Temporally Coherent Cost Volume Filtering-based Depth Propagation in Videos

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieurin

im Rahmen des Studiums

Medieninformatik

eingereicht von

Tanja Schausberger
Matrikelnummer 0826211

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Ao. Univ.-Prof. Dipl.-Ing. Mag. Dr. Margrit Gelautz
Mitwirkung: Dipl.-Ing. Mag. Nicole Brosch

Wien, 15.02.2015

(Unterschrift Verfasserin) (Unterschrift Betreuung)

Temporally Coherent Cost Volume Filtering-based Depth Propagation in Videos

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieurin

in

Media Informatics

by

Tanja Schausberger
Registration Number 0826211

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Ao. Univ.-Prof. Dipl.-Ing. Mag. Dr. Margrit Gelautz
Assistance: Dipl.-Ing. Mag. Nicole Brosch

Vienna, 15.02.2015

(Signature of Author) (Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Tanja Schausberger
Annenhofsiedlung 28a, 3160 Traisen

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum)

(Unterschrift Verfasserin)

Acknowledgements

I would like to thank my supervisor Margrit Gelautz and her assistant Nicole Brosch for their support throughout the whole process of this master thesis.

Moreover, I would like to thank my parents who supported me financially and made it possible for me to study. Especially, I would like to thank Ingrid, Peter, Stefan, Robert and Anja for at once annoying and supporting me. Also, I would like to thank my nieces Lena and Jana who surprise me over and over again.

The diploma thesis was carried out in close collaboration with projects supported by the Vienna Science and Technology Fund (WWTF, ICT08-019) and the Austrian Research Promotion Agency (FFG, ‚Hyperion3D’-840222).

Abstract

The growing availability of three-dimensional (3D) displays in the commercial sector and related trends in this field increase the demand for 3D content. Therefore, the development of tools for the generation of 3D content is of academic and commercial interest. This thesis proposes a semi-automatic 2D-to-3D depth propagation algorithm, which requires only little user-effort and delivers good quality results. To perform the propagation, the user only needs to assign depth by annotating scribbles on the first and last frame. The proposed algorithm builds upon a cost volume filtering-based video object segmentation algorithm and propagates the user-assigned depth values to all frames of the video. Our work focuses on several quality attributes including temporal consistency, edge-sharpness mismatches and depth changes over time. The proposed algorithm achieves improved temporal consistency by enhanced temporal filtering. Moreover, the algorithm accounts for the depth change of an object moving throughout the video by the incorporation of the depth order of the video. Evaluations show that the proposed algorithm requires minimal user interaction and generates spatio-temporally coherent depth maps with a perceptually consistent depth change for objects that change their depth in time. Moreover, a quantitative and visual comparison of the proposed algorithm to a related 2D-to-3D conversion algorithm is performed. It is shown that high-quality results can be achieved by using a robust optical flow estimation and a comfortable scribble annotation by the user.

Kurzfassung

Die wachsende Verfügbarkeit von 3D-Bildschirmen im kommerziellen Bereich und damit verbundene Trends steigern die Nachfrage nach 3D-Inhalten. Daher ist die Entwicklung von Technologien zur Generierung von 3D-Inhalten von akademischem und kommerziellem Interesse. Im Zuge dieser Diplomarbeit wird ein semi-automatischer 2D-zu-3D Tiefenpropagierungsalgorithmus vorgestellt, welcher nur wenig Benutzeraufwand benötigt und Ergebnisse von guter Qualität liefert. Um die Propagierung durchzuführen, müssen BenutzerInnen lediglich grobe Tiefeninformationen auf dem ersten und letzten Bild des Videos angeben. Der vorgestellte Algorithmus baut auf einem Objektsegmentierungsalgorithmus für Videos auf, welcher mit Kostenfilterung arbeitet, und propagiert automatisch die von den BenutzerInnen angegebene Tiefe auf das gesamte Video. Diese Diplomarbeit fokussiert dabei besonders auf einige Qualitätsmerkmale wie zeitliche Kohärenz, unscharfe Tiefenkanten und zeitliche Tiefenänderungen. Der vorgestellte Algorithmus erzielt durch eine erweiterte zeitliche Filterung verbesserte Ergebnisse bezüglich der zeitlichen Kohärenz. Außerdem wird die zeitliche Tiefenänderung von Objekten, die sich im Laufe des Videos in der Tiefe bewegen, durch die Miteinbeziehung der Tiefenordnung der Objekte im Video ermöglicht. Evaluierungen zeigen, dass der vorgestellte Algorithmus mit minimalem Benutzeraufwand zeitlich-kohärente Tiefenkarten mit wahrnehmungskonsistenten Tiefenänderungen erzeugt. Zusätzlich wird der vorgestellte Algorithmus mit einem ähnlichen 2D-zu-3D Konvertierungsalgorithmus verglichen. Es wird gezeigt, dass hohe Qualität durch die Verwendung von robust geschätzten Optical Flow Vektoren und komfortable Tiefenannotierung durch BenutzerInnen erzielt werden kann.

Contents

Acronyms **xi**

1 Introduction **1**
- 1.1 Motivation and Problem Statement 1
- 1.2 Aim of the Thesis 4
- 1.3 Structure of the Thesis 5

2 State of the Art: Semi-Automatic 2D-to-3D Conversion **7**
- 2.1 Filtering-based Methods 7
- 2.2 Machine Learning Algorithms 9
- 2.3 Optimization-based Methods 10
- 2.4 Segmentation-based Methods 12
- 2.5 Summary 17

3 Principles of Video Segmentation and Cost Volume Filtering **19**
- 3.1 Representation and Features for Segmentation 19
- 3.2 Segmentation 22
- 3.3 Cost-Volume Filtering 25
- 3.4 Summary 32

4 Cost Volume Filtering-based Depth Propagation **33**
- 4.1 Naive Extension to a Simple Depth Propagation Framework 34
- 4.2 Temporal Consistency Improvement 37
- 4.3 Spatial Influence Extension 39
- 4.4 Temporal Depth Change Extension 46
- 4.5 Summary 68

5 Evaluation **69**
- 5.1 Evaluation Methodology 69
- 5.2 Evaluation with Depth Ground Truth 74
- 5.3 Comparison with a Similar Algorithm 90

6 Summary and Outlook **99**

Bibliography **101**

Acronyms

2D	Two-dimensional
3D	Three-dimensional
CVF	Cost Volume Filtering
DAG	Directed Acyclic Graph
DB	Depth Blending
FPS	Frames Per Second
GT	Ground Truth
KLT	Kanade-Lucas-Tomasi
MLA	Machine Learning Algorithms
MST	Minimum Spanning Tree
NE	Naive Extension
RGB	Red, Green, Blue
SIE	Spatial Influence Extension
SIFT	Scale-Invariant Feature Transform
SVM	Support Vector Machine
TCI	Temporal Consistency Improvement
TDCE	Temporal Depth Change Extension
UI	User Interface
WTA	Winner-take-all

CHAPTER 1

Introduction

1.1 Motivation and Problem Statement

During the last years the popularity of three-dimensional (3D) films increased more and more. The future trend and the availability of 3D displays in the commercial sector direct to a higher demand of 3D media. When watching a 3D film, the depth impression is generated through two slightly shifted images from the same scene. The human brain interprets the shift between the two images as depth and one perceives three dimensions. For producing 3D content either special cameras, which record two or more images, or technologies for the conversion of existing two-dimensional (2D) media to 3D are used. This work is focusing on the latter, i.e., 2D-to-3D conversion.

On the one hand new 3D productions can be produced with special equipment, which is usually quite expensive. With a stereoscopic camera it is possible to record two images from a slightly different angle at the same time. Similarly, two or more cameras can be used but have to be aligned properly. Furthermore, a special depth camera (e.g. Time-of-Flight [41] or Kinect [74]) can record depth information of the scene. These cameras are saving the actual *depth*, that is the distance from the camera to the point in depth maps. In contrast, *disparity* describes the shift of a point between the left and the right image. Thus, disparity corresponds to the relative depth [33]. In the course of this thesis the terms depth and disparity are used synonymously. Moreover, a *depth map* is usually visualized as a grey scale image with the same dimensions as the corresponding frame. The intensity of each pixel value represents the distance to the camera. The darker the value, the larger is the distance between the camera and the point and vice versa (cf. Figure 1.1). Depth maps are used to generate new or adapt existing 3D material to different media. Since the shift between two images adapted for a cinema screen appears larger on a small screen caused by reasons of geometric relations, 3D material has to be adopted to the size of the screen. Thus, if not adopted the 3D material of a cinema production would cause an uncomfortable 3D effect when watching it on television. For both these above mentioned options, i.e., recording two images with a stereoscopic camera or recording depth with a special depth camera, it has to be decided previously if the production should be in

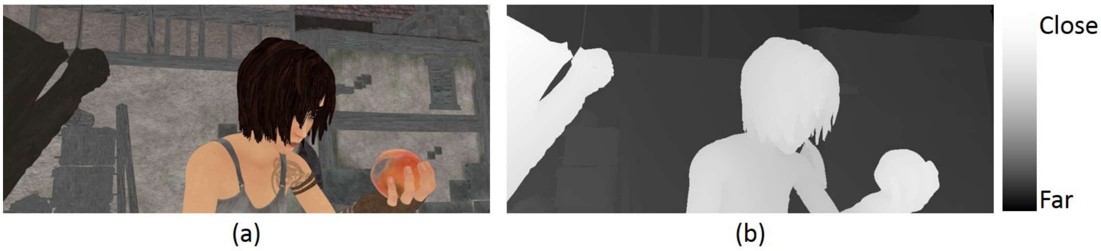
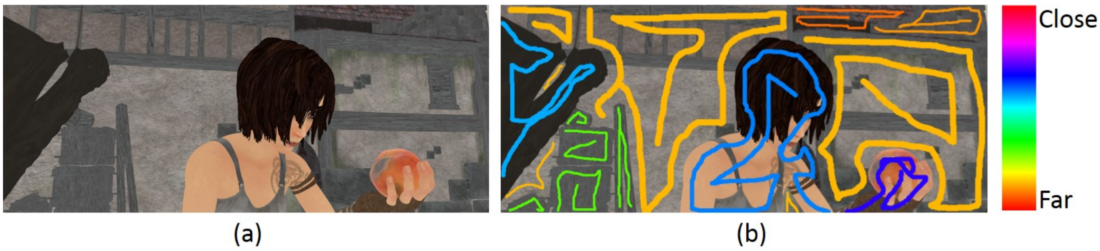


Figure 1.1: (a) Input image and (b) corresponding depth map. The depth values are presented as grey values from dark (far) to white (close). [16]

3D. Due to the high costs of the special equipment, this is usually just profitable for cinema productions such as the film series ‚The Hobbit' by Peter Jackson. For example, the official production costs for the three film series recorded in 3D with high frame rate, that is 48 frames instead of 24 frames per second (FPS), were about 560 Mio. US $ [22].

