, 70

[102] Levinkov, E., Kirillov, A., and Andres, B. (2017). A comparative study of local search algorithms for correlation clustering. In *GCPR*. 4, 26, 68, 72

[103] Lezama, J., Alahari, K., Sivic, J., and Laptev, I. (2011). Track to the future: Spatio-temporal video segmentation with long-range motion cues. In *CVPR*. 69

[104] Li, F., Kim, T., Humayun, A., Tsai, D., and Rehg, J. M. (2013a). Video segmentation by tracking many figure-ground segments. In *ICCV*. xvii, xxv, 15, 37, 44, 48, 49

[105] Li, Z., Guo, J., Cheong, L., and Zhou, S. (2013b). Perspective motion segmentation via collaborative clustering. In *ICCV*. 2, 69

[106] Lim, H., Lim, J., and Kim, H. J. (2014). Real-time 6-dof monocular visual slam in a large-scale environment. In *ICRA*. 19, 36, 41

[107] Lucas, B. D. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2*. 18

[108] Maczyta, L., Bouthemy, P., and Meur, O. (2019). Cnn-based temporal detection of motion saliency in videos. *Pattern Recognition Letters*. 22

[109] Maninis, K., Pont-Tuset, J., Arbeláez, P., and Gool, L. V. (2017). Convolutional oriented boundaries: From image segmentation to high-level tasks. *TPAMI*. 19, 20, 36, 37, 42, 44

[110] Maninis, K.-K., Caelles, S., Chen, Y., Pont-Tuset, J., Leal-Taixé, L., Cremers, D., and Van Gool, L. (2018). Video object segmentation without temporal information. *TPAMI*. xix, 21, 38, 46, 47, 48

[111] Märki, N., Perazzi, F., Wang, O., and Sorkine-Hornung, A. (2016). Bilateral space video segmentation. In *CVPR*. 46, 47

[112] Martin, D. R., Fowlkes, C. C., and Malik, J. (2004). Learning to detect natural image boundaries using local brightness, color, and texture cues. *TPAMI*. 17

[113] Mayer, N., Ilg, E., P.Häusser, Fischer, P., Cremers, D., Dosovitskiy, A., and Brox, T. (2016). A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *CVPR*. 19

[114] McGuire, K., de Croon, G., De Wagter, C., Tuyls, K., and Kappen, H. (2017). Efficient optical flow and stereo vision for velocity estimation and obstacle avoidance on an autonomous pocket drone. *IEEE Robotics and Automation Letters*. 19, 36

[115] Meilă, M. (2007). Comparing clusterings—an information based distance. *J. Multivar. Anal.* 17, 95, 99, 114, 116

[116] Müller, S., Ochs, P., Weickert, J., and Graf, N. (2016). Robust interactive multi-label segmentation with an advanced edge detector. In *Pattern Recognition*. 21, 38, 42, 54

[117] Nagaraja, N., Schmidt, F., and Brox, T. (2015). Video segmentation with just a few strokes. In *ICCV*. 2, 19, 21, 36, 38

[118] Nguyen, A., Kanoulas, D., Caldwell, D. G., and Tsagarakis, N. G. (2017). Object-based affordances detection with convolutional neural networks and dense conditional random fields. In *IROS*. 35

[119] Nieuwenhuis, C. and Cremers, D. (2013). Spatially varying color distributions for interactive multilabel segmentation. *TPAMI*. 3, 21, 37, 38, 40, 41

[120] Nowozin, S. and Jegelka, S. (2009). Solution stability in linear programming relaxations: Graph partitioning and unsupervised learning. In *ICML*. 30

[121] Ochs, P. and Brox, T. (2011). Object segmentation in video: a hierarchical variational approach for turning point trajectories into dense regions. In *ICCV*. xx, 69, 78, 81

[122] Ochs, P. and Brox, T. (2012). Higher order motion models and spectral clustering. In *CVPR*. 32, 69, 76, 77, 78, 80

[123] Ochs, P., Malik, J., and Brox, T. (2014). Segmentation of moving objects by long term video analysis. *TPAMI*. xvii, xviii, xix, xx, xxiv, xxv, 3, 4, 15, 16, 23, 24, 25, 30, 31, 32, 33, 52, 53, 54, 55, 60, 61, 62, 63, 65, 69, 74, 76, 78, 79, 80, 85, 86, 105, 106, 107, 114, 118, 122

[124] Orbanz, P. and Buhmann, J. (2008). Nonparametric bayesian image segmentation. *ICCV*. 107