On the other hand, 2D-to-3D conversion techniques are often a cheaper alternative for 3D content generation. Thus, it is possible to generate new stereoscopic (3D) content from already existing monocular (2D) videos. These 2D-to-3D conversion techniques focus on estimating a depth map for every frame of the film without a given second view. Special rendering algorithms [25] compute with the monocular colour image and the corresponding depth map the additional images (a second view) needed for the 3D perception. For that purpose, depth maps have to be high-quality to receive a comfortable depth impression. In this context, temporal consistency, no flickering and smooth depth changes of moving objects in the scene are a few quality features. Furthermore, it is important to convert the film within a reasonable time. In the case of a 90 minutes film with 24 FPS, about 130000 depth maps have to be computed. Thus, there are several features which have to be considered when performing a 2D-to-3D conversion. Generally, there are three kinds of 2D-to-3D conversion techniques, which differ in time, quality, costs and artistic freedom:

1. *Manual 2D-to-3D conversion:* This technique typically consists of three steps. First, users extract objects and surfaces manually frame by frame, which is called *rotoscoping*. While this segmentation is still mainly manually, it is worth to note that software can be used to facilitate this process. Moreover, software helps to segment objects and track them to the next frame [2]. In a second step, depth is assigned by the user to the segmented objects and surfaces and, in doing so, a depth map is created for each frame. The last step is to fill occlusions which occur after shifting the scene to a slightly different angle. Those occlusions appear in the final rendered second view since there is no information available for regions which are occluded in the original view. Thus, those regions have to be either manually filled or with the help of inpainting algorithms (e.g., [4]). The advantages of manual conversion are artistic freedom and high-quality results. However, it takes a lot of time to work with each frame separately which results in high costs. This method is usually used to professionally convert old 2D films to 3D for cinema or high-quality 3DTV. For example, the conversion of ‚Starwars Episode 1: The Phantom Menace' took



(a) (b)

Figure 1.2: (a) Input image and (b) corresponding user-defined scribble map. The depth values are represented in the form of coloured scribbles as the hue scale shows. [16]

10 months with more than 1000 rotoscope artists. The costs for converting the film are estimated to be even higher than the original production costs which were 115 Mio. US $ [46], [71], [77].

2. *Automatic 2D-to-3D conversion:* These methods automatically convert full monocular videos to stereoscopic videos, thus, without any user work. Companies like Samsung or JVC are already using such real-time automatic algorithms within their 3D displays [75]. Automatic conversion algorithms (e.g. [21], [52], [69]) are using monocular depth cues such as focus and defocus, perspective, relative size, light and shading to estimate the depth. Depth also can be estimated from motion. The basic idea is that objects which are close to the camera move faster than objects which are far [85]. However, since there is basically no interaction by the user it is hard to control the whole process. Therefore, such algorithms might work good on certain 2D scenes but often result in low quality. However, it is possible to convert videos in real-time [75].

3. *Semi-automatic 2D-to-3D conversion:* As discussed above, on the one hand there is manual conversion, which is time-consuming and expensive and on the other hand there are automatic algorithms, which are fast but of lower quality. Researchers got aware of this problem and proposed semi-automatic methods. Semi-automatic techniques (e.g., [18], [33], [36], [47], [66], [83], [86], [87]) allow user interaction while the user effort stays low. Generally, only some keyframes are marked with depth values by users. With this user input, depth is propagated to all pixels of a video. Therefore, semi-automatic conversion methods close the gap between manual and automatic techniques. They provide artistic freedom through sparse user input which results in good quality and is still cheap.

2D-to-3D conversion methods become more and more common not only because of financial but also because of technical and artistic reasons. Besides the lack of 3D content and the high costs of the equipment for shooting in 3D, stereo cameras have restrictions. Due to their weight and size it is difficult to use them in action scenes. In that case, the scene has to be converted in the post-production anyway. For these reasons, it is important to reduce the time and increase the quality of 2D-to-3D conversion.

This master thesis focuses on the semi-automatic 2D-to-3D conversion and the improvement of quality with sparse user interaction. In existing algorithms (e.g., [33], [36], [47], [66], [86]), the user has to input the depth information on a few keyframes by annotating scribbles (cf. Figure 1.2). Other methods (e.g., [18], [87]) are working with pre-segmented objects on which depth is assigned by the user. Since object segmentation has to deal with similar problems such as edge-sharpness and consistency over time, this thesis is extending an interactive video segmentation algorithm [11] to a 2D-to-3D conversion method. While segmentation partitions the video into segments that are homogeneous in a certain feature space (e.g., colour), 2D-to-3D conversion identifies parts in a video related to their depth position. The segmentation algorithm of [11] works with minimal user input. The basic idea is that only one scribble is needed to mark the foreground object in the first frame to get a segmentation of fore- and background in every frame of the scene. The challenge of semi-automatic approaches is keeping depth maps temporally coherent with only a few keyframes as input. To this end, it is possible to improve the temporal consistency with filter operations [47]. Moreover, temporal depth changes of objects have to be considered to achieve improved quality.

1.2 Aim of the Thesis

The main goal of this master thesis is to extend the video object segmentation algorithm from [11] to a 2D-to-3D depth propagation algorithm which achieves improved quality compared to state-of-the-art methods (cf. Chapter 2) while keeping the user work low. The main issue is to find a compromise of a good quality achieving method combined with a minimal user input. To this end, this thesis is focusing on some quality features to improve the quality while keeping the user effort low. Thus, *temporal consistency* is an important quality attribute. Temporal depth changes which result in flickering should be avoided and therefore smooth depth changes of objects should be considered. Furthermore, some conversion techniques cannot deal with *edge-sharpness mismatches*, which are the result of over-smoothing at depth edges. Different to that is the *cardboard effect*, where objects suffer from missing depth variation within their bounds. Another desired effect is the *depth change* over time. Thus, it should be able to model an object moving closer to the camera throughout the video.

By using the example of the algorithm of [66], it can be seen that high quality with depth variations and depth changes over time is possible, but at the cost of increased user input. To achieve high quality, the authors recommend to mark every frame of the video with scribbles (cf. Figure 1.2). Therefore, a key goal of this thesis is to achieve good quality with minimal user input. To this end, only the first and the last frames are keyframes with annotated scribbles. Thus, the applied algorithm enables depth changes over time, which is another desired quality attribute. Another issue many algorithms are dealing with is temporal consistency and stability. Often filtering techniques are used in combination with motion estimation (e.g., [83], [47]) to handle consistency and avoid abrupt changes between frames and flickering. Similar to that, the proposed algorithm (cf. Chapter 4) uses an extended version of the guided filter [37] in combination with optical flow vectors.

The extension from a video object segmentation to the 2D-to-3D conversion method consists of several steps which results in a good quality algorithm with minimal user input. In a first step, the segmentation algorithm from [11] (cf. Section 3.3) is extended to perform a naive depth propagation (cf. Section 4.1). For this purpose, cost volume filtering is used (cf. Section 3.3). To perform the propagation, the user only needs to assign depth by annotating scribbles on the first and last frame. In a next step, temporal consistency is improved by adding an enhanced temporal filtering approach similar to [47] using a guided filter implementation for videos [37] (cf. Section 4.2). Next, a spatial influence extension is applied by adding spatial costs to the cost volume term in order to reduce the influence of pixels which are similar in colour but different in depth and far away from the current user-assigned scribble (cf. Section 4.3), which would result in noise. In a last step, temporal depth changes are being enabled (cf. Section 4.4) by incorporating the depth order of the video in order to model the movement of an object in depth throughout the video. As results show in the evaluation (cf. Chapter 5), the proposed algorithm requires minimal user interaction and generates spatio-temporally coherent depth maps with a perceptually consistent depth change for objects that change their depth in time by incorporating the depth order of the video.

1.3 Structure of the Thesis

This master thesis is organized in six chapters. The current one, Chapter 1, overviews the motivation and the problem as well as the aim of the work. In the following, the remaining chapters are discussed:

- Chapter 2 gives an overview of currently important 2D-to-3D conversion algorithms. Since the focus of this thesis is to improve quality while keeping the user input low, this trade-off will be part of the discussion of various approaches in this chapter.

- Chapter 3 discusses image and video features and related segmentation techniques, which are important for the 2D-to-3D conversion. Moreover, since the algorithm introduced in the following chapter is based on a cost volume filtering approach [11] to segment objects, the technique of cost volume filtering is discussed in more detail together with filtering approaches.

- Chapter 4 introduces a new approach of a 2D-to-3D conversion method. The implemented cost volume filtering-based depth propagation algorithm represents an extension of a video object segmentation approach [11]. Moreover, various extensions, including a temporal consistency improvement, a spatial influence extension and a temporal depth change extension and their impacts are presented and discussed in separated sections.

- Chapter 5 evaluates the proposed extensions of the 2D-to-3D algorithm and compares the implemented cost volume filtering-based depth propagation algorithm that is presented in this thesis to a related algorithm.

- Chapter 6 summarizes the achieved results and findings of the newly introduced 2D-to-3D depth propagation approach and discusses possibilities for enhancements.

CHAPTER 2

State of the Art: Semi-Automatic 2D-to-3D Conversion

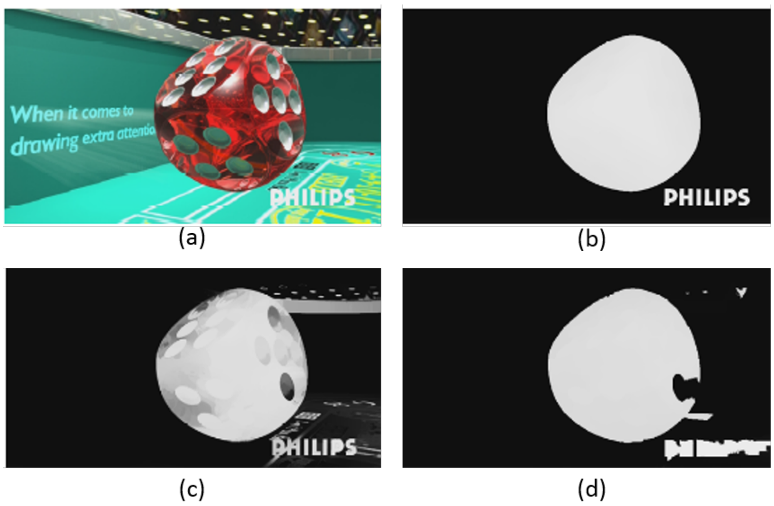
This chapter gives an overview of prior work on semi-automatic 2D-to-3D conversion. The goal of these semi-automatic algorithms is to propagate sparsely given depth values to all pixels. In particular, users sparsely initialise depth values on keyframes by indicating which image regions are close or far from the camera. This can be done by using a few scribbles ([33], [36], [47], [66], [86]) or marking depth on the whole frame ([18], [83], [87]). For the purpose of 2D-to-3D conversion, this assignment has not to be accurate in terms of depth values, but has to be perceptually consistent [33]. Moreover, cognitive studies, i.e., [44], have shown that the depth order influences the perception of the scene more than the absolute depth.

The following overview highlights in particular how the discussed conversion algorithms handle the following challenges: (i) spatial depth variations within objects, (ii) temporal depth changes of objects and (iii) temporal consistency. The lack of depth variations, i.e., (i), is referred to as *cardboard effect*. Missing temporal consistency, i.e., (iii), results in disturbing flickering and abrupt changes over time.

In the following, four groups of 2D-to-3D conversion algorithms are presented: (i) Filtering-based methods that propagate depth values from a keyframe to the next one by using a filter, (ii) Machine Learning Algorithms that learn the relationship between user-defined depth samples and their corresponding colour to propagate given depth to frames of a 2D video, (iii) Optimization-based approaches that express the problem of assigning depth as a global energy function, and (iv) Segmentation-based methods that propagate depths within predetermined segments.

2.1 Filtering-based Methods

The basic idea of filtering-based conversion methods is to iteratively propagate a weighted mean of existing depth values that was computed on one frame to the following frames, without given



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

Figure 2.1: Depth propagation after 100 frames from [83]. (a) source image, (b) depth ground truth, (c) depth propagation using bilateral filter and (d) depth result after the correction step. [83]

depth values. Therefore, users assign depth of the first frame of a video on the entire keyframe. By reducing the assignment of depth from every frame to a few keyframes, the user effort can be significantly reduced compared to a manual conversion. A well-known technique in the field of propagation filtering is the bilateral filter [81] as used in [83]. The bilateral filter is an edge preserving filter and was originally introduced to smooth images. The idea of this filter is to smooth pixels in the filter window according to their distance to the window centre and their colour distance to the pixel centred in the window. Edges can be preserved, because of the colour difference between, e.g., an object and the background. In [83] the bilateral filter is used to propagate depth from the keyframe to the next one.

Basically, filter-based methods are a fast and simple possibility to propagate depth. However, as observed in [83], these methods show a few drawbacks:

1. *Depth ambiguity problem:* The colour of two pixels within a filter window is the same, but depths differ. Thus, only due to the colour information it is not possible to decide which depth value should be propagated. Such ambiguity might happen, if two objects with equal colour, but different depth have an overlapping region.

2. *New colour problem:* A colour that is not present in the reference frame appears in a following one. Thus, the filter has no depth information to the new colour value. That usually happens when new areas of the background are uncovered or a new object enters the scene.

3. *Recursive propagation of errors:* If the errors described above are not corrected, they are recursively propagated to the following frames. Therefore, errors increase from frame to frame and influence the resulting depth map.

The authors in [83] address these problems by correcting the propagation errors after they appear. To this end, they assume that the previous frame (i.e., the keyframe or frame that has already been depth corrected) is correct. They use motion compensation to overwrite the depth values which resulted from using the bilateral filter to propagate depth from the keyframe to the following frame. The motion vectors are estimated from the depth map of the previous and the filtered depth map of the current frame. As shown in Figure 2.1(d) the described problems are being reduced. However, it can also be seen (cf. Figure 2.1(d)) that errors remain, e.g., at the bottom left side of the cube, which might be propagated recursively to the following frames. Since depth values are only assigned in the first frame - the keyframe - and then propagated from one to the next frame, smooth temporal depth changes are not modelled. Hence, some authors such as [18] or [50] adapted the basic filtering-based conversion in order to model such changes. More precisely, the authors of [18] additionally annotate the keyframe at the end of the video and combine the conversion result based on the first and last frame for all frames in between. Moreover, the authors improved the motion compensation by using the colour images instead of the depth maps, for estimating the motion vectors. Since there is little structure in depth maps, the computed motion vectors in [83] are not accurate. By using the colour images for the computation of the motion vectors, the resulting flow is more accurate. However, by adding a keyframe at the end, the user effort increases again, but it is possible to model depth changes.

2.2 Machine Learning Algorithms

Another group of 2D-to-3D conversion algorithms exploit machine learning techniques to convert a given colour video to 3D. A *Machine Learning Algorithm* (MLA) acts like a black box that learns the relationship between the input and the output, which in our case of 2D-to-3D conversion are colour patterns in an input frame and the disparities.

For example, in [36] the creation of depth maps is based on such MLAs. Thus, the input for the MLA is a colour video and the output are depth maps for each frame. The algorithm of the rapid 2D-to-3D conversion [36] is using MLAs in two phases:

1. *Depth mapping:* In the beginning phase an MLA is applied to some keyframes. These can be chosen by some techniques used for shot transitions (e.g. [67]) or manually by the user. For each of the chosen keyframes a proper MLA is trained by samples defined by the user. Therefore, the user marks specific regions with depth values on a coloured keyframe. The MLAs learn the relationship between the coloured samples and their associated depth and use that to generate the depth map for this keyframe.

2. *Depth tweening:* In a second phase the MLAs are used to assign depth to tween frames. *Tween frames* are all frames between two keyframes. The MLAs results from the two surrounding keyframes are used to assign depth. The resulting depths from each MLA are combined in a weighted manner.

Regarding [36], the use of MLA reduces the amount of user interaction significantly compared to manual conversion. A result is shown in Figure 2.2, where for a video with 43 frames only

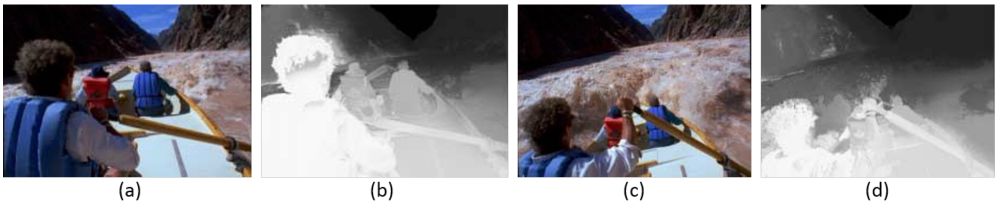


Figure 2.2: Resulting depth maps from [36]. 2D-to-3D conversion of a video with 43 frames with keyframes at 1, 14 and 43 by providing 8.000 training samples over the three keyframes. (a) shows the source image and (b) the resulting depth map by depth tweening at frame six. (c) shows the source image and (d) the resulting depth map by depth tweening at frame 32. [36]

three keyframes were used for the depth propagation. However, the error of the depth map increases with the increasing distance to the nearest keyframe. Additionally, some significant changes in light conditions or fast abrupt motion between two keyframes can lower the quality of the conversion result [36]. The authors state that in this case the keyframes have to be chosen carefully.

2.3 Optimization-based Methods

In case of optimization-based algorithms the propagation step is designed with a global energy function. Particularly, depth is propagated by solving a large linear system that implements a set of constraints by minimisation. Usually this function consists of a data cost term, which represents the costs when a pixel is assigned to a specific depth value or label, and a smoothness term, which is responsible for the propagation, e.g., by considering the similarity of neighbouring pixels.

For example, the algorithm from [33] was one of the first methods that used user-defined scribbles to compute a depth map. Like the algorithm implemented in this work, the method of [33] only uses input scribbles from the first and the last frame of a video shot. In that way it is also possible to model depth changes over time.

In a first step, these user-defined depth values are propagated on frames on which they were drawn. Then *Support Vector Machine* (SVM) [33] classifiers are trained and applied for each depth value defined by the user separately for the first and last frame. By using *Scale-Invariant Feature Transform* (SIFT) [53] some anchor points, which are points with a high confidence value of the classifier, are detected and followed throughout the video. Those are assigned to depth values by choosing the classifier with the lowest error rate. The depth values of non-anchor points are assigned by solving an optimization problem in least squares manner. Specifically, this conversion algorithm minimises a linear system of equations that consists of four types of constraints per pixel:

1. The disparity of a pixel is similar to the disparity of its spatial neighbour.

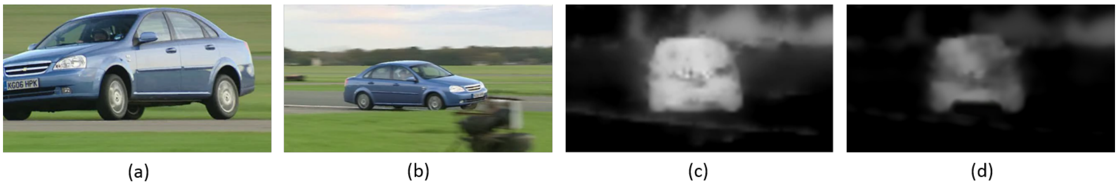


Figure 2.3: Resulting depth map from [33]. (a) the first frame, (b) the last frame, (c) and (d) disparity maps of the same frame between the two keyframes. (c) shows the result without motion-based weighted and (d) with weighting. [33]

2. Disparity changes continuously over time according to motion vectors computed by the optical flow from [61].
3. The disparity values of user-defined scribbles are fixed.
4. The anchor points chosen by the classifiers further guide the propagation.