[125] Papazoglou, A. and Ferrari, V. (2013). Fast object segmentation in unconstrained video. In *ICCV*. 22

[126] Paul, M., Mayer, C., Gool, L. V., and Timofte, R. (2020). Efficient video semantic segmentation with labels propagation and refinement. In *WACV*. 2, 19, 21, 38

[127] Perazzi, F., Khoreva, A., Benenson, R., Schiele, B., and Sorkine-Hornung, A. (2017). Learning video object segmentation from static images. In *CVPR*. 21, 36, 38, 46, 47, 48

[128] Perazzi, F., Pont-Tuset, J., McWilliams, B., Gool, L. V., Gross, M., and Sorkine-Hornung, A. (2016). A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*. xvii, xviii, xix, xxiii, xxiv, 2, 7, 13, 15, 19, 20, 23, 32, 33, 36, 37, 43, 44, 53, 54, 60, 61, 62, 64, 84, 105, 106, 107, 114, 117, 122

[129] Perazzi, F., Wang, O., Gross, M., and Sorkine-Hornung, A. (2015). Fully connected object proposals for video segmentation. In *ICCV*. 46, 47

[130] Pishchulin, L., Insafutdinov, E., Tang, S., Andres, B., Andriluka, M., Gehler, P., and Schiele, B. (2016). Deepcut: Joint subset partition and labeling for multi person pose estimation. In *CVPR*. 30

[131] Pont-Tuset, J., Perazzi, F., Caelles, S., Arbeláez, P., Sorkine-Hornung, A., and Van Gool, L. (2017). The 2017 DAVIS challenge on video object segmentation. *arXiv:1704.00675*. xvii, 2, 7, 14, 15, 36, 37, 44

[132] Price, B. L., Morse, B. S., and Cohen, S. (2009). LIVEcut: Learning-based interactive video segmentation by evaluation of multiple propagated cues. In *ICCV*. 2, 19, 21, 38

[133] Rao, S. R., Tron, R., Vidal, R., and Ma, Y. (2008). Motion segmentation via robust subspace separation in the presence of outlying, incomplete, or corrupted trajectories. In *CVPR*. 25, 52

[134] Rempfler, M., Lange, J. H., Jug, F., Blasse, C., Myers, E., Menze, B., and Andres, B. (2017). Efficient algorithms for moral lineage tracing. *ICCV*. 30

[135] Ronneberger, O., P.Fischer, and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer. (available on arXiv:1505.04597 [cs.CV]). xvii, 4, 20, 21, 53, 55, 58, 119

[136] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *IJCV*. 23

[137] Shi, F., Zhou, Z., Xiao, J., and Wu, W. (2013). Robust trajectory clustering for motion segmentation. In *ICCV*. 2, 25, 52, 69

[138] Shi, J. and Malik, J. (1998). Motion segmentation and tracking using normalized cuts. In *ICCV*. 32

[139] Siam, M., Elkerdawy, S., Jagersand, M., and Yogamani, S. (2017). Deep semantic segmentation for automated driving: Taxonomy, roadmap and challenges. In *International Conference on Intelligent Transportation Systems*. 58

[140] Siam, M., Gamal, M., Abdel-Razek, M., Yogamani, S., and Jagersand, M. (2018a). Rtseg: Real-time semantic segmentation comparative study. In *International Conference on Image Processing*. 58

[141] Siam, M., Jiang, C., Lu, S. W., Petrich, L., Gamal, M., Elhoseiny, M., and Jägersand, M. (2019). Video object segmentation using teacher-student adaptation in a human robot interaction (HRI) setting. In *ICRA*. 21, 35, 38

[142] Siam, M., Mahgoub, H., Zahran, M., Yogamani, S., Jagersand, M., and El-Sallab, A. (2018b). Modnet: Motion and appearance based moving object detection network for autonomous driving. In *International Conference on Intelligent Transportation Systems (ITSC)*. 22

[143] Song, J., Andres, B., Black, M., Hilliges, O., and Tang, S. (2019). End-to-end learning for graph decomposition. In *ICCV*. 5, 103, 126

[144] Sundberg, P., Brox, T., Maire, M., Arbeláez, P., and Malik, J. (2011). Occlusion boundary detection and figure/ground assignment from optical flow. In *CVPR*. xx, 39, 84, 86

[145] Swoboda, P. and Andres, B. (2017). A message passing algorithm for the minimum cost multicut problem. In *CVPR*. 26