Since the amount of linear equations increases quadratic with the amount of pixels in a video, the resolution of the video is reduced by the factor of four to make the algorithm more efficient. Afterwards, the solution is upsampled with a joint bilateral technique from [45]. A resulting disparity map of a video is shown in Figure 2.3. The disparity maps in (c) and (d) show the corresponding depth of a frame between the keyframes (a) and (b). Due to the motion-based weighting, i.e., constraint two, the disparity of the car is lower (darker) in (d) than in (c), which refers to a farther distance to the camera.

In [86] the authors solve a similar system of equations as the above discussed algorithm (i.e., [33]). The so called *StereoBrush* framework works with sparse scribble input, where low intensities represent far objects and vice versa. The user can define common scribbles, with only one intensity value, or gradient scribbles, which are useful for depth variation as in slanted or rounded objects. Contrary to [33], in [86] the goal is to directly generate a stereoscopic pair with a left and a right view, instead of computing a depth map. Therefore two *image warps* are computed, which are defined as the mapping functions of the given image to a novel right and left view. Afterwards, the energy function is indirectly solved for these novel views. Instead of propagating scribble values directly to the video, they are put into a disparity hypothesis with a per-pixel set of constraints with an incorporated confidence weight. In that way they enforce the disparity value to be close to the original value of the scribble. The discontinuity preserving smoothness term is based on colour edges and thus helps to preserve object structure. A *saliency map* detects context-aware regions in an image. Such *context-aware* or *salient* regions are parts of an image, which are useful for the observer to describe the image. Those regions are shown in a *saliency map*. The additional saliency weight reduces disparity over-smoothing in such regions. Finally, the authors use a content-and-disparity-aware resizing method to stretch background regions and fill disocclusions with a hole filling method to get the finished new left and right view. A resulting disparity map is shown in Figure 2.4. It can be seen that disparities

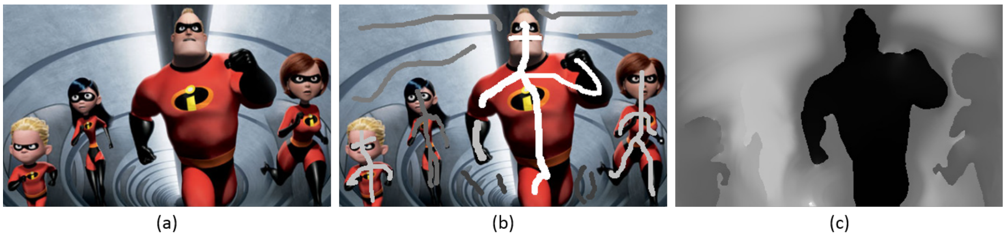


Figure 2.4: Disparity propagation by [86]. (a) the source image, (b) the user-defined scribble map and (c) the resulting disparity map. [86]

at object borders are sharp. Thus, compared with [33], over-smoothing in salient image regions is reduced due to the additional salient weight in the discontinuity preserving smoothness term. Contrary to [33], StereoBrush is implemented as a multi-scale GPU implementation and, thus, the input does not have to be scaled down. Moreover, the user immediately receives feedback on a per keyframe level.

The authors of [47] present an efficient and simple method to solve the global optimization problem by separating the data from the smoothness term and thus solving or approximating the optimisation more efficiently. The separated terms are then solved in series, where the smoothness term is approximated by a filtering operation. Thus, it can be used in various image-based optimization problems as segmentation (e.g., [11]) or depth propagation (e.g., [40], [47]).

In [47] the authors replace the smoothness term by an efficient edge aware filtering operation. The special feature of this edge aware filter is the temporal filtering approach. The authors are using a box filter (median filter) implementation that also filters temporally in addition with motion paths. These paths are computed by following optical flow vectors from frame to frame. The box filter filters analogue to the spatial direction temporally along these motion paths. More about this filtering technique is presented in Section 3.3. Moreover, this approach is used similarly in the algorithm proposed in this thesis (cf. Chapter 4). Prior to that approximation of the smoothness term, the data term enforces the given scribble map, defined by the user. Each pixel is related to one of the given scribble disparities due to colour similarities.

The authors apply their method on different applications such as disparity estimation, depth upsampling, and colour and scribble propagation (cf. Figure 2.5). According to the authors of this implementation, in the case of scribble propagation one keyframe for every 20 frames is sufficient to propagate depth to a video. Since one of the user-defined disparity values is going to be assigned to the final map, there is no disparity variation within objects, which results in a cardboard effect. Furthermore, the implementation of [47] does not enable disparity changes over time. However, due to the temporal filtering approach, temporal consistency is improved.

2.4 Segmentation-based Methods

Since segmentation is closely related to 2D-to-3D conversion, there are various approaches exploiting different types of segmentation approaches. While segmentation partitions an image or

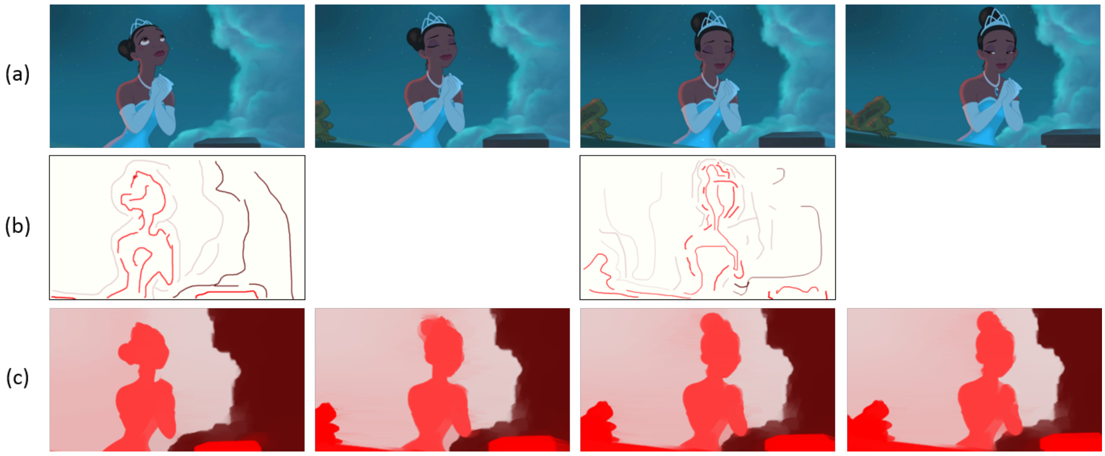


Figure 2.5: Colour scribble propagation from [47] over a video with 35 frames. (a) source images, (b) scribble maps on keyframes and (c) colour propagation. [47]

a video into homogeneous segments due to a certain feature space, e.g. colour, 2D-to-3D conversion identifies parts in a video related to their position in depth and assigns depth values to each pixel. More about segmentation and used techniques is discussed in Chapter 3. Basically, segmentation-based conversion methods consist of two steps: First, the video is segmented into objects or regions. Second, segments are assigned to depth throughout the whole video. In the following we discuss two segmentation-based methods and how they implement the above mentioned steps.

We refer to the algorithms from [18] and [87] as *interactive segmentation-based 2D-to-3D conversion methods*. Both are working very similarly but differ in the context of depth propagation to non-keyframes. In total there are four steps:

1. Selecting keyframes.
2. Performing a segmentation of objects and areas in the chosen keyframes.
3. Assigning depth values to the segmented areas.
4. Propagating depth from keyframes to non-keyframes.

After choosing keyframes, both algorithms ([18], [87]), are using an image snapping tool called Lazy Snapping [49] to perform an interactive object segmentation. Since this tool initially is used to segment only one foreground object from the background on a single frame, the authors of [18] and [87] adapted the method. The use of this tool results in multiple segmented objects in the chosen keyframes. The user assigns disparity to each segmented object with scribbles. The authors of [18] additionally provide pre-set disparity models such as for planar, spherical or cylindrical surfaces and objects. Up to this second step, both algorithms are basically doing the same, but vary now in the following depth propagation step.

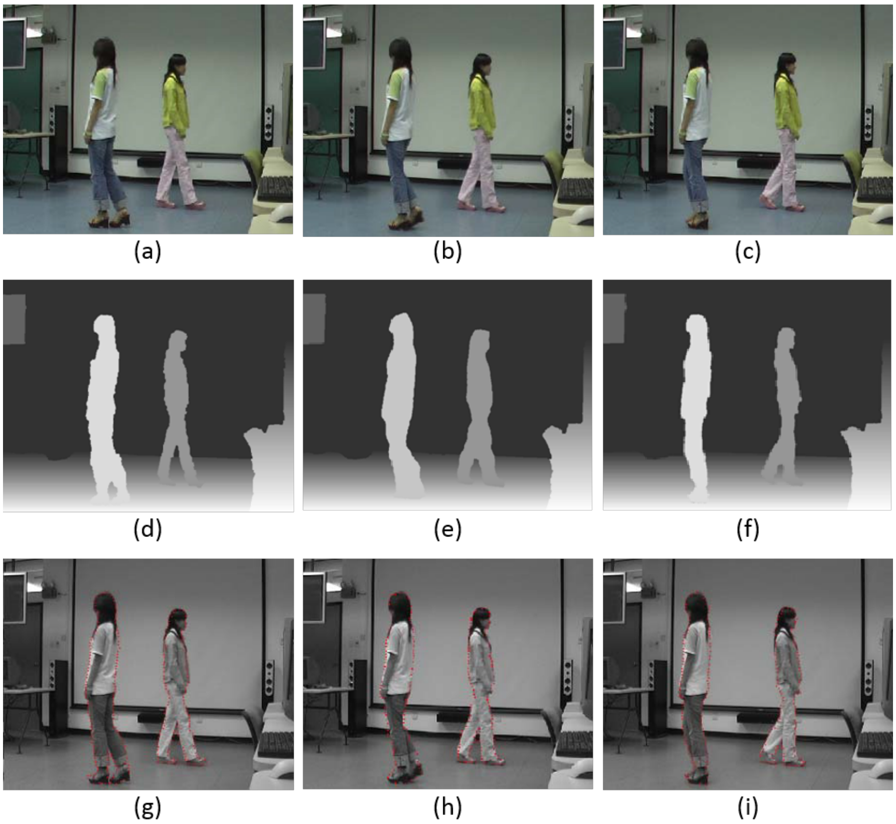


Figure 2.6: The resulting depth maps from [87]. (a) shows keyframe 1, (b) frame 5, (c) keyframe 10, (d) and (f) the result of keyframes using the segmentation tool, (g) and (i) KLT feature points for tracking, (h) the tracked points in frame 5 and (e) the final depth map of frame 5. [87]

In [87], an additional tracking algorithm between each pair of keyframes is performed to track the contours of each object. Therefore, the authors adopted an algorithm which is based on the *Kanade-Lucas-Tomasi* (KLT) feature tracker [73]. That way, a segmentation of the objects throughout the whole video is possible. Since the user assigns depth on keyframes for each object, these depth values can be carried on to non-keyframes. In order to model depth changes, the depth is linearly interpolated between two keyframes. On the contrary, [18] propagates depth based on the video filtering method from [83], which uses bilateral filtering in combination with optical flow vectors. Instead of using just one keyframe as input like in [83], the adapted method in [18] combines the propagated depth values of the two surrounding keyframes weighted by distance to the nearest keyframe.

In [87], the quality of the resulting depth maps (cf. Figure 2.6) depends on the result of the tracking method. However, the used tracking algorithm in [87] has difficulties with camera motion, occlusions between objects and background changes. According to the authors, an improved tracking algorithm would lead to improved quality. The authors point out that for a

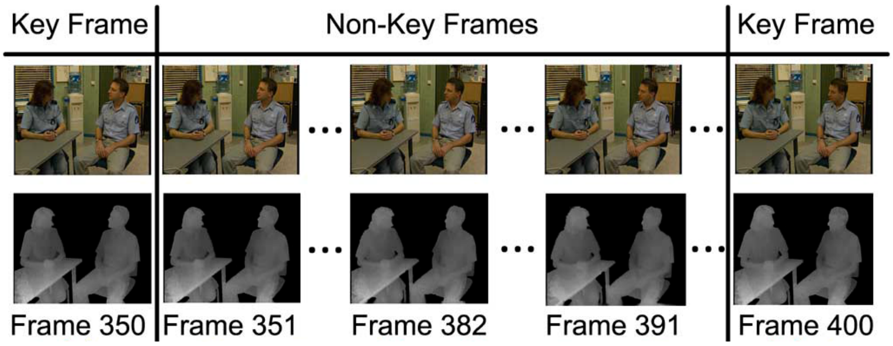


Figure 2.7: Resulting depth maps from [18]. The first and last frames are the keyframes. Frames in between are the result of the propagation. [18]

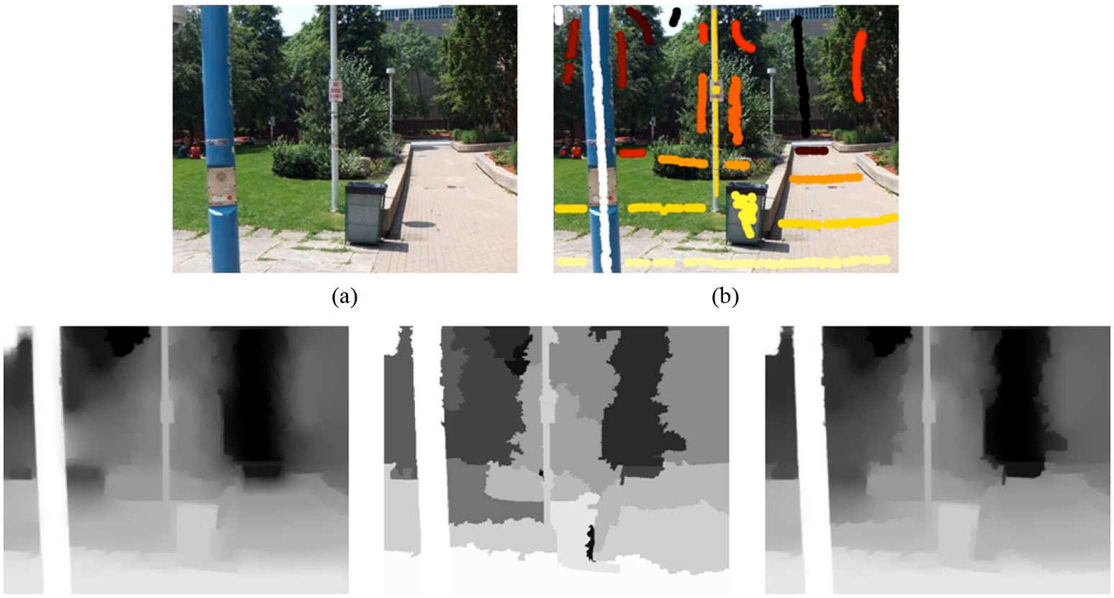
good quality conversion every tenth frame has to be annotated. Hence, there is still a big amount of work to be done by the user. Moreover, since each segmented object is assigned with one depth value the resulting depth map suffers under the cardboard effect. The main advantage of [18] and [87] is the hard segmentation of the objects which provides clear boundaries.

In [18], the cardboard effect is prevented by the pre-set depth models, which are assigned to the segmented objects (cf. Figure 2.7). Furthermore, first and last frame as keyframes are acceptable for a video with 50 frames because of the used video filtering method. Although the user can use the adapted snapping tool to segment the objects, it is still time-consuming.

To overcome these problems, combined approaches have been proposed, e.g., [66]. The authors of [66] use two *interactive image segmentation algorithms*, Random Walks [31] and Graph Cuts [8], [9], to perform a 2D-to-3D conversion which produces high quality depth maps. Similar to [12], this fusion algorithm is a graph-based conversion.

Random Walks and Graph Cuts are two image segmentation algorithms which treat an image as graph, representing each pixel as a node connected by weighted edges. The weight depends on the similarity between two nodes. In the case of an image, the graph is 4-connected and is extended to a 6-connected graph for video in [66]. Therefore, each node additionally gets an edge to the pixel in the same spatial location in the previous and next frame. In the fusion framework, the user marks objects with depth assigned strokes as closer or farther from the camera. To combine the two mentioned algorithms, the result of the Graph Cut method acts as input for Random Walks.

The Graph Cuts algorithm was intended to solve a binary classification problem by minimising an energy function. Each pixel is assigned to either the foreground or the background. For the depth propagation each user-defined scribble represents a label and is defined as a binary classification problem. To solve this multi-label classification problem, the Graph Cuts algorithm is performed for each user-defined label. Hence, in each iteration the current label is assigned to the foreground while all other labels remain in the background. After performing all Graph Cuts iterations, each pixel is assigned to the label corresponding to the least amount of required energy. It might happen that a pixel was not assigned to any label. Such a pixel was



(a) (b)

(c) (d) (e)

Figure 2.8: Resulting conversion from [66]. (a) shows an example image, (b) an example image with user-defined scribbles, (c) the depth map using modified Random Walks algorithm, (d) the depth map using modified Graph Cuts algorithm and (e) the combined depth map from (c) and (d). [66]

never part of a foreground object in one of the Graph Cuts iterations. In that case, region filling methods are used to correct those missing values. The algorithm results in a hard segmentation with clear object boundaries and no depth variations within the regions (cf. Figure 2.8(c)), which appear as an undesired cardboard effect.

The Random Walks method is also a graph-based optimization algorithm [31]. It starts at a randomly chosen node in the graph and walks around visiting all unlabelled nodes. To this end, the weight of the edges influences the decision where to go. Since the weight depends on the similarity between nodes, it is more probable to visit similar nodes. The goal is to assign each node to one of K possible labels. Each label represents one of the user-defined depth values. The algorithm calculates the probability that each pixel belongs to one of these labels by solving a linear system of equations [31], [66]. The depth values resulting from the optimisation can differ from the user-defined values. On one hand, this can eliminate the cardboard effect but on the other hand leads to mixed depth values at object boundaries (cf. Figure 2.8(d)).

By using the Graph Cuts result as input for the Random Walks iteration, the result is a depth map with hard object boundaries and smooth depth variation within the regions (cf. Figure 2.8(d)). To convert videos, the user-defined labels are tracked throughout the whole video by the use of one of two proposed methods. First, there is an object tracking approach. Thereby, with a tracking method, the segmented objects are tracked and their depth values are assigned along the trajectories, which describe the movement of the object from one frame to the next throughout the whole video. However, the authors of [66] point out that this approach is very time consuming. Thus, another method is offered. This less computational alternative computes the optical flow, i.e., the movement of each pixel from one frame to the next one. Again the depth is assigned along the optical flow vectors from one frame to the next. According to the authors of [66], this method achieves good results for videos with horizontal motion and is therefore used instead of the more computationally demanding object tracking approach. However, labelling every frame would produce the best results but is user-intense and time-consuming.

2.5 Summary

In this final section, the above presented algorithms are briefly summarized and discussed once more with regard to their quality and user effort, starting with the filtering-based techniques. Since every used keyframe has to be fully assigned with depth values, filtering-based techniques require the most work done by the user. To determine depth values, some algorithms (e.g., a stereo algorithm from [6]) can be used. However, this approach needs more time than drawing scribbles. Additionally, depth variation within objects can be modelled and the undesired cardboard effect is avoided. In [18], the extended version of the filter-based algorithm from [83] enables the modelling of depth changes. Difficulties for this type of algorithms are new uncovered regions and colours, which can be handled with the use of more keyframes at the expense of increasing user input.