[146] Tang, S., Andres, B., Andriluka, M., and Schiele, B. (2016). Multi-person tracking by multicut and deep matching. *CoRR*. 6

[147] Tang, S., Andriluka, M., Andres, B., and Schiele, B. (2017). Multiple people tracking by lifted multicut and person re-identification. In *CVPR*. 6, 27, 30

[148] Taylor, B., Karasev, V., and Soattoc, S. (2015). Causal video object segmentation from persistence of occlusions. In *CVPR*. xxvi, 83

[149] Tokmakov, P., Alahari, K., and Schmid, C. (2017a). Learning motion patterns in videos. In *CVPR*. xxvi, xxvii, 22, 23, 33, 52, 53, 82, 83, 85

[150] Tokmakov, P., Alahari, K., and Schmid, C. (2017b). Learning video object segmentation with visual memory. In *ICCV*. xxvi, 7, 22, 23, 52, 62, 63, 82, 83, 84

[151] Tomczak, J., Zareba, S., Ravanbakhsh, S., and Greiner, R. (2018). Low-dimensional perturb-and-map approach for learning restricted boltzmann machines. *Neural Processing Letters*. 107

[152] Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jegou, H. (2021). Training data-efficient image transformers and distillation through attention. In *ICML*, Proceedings of Machine Learning Research. 127

[153] Tsai, Y. H., Yang, M. H., and Black, M. J. (2016). Video segmentation via object flow. In *CVPR*. 2, 19, 20, 21, 22, 38, 46, 47

[154] Vazquez-Reina, A., Avidan, S., Pfister, H., and Miller, E. (2010). Multiple hypothesis video segmentation from superpixel flows. In *ECCV*. 8

[155] Voigtlaender, P. and Leibe, B. (2017). Online adaptation of convolutional neural networks for video object segmentation. In *BMVC*. 20, 21, 36, 38

[156] Wang, Q., Zhang, L., Bertinetto, L., Hu, W., and Torr, P. H. (2019a). Fast online object tracking and segmentation: A unifying approach. In *CVPR*. 46, 47

[157] Wang, Z., Xu, J., Liu, L., Zhu, F., and Shao, L. (2019b). Ranet: Ranking attention network for fast video object segmentation. *ICCV*. 36

[158] Wannenwetsch, A. S., Keuper, M., and Roth, S. (2017). Probflow: Joint optical flow and uncertainty estimation. In *ICCV*. 1

[159] Weinzaepfel, P., Revaud, J., Harchaoui, Z., and Schmid, C. (2013). DeepFlow: Large displacement optical flow with deep matching. In *ICCV*. 1, 45

[160] Wen, L., Du, D., Lei, Z., Li, S. Z., and Yang, M. H. (2015). JOTS: Joint online tracking and segmentation. In *CVPR*. 48

[161] Xie, S. and Tu, Z. (2015). Holistically-nested edge detection. In *ICCV*. 19, 20, 36, 37, 42, 44

[162] Yang, C., Lamdouar, H., Lu, E., Zisserman, A., and Xie, W. (2021a). Self-supervised video object segmentation by motion grouping. In *ICCV*. 23, 85

[163] Yang, G. and Ramanan, D. (2021). Learning to segment rigid motions from two frames. In *CVPR*. 23

[164] Yang, L., Wang, Y., Xiong, X., Yang, J., and Katsaggelos, A. K. (2018). Efficient video object segmentation via network modulation. *CVPR*. xix, 46, 48

[165] Yang, Y., Lai, B., and Soatto, S. (2021b). Dystab: Unsupervised object segmentation via dynamic-static bootstrapping. In *CVPR*. 23, 85

[166] Yarkony, J., Ihler, A., and Fowlkes, C. (2012). Fast planar correlation clustering for image segmentation. In *ECCV*. 30

[167] Yarkony, J., Zhang, C., and Fowlkes, C. (2015). Hierarchical planar correlation clustering for cell segmentation. In *EMMCVPR*. 30

[168] Yoon, J. S., Rameau, F., Kim, J., Lee, S., Shin, S., and Kweon, I. S. (2017). Pixel-level matching for video object segmentation using convolutional neural networks. In *ICCV*. 46, 47

[169] Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., and Torr, P. (2015). Conditional random fields as recurrent neural networks. In *ICCV*. 59, 60, 62

[170] Zingg, S., Scaramuzza, D., Weiss, S., and Siegwart, R. (2010). Mav navigation through indoor corridors using optical flow. In *ICRA*. 19, 36

List of figures

List of tables