The machine learning algorithm from [36] is less time-consuming for the user than filtering-based (each keyframe fully assigned with depth values) and manual conversion (each frame fully assigned with depth). For a video with 43 frames the user has to define about 8000 samples on three keyframes to achieve good quality. The definition of different depth-defined samples within an object avoids the cardboard effect. However, this algorithm does not model smooth depth changes over time.

The most complex 2D-to-3D conversion methods are the optimization-based algorithms [33], [47], [86]. They solve the complex optimization problem by separating the data and the smoothness term. Thus, they present a more efficient algorithm. Optimization-based algorithms score with minimal user input, since only some user-defined scribbles on a few keyframes are necessary to achieve good quality. With the use of keyframes in the first and the last frame the authors of [33] are able to model depth changes over time. The ability for depth variations depends on the optimization constraints and the framework. Hence, the algorithm of [33] is able to avoid the cardboard effect, while the resulting depth maps of [47] and [86] suffer from these missing depth variations. However, [86] can avoid over-smoothed regions around the object edges.

The time effort of segmentation-based methods depends on the chosen segmentation technique. The presented interactive algorithms [18], [87] are time-consuming, because the segmentation is done more or less manually with a segmentation tool. However, the graph-based [66] approaches do need less time to initialise the segmentation. Users have to mark the desired depth value on keyframes with a few scribbles. According to the authors of the graph-based fusion algorithm [66], every frame has to be labelled in order to achieve high quality results. The ability of modelling depth variation within objects and depth changes over time and furthermore depends on the applied propagation step. Thus, both of the interactive segmentation techniques, i.e., [18] and [87], are able to use depth changes, since their propagation is filtering-based [18] or uses a tracking technique [87]. Hence, especially filtering-based methods [18] enable depth variation within objects. The graph-based fusion method from [66] is able to avoid the cardboard effect and provides depth changes over time.

Concluding, it seems that the segmentation-based methods [18], [66], [87], together with the optimization-based algorithms [33], [47], [86] require the least user input while achieving good quality results. Regarding to the segmentation-based conversions, the combination of the segmentation step and the propagation step is crucial for the quality of the results. For this reason and since the proposed 2D-to-3D conversion algorithm implemented in this thesis is based on segmentation, segmentation is discussed in more detail in the following Chapter 3.

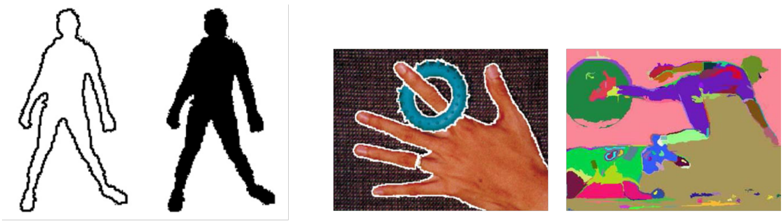
CHAPTER 3

Principles of Video Segmentation and Cost Volume Filtering

It can be concluded from previous chapters, e.g., Chapter 2, that video analysis and segmentation can be important in 2D-to-3D conversion. Generally, segmentation is one of the first steps in a lot of image processing and computer vision applications such as depth propagation (e.g., [18], [66], [87]) and tracking (e.g., [43], [73]). The goal of segmentation is to partition a given image or video into parts, which are referred to as *segments* or *regions*. These segments can be represented in different forms, which are discussed in Section 3.1. Segments, typically, are homogeneous in a certain feature space, e.g., only contain pixels with similar colours. Additional to colour, Section 3.1 also discusses other features such as motion. In Section 3.2, we discuss (interactive) image and video segmentation techniques (e.g., [23], [43], [59], [72], [88]) that are related to the 2D-to-3D conversion approach that is proposed in this thesis. This discussion also covers other relevant video analysis techniques, such as motion estimation (e.g., [13], [48]) and tracking (e.g., [73]). Finally, Section 3.3 puts emphasis on a label-based optimization framework, namely cost volume filtering (CVF), which will be exploited in our proposed 2D-to-3D conversion approach (cf. Chapter 4). Specifically, we discuss the general CVF-framework and some filtering techniques such as the guided filter [37] as well as CVF for the special case of image segmentation.

3.1 Representation and Features for Segmentation

The result of an image or video segmentation algorithm is commonly represented either by contours or regions. A *contour* (cf. Figure 3.1(a) and 3.1(c)) defines the boundaries of segments while a *region* (cf. Figure 3.1(b) and 3.1(d)) represents the area inside those boundaries. The choice of representation depends on the segmentation technique and the further application as well as the used features to detect the different regions in an image or video. According to [35],



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

Figure 3.1: (a) shows an object contour representation [90], (b) an object region representation [90], (c) a segmentation by the mean shift technique using contours [79], and (d) a segmentation by a graph-based merging technique using regions [79].

contours and regions should be simple. This means, while contours should not be ragged, regions should not contain many holes.

Independent of the chosen representation, segments are typically homogeneous in a certain feature space [90]. There are different features of an image or video that can be used to perform a segmentation. Examples of well-known features are colour, edges, motion and texture, which are disussed in the following.

Colour

Colour is a property of the surface of each object in an image and influenced by two physical factors. First, the distribution of the illuminant and, secondly, the reflectance properties of the object [90]. The combination of these two attributes makes colour sensitive to illumination variation in an image or video. When using different colour spaces, e.g., the RGB or HSV space, it is possible to explore different aspects of colour such as brightness, saturation or hue. With colour transformations it is possible to convert an image given in one colour space to another colour space.

Most commonly, the RGB colour space is used [64], where a colour is represented in its components of red (R), green (G) and blue (B). However, according to the human visual system this colour space is not perceptually uniform [64]. In other words, the colour difference perceived by humans does not correspond to the difference between the colours in the RGB space [64]. Another well known colour space is HSV, which represents colour in terms of hue (H), saturation (S) and value (V), and is an approximately uniform colour space [64]. Hue refers to the perceived colour, e.g., red, while saturation is used to specify the purity of the colour, and the value parameter defines the intensity of the colour. Due to these three components it is more intuitive for humans to work with such a colour space. The CIE L*a*b* [57] and CIE L*u*v* [20] colour models were introduced to better assess perceptual differences among colours. These spaces are designed in a way that by using a simple Euclidean distance it is possible to calculate perceptual colour differences [55].

Edges

Edges are a salient feature in images. An edge can be described as an area in an image where differences of colour, intensity or texture occur [79]. The larger these differences are, the ‚stronger' the edge is perceived by the human observer. This fact can also be used in computer vision to filter out weak and potentially not important edges. Subsequent to the detection of edges in an image, those edges can be linked together into continuous contours, which ideally segment an object in an image. Compared to the feature colour, edges are less sensitive to changes in illumination [90]. Since an illumination change usually has impact on the whole image, the values of, e.g., intensity are shifted in all parts and, thus, discontinuities remain.

There are several approaches to detect edges in images (e.g., [17], [19], [24]). Most of them locate abrupt changes in image intensities and have been developed for greyscale images. Edge detection results of each independent colour channel can be combined to detect edges in colour images or videos [79]. Greyscale edge detectors often use gradient functions (e.g., [24]). Commonly used edge detectors are the Sobel and Prewitt operator [24], or the Canny edge detection approach [17].

Motion

Motion is a feature often used in motion-based or tracking applications. In the case of tracking, motion is used to follow selected objects throughout the video (e.g., [87]). This is especially important when segmenting videos and will, in this context, be discussed more explicitly in Section 3.2 and Section 3.3.

A well-known approach to estimate motion from a video is optical flow estimation. *Optical flow estimation* is described as the computation of a dense field of motion vectors of each pixel. These vectors contain each pixel's motion direction and motion velocity in regard to its new position in the next frame. Optical flow estimation approaches determine pixel motion between consecutive frames based on the *brightness constancy constraint*: $I(x,y,t) - I(x+dx, y+dy, t+1) = 0$ [38]. This constraint assumes that the intensity $I(x,y,t)$ of a pixel (x,y,t) does not change between frames. Hence, the intensity $I(x+dx, y+dy, t+1)$ in the following frames stays the same. Two well-known optical flow estimation methods are the Horn-Schunck formulation [38] and the Lucas-Kanade method [54]. Horn-Schunck [38] deals with the problem of solving two unknown terms (direction and velocity) for each pixel *globally* by minimising an energy function with an additional smoothness term. Contrary to the global method of [38], the Lucas-Kanade method [54] is a *local* approach. It solves the optical flow equation by combining information from local pixel neighbourhoods and minimising the energy function by using Least-Squares Estimation (LSE) [5]. While local methods such as Lucas-Kanade are often more robust under noise, they do not compute a dense flow field. Contrary, global methods, e.g., Horn-Schunck, give a dense flow field. However, the computation is more expensive and is more sensitive to noise [14]. For an overview of more recent optical flow estimation approaches, interested readers are referred to [70]. Based on the brightness constancy assumption and additional assumptions, such as global smoothness, optical flow estimation methods can compute motion vectors. The resulting motion vectors can be used for, e.g., tracking contours, recognizing moving objects or segment objects due to similar motion.

Texture

Texture refers to the properties of the surface or structure of an object [76]. Texture can be described as a measure of the intensity variation of a surface [90]. A particular texture is described by a texture descriptor, which models its intensity variations and contains a deterministic or probabilistic rule to classify the texture. An example of descriptors are Grey-Level Co-occurrence Matrices (GLCMs) by [34], which capture co-occurrences of intensities in a specified direction and distance to describe and recognize texture in an image. A texture can be described, for example, as fine, coarse or smooth, rippled, moulded, irregular or lined [34]. Like edges, texture is less sensitive to illumination changes than colour [90].

3.2 Segmentation

From Section 2.4 can be concluded that segmentation and 2D-to-3D conversion share some similarities. The goal of segmentation is to partition the image or video into segments that are homogeneous in a certain feature space (e.g., colour) [76]. A fixed label is assigned to each segment. In 2D-to-3D applications the video can also be grouped into segments. However, 2D-to-3D conversion identifies parts in a video related to their depth position and assigns depth values to each pixel. Furthermore, those depth values can vary inside an object due to roundness or skewed surfaces. As discussed in Section 2.4, many 2D-to-3D conversion approaches use segmentation techniques to partition the image or video and further use other techniques or user interaction to assign depth to the objects.

Segmentation techniques can be classified into *spatial methods*, which work on images, and *spatio-temporal methods*, which are applied to videos and result in segments that extend over time. The latter are often based on spatial approaches to detect segments in individual frames and further track them from frame to frame. Other approaches analyse the whole video at once to perform a segmentation.

Image Segmentation

Image segmentation methods can be classified into region-based techniques (e.g. [23], [72], [88]) and contour-based techniques (e.g. [43], [59]). While region-based methods use features such as colour to group neighbouring pixels into regions, contour-based methods typically exploit edges.

In the group of region-based approaches, the mean shift algorithm and graph cuts approaches are popular. The *mean shift* algorithm ([23], [29]) finds peaks in the multidimensional spatial and colour space $[l, u, v, x, y]$. Here $[l, u, v]$ are the colour components of the earlier mentioned CIE L*u*v* colour model and $[x, y]$ is the spatial position of a pixel. First, cluster centres, which at the end represent the peak of a found cluster, are randomly defined over the discrete data. During each mean shift iteration, an ellipsoid is positioned at the cluster centre and examines the mean of the data inside the ellipsoid. Next, the centre is moved to the mean value [90]. This process is repeated until the cluster centre does not change the position anymore and thus, a peak is found. It may happen that during the iterations some cluster centres are merged. Finally, every cluster centre represents a peak in the multidimensional space and is defined as a segment.

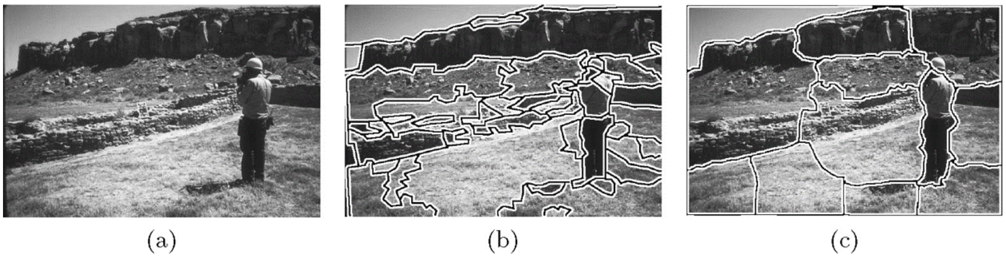

(a) (b) (c)

Figure 3.2: Segmentation of the image in (a), by using mean-shift in (b) and using normalized cuts in (c) [90].

All points that belong to that segment can be identified by taking all points from the input space that climb to the segment's peak. As a result, similar pixels in colour and space are part of the same segment. For a resulting segmentation see Figure 3.2(b). It can be seen that the resulting segments are similar in colour (and intensity). The algorithm works automatically, thus, the user only sets a few parameters to obtain a better segmentation.

In another region-based approach [72], the authors are treating the segmentation problem as a graph partition problem. In their approach of the *normalized cut*, a node represents a pixel in an image. Each node is connected to its neighbours by an undirected weighted edge. The weight determines the similarity between two nodes regarding the used feature space, e.g., colour or texture similarity. The goal is to partition the graph into disjoint subgraphs by pruning the weighted edges to generate subgraphs that are not similar enough. A *cut* is defined as the total weight of the pruned edges between two subgraphs [79]. In [88], the authors' segmentation is based on finding the minimum cut value that divides the graph. This approach leads to *oversegmentation*, i.e. many small segments. Thus, the normalized cut was further developed in [72]. In [72], both the similarities and dissimilarities between different groups of nodes are studied to separate groups that are not similar enough. The value of the cut not only depends on the total weight of pruned edges but is also computed as part of the total edge connections to all nodes in the graph [72]. The result of a normalized cuts segmentation is shown in Figure 3.2(c). As can be recognised compared to the mean shift algorithm (cf. Figure 3.2(b)), the normalized cut method leads to less oversegmention. The method of [72] requires only a few parameters and works automatically.

The concept of cost volume filtering (CVF) can also be used to solve the segmentation problem, e.g., with the interactive object segmentation of [11], which is actually a video object segmentation approach. After an interactive step by the user, a foreground and background segmentation is computed. With a CVF approach the result is generated in a temporally-coherent way to perform a binary segmentation. CVF and the video object segmentation algorithm of [11] are discussed in more detail in Section 3.3.

The second group, contour-based approaches, detect object boundaries based on image-domain based properties such as edges. An example are *active contours* [79]. Active contours may be initialised by user annotations around the boundaries of an object. In [43], the resulting closed contours which surround the object region are called *snakes*. After the initialisation the algorithm seeks for a *spline*, which is a function that piecewise defines a polynomial curve [84].

The method from [59] works interactively and snaps the contour to boundaries of an object, while the user is still moving and drawing rough boundaries around an object. To this end, the image is first pre-processed to link low costs with edges which are likely to be part of a boundary. The algorithm continuously seeks for the lowest cost path between the starting point and the current end position of the user drawn path.

Video Segmentation

Video segmentation partitions a video into segments, which are uniform in a feature space, e.g., motion. There are two common approaches to perform a video segmentation: (i) the segmentation is done frame by frame or (ii) the whole video is analysed and processed at once. In both cases motion estimation can be incorporated to achieve temporally consistent results. A well-known technique is the already discussed optical flow estimation (cf. Section 3.1), which computes optical flow vectors of each pixel in consecutive frames. Furthermore, *tracking* is used to estimate motion of points, contours or regions. Since tracking is a challenging problem there are a lot of different approaches and so this thesis can only provide a short overview of some selected methods. A good overview of object tracking algorithms can be found in [90].

The goal of tracking is to locate an object's position in every frame of the video. The object may be represented in the form of different models such as points, contours or regions. As discussed in Section 2.4, the authors of [87] used an adapted version of the KLT tracker [73]. The tracker iteratively computes the translation of a region centred on an interest point in consecutive frames. In the case of [87] the object's contour is used to detect interest points and to track them to successive frames. Afterwards, the object's contour is recovered by using *snakes* [43], the earlier discussed active contours algorithm. In that way, objects and their segments are tracked frame by frame through the video.

The *snakes* algorithm [43] can also be used for segmentation without any additional tracking method. Therefore, the resulting snake from one frame is used as rough contour in the consecutive frame. Thus, an object's contour is tracked through the whole video.

CVF was already mentioned in the discussion of image segmentation algorithms. The concept can also be used to solve the segmentation problem in the temporal space, e.g., [11]. After sparse user input, an object segmentation is processed frame by frame in a temporally coherent way. The concept of CVF and the approach of [11] are discussed in the following Section 3.3.

The second approach is to analyse and process the whole video at once. Graph-based approaches such as [26] can be extended to video. Therefore, nodes which represent pixels in a video frame are connected by *edges* not only to their spatial neighbours but also temporally, i.e., an edge connects a pixel to the pixel at the same location in the previous and next frame. In [32] the authors further improve the temporal component by using dense optical flow vectors to temporally connect the nodes. Other approaches (e.g., [13], [48], [60]) use optical flow to generate long-term motion paths and enhance a segmentation with this additional information. *Long motion paths* are built by following the optical flow vectors from frame to frame (as discussed in more detail in Section 3.3). The long motion paths are used to group paths according to similar motion. By taking not only neighbouring paths into account but also all similar paths in the whole video, objects are rediscovered after an occlusion occurred [48], [60].

3.3 Cost-Volume Filtering

Interactive segmentation can also be formulated as a labelling problem, where the aim is to assign each object a different label. Therefore, the user predefines some labels on objects, e.g., with scribbles. Interactive segmentation solves the problem of assigning each pixel to one of these user-defined labels. Not only segmentation, but also other computer vision applications can be formulated as label-based including stereo matching [40], optical flow estimation [40] or colourisation [47].

A common approach to solve such problems is by minimising an energy function, where the label costs are expressed in a data term E_{data} and the spatial smoothness with edge-aligned label changes is enforced by a smoothness term E_{smooth} for the unknown solution J. Without smoothing the cost volume the result would be noisy, as can be seen in the case of a stereo matching application in Figure 3.3(b).

$$E(J) = E_{data}(J) + \lambda * E_{smooth}(J) \tag{3.1}$$

This energy function can, e.g., be solved by global minimisation techniques [40]. Such global methods are, typically, computationally expensive. The main idea of CVF is to overcome this problem by splitting up the two terms and replacing the smoothness term by an efficient edge aware filtering operation.

More precisely, the authors of [40] propose three steps to efficiently solve multi-labelling problems: (i) constructing a cost volume, (ii) fast cost volume filtering and (iii) winner-take-all (WTA) label selection. This means, the final selection of the label is done by choosing the label with the lowest cost at each pixel.

Consider a labelling problem that aims to assign each pixel i with coordinates (x, y) to a label l from a set of all labels $\mathfrak{L} = (1, ..., L)$. After computing the costs of each pixel i for choosing label l, the cost volume C is a three-dimensional array (x, y, l).

In a second step the L slices of the cost volume C are filtered. The filtering step locally smooths the label costs spatially or spatio-temporally, when a temporal filter is used. Thus, the solution is regularised and higher quality in the final result is achieved [40] as can be seen in Figure 3.3(c) and (d). For this purpose, different filters can be used. In the following, three filters are discussed: At the beginning, the box filter [15] and an extended version that enables temporal filtering following the motion of points between frames, as done in [47], are presented. Then, the guided filter [37] as an extended spatio-temporal implementation like the authors of [40] used it, is discussed.

Box Filter

The *box filter* is a linear filter which computes the average of pixels in a $K \times K$ window in order to smooth the image and remove noise. More precisely, the box filter is applied on the neighbourhood of a given pixel $i = (x, y)$ in the input image I and computes a weighted sum in order to define the new filtered output value in the output image G [79].

$$G_i = \sum_k I_{i+k} H_k. \tag{3.2}$$

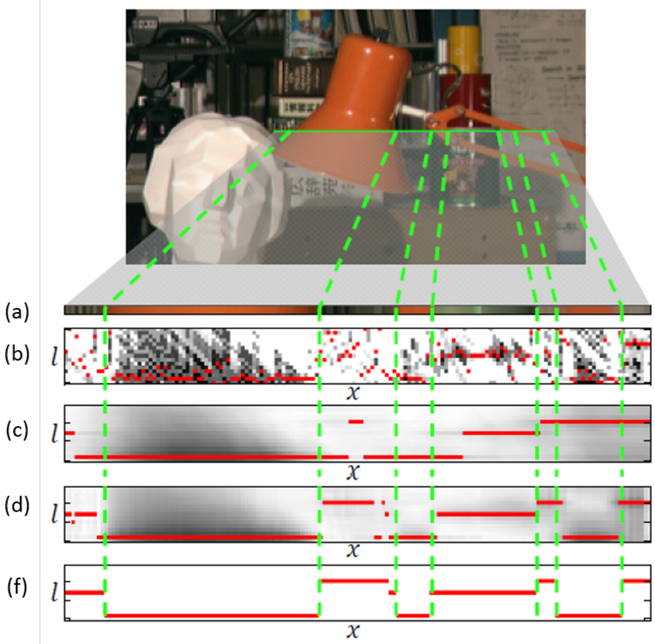


Figure 3.3: Cost volume filtering used for stereo matching by [40]. (a) Zoom of the green line in the input image, (b) slice of cost volume (white/black/red: high/low/lowest cost) for line in (a), (c) and (d) cost slice smoothed along x and y-axes (y is not shown here) with box filter and guided filter [37], respectively, (f) ground truth labelling. [40]

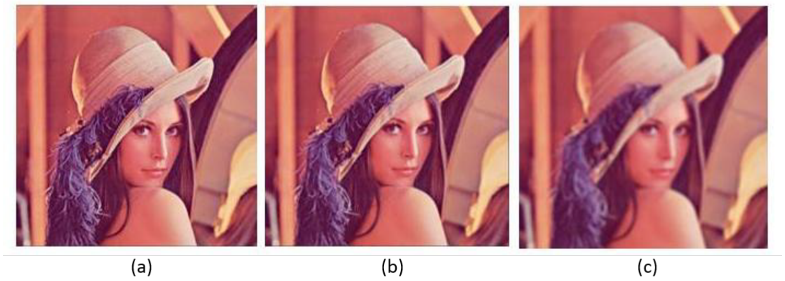


Figure 3.4: Linear filtering with a box filter: (a) input image, (b) filtered with a 3×3 box filter and (c) filtered with a 8×8 box filter. Because of the larger filter kernel in (c) the image is smoother, but diffuse.

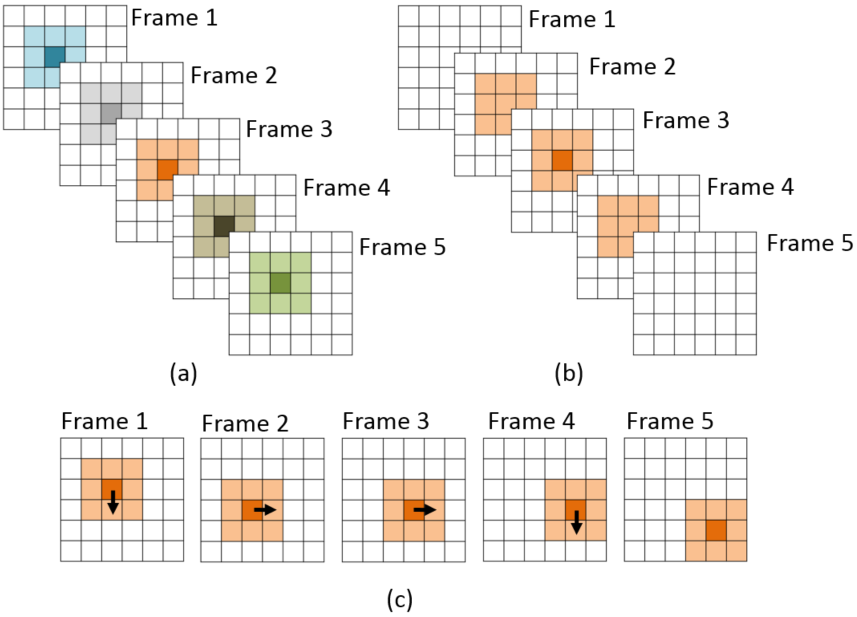


Figure 3.5: Box filtering of a video with 5 frames in various versions. (a) shows a spatial box filter in two dimensions where each coloured kernel is a separate filter operation with a 3×3 filter kernel. (b) shows one filter operation of a spatio-temporal box filter in three dimensions with a $3 \times 3 \times 3$ filter kernel. (c) shows one filter operation of a motion-based box filter in three dimensions along long motion paths.

The size and the weights of the *filter kernel* H_k specify the effect of the filter. $k = (m, l)$ is the weight at position (m, l) in the filter kernel. The entries of the filter kernel, the *filter coefficients*, of a box filter are all equal [15], e.g. a 3×3 box filter kernel looks as follows:

$$H_k = \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix} = \frac{1}{9} * \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}. \tag{3.3}$$

Each of the pixels in the filter kernel receives a weight of $\frac{1}{9}$. In that way, the box filter smooths the input image. The larger the filter kernel is, the smoother is the output image (cf. Figure 3.4).

The two-dimensional filter kernel $K \times K$ can perform a spatial filtering (cf. Figure 3.5(a)). To extend this to the spatio-temporal space, a three-dimensional filter kernel $K \times K \times K$ is needed (cf. Figure 3.5(b)). By doing that, the frames of a video can be temporally filtered. With this extension, not only spatial neighbours, i.e., pixels from the current frame, but also temporal neighbours, are included in the weighted sum for the output value.

The box filter can be used to remove details in an image, e.g., noise. However, these details may also include edges which we want to preserve. Edge-preserving filers such as the guided filter [37] can be applied to preserve edges.

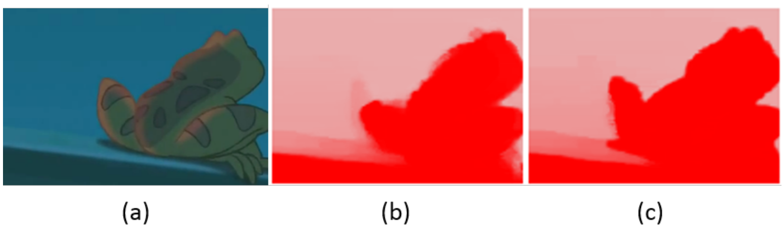


Figure 3.6: Filtering of scribble propagation results: (a) the colour input image, (b) the result of the spatio-temporal box filter, which filters straight through the video and (c) the result of the extended motion-based box filter, which filters along long motion paths. As can be seen, the regions around the edges are clearly improved in (c). [47]

Motion-based Box Filter

The authors of [47] present an extended box filter which filters the temporal direction in a different way than the above presented spatio-temporal box filter. More precisely, the temporal filtering step follows the motion of points between frames. The main motivation of this approach is to prevent information from being incorrectly averaged across object boundaries in order to achieve improved results [47].

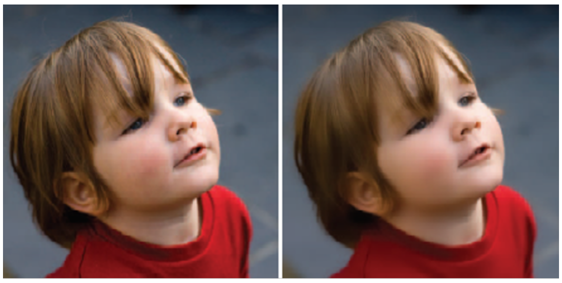
The *motion-based box filter* is based on *long motion paths*, which are generated using optical flow fields. Such a path is defined as the sequence of vectors of pixels that correspond to the motion of one point over time [47]. It is computed by following the optical flow vector from one frame to the next. A path ends, when one of the following three cases occurs [47]:

1. The path has to end, if it leaves the scene.
2. If multiple paths converge on the same pixel, just one of them continues (picked randomly).
3. If a pixel does not belong to a path in the previous frame, a new path starts.

Considering these properties, every pixel belongs to exactly one path. The motion-based box filter is following the paths to filter in the temporal direction (cf. Figure 3.5(c)). Compared to the common spatio-temporal box filter, which filters straight through the video volume, the edge regions are improved by using the motion-based box filter (cf. Figure 3.6). Thus, the motion-based box filter can preserve temporal edges.

Guided Video Filter

The *guided filter* was first introduced for images [37] to perform an effective and efficient edge-preserving smoothing operation. The filtered output is generated by considering the content of a given guidance image [37]. The guidance image can either be the input image itself or another image. The filter kernel of the guided filter can be extended to a spatio-temporal kernel [39] to use the filter for temporal video filtering as well.



(a) (b)

Figure 3.7: Guided Image Filter with a colour guidance image. (a) input image and guidance image and (b) colour-guided result. [37]

The filtered output q_i at pixel $i = (x, y)$ is a weighted average [37]:

$$q_i = \sum_j W_{i,j}(I) p_j, \tag{3.4}$$

where i and j are pixel indeces, $W_{i,j}$ the guided filter kernel. I is the guidance image and p denotes the input image to be filtered. The spatio-temporal filter kernel [39] with coloured guidance image I is defined as [39]:

$$W_{i,j} = \frac{1}{|w|^2} \sum_{k:(i,j)\in w_k} (1 + (I_i - \mu_k)^T (\Sigma_k + \epsilon U)^{-1} (I_j - \mu_k)) \tag{3.5}$$

I_i and I_j are 3×1 colour vectors. μ_k and Σ_k are the mean and covariance matrix of I and p in a temporal window w_k with dimensions $w_x \times w_y \times w_z$, centred at pixel k. The mean μ_k is a 3×1 vector, while the covariance matrix Σ_k and the identity matrix U are of size 3×3. The number of pixels in the window is denoted $|w|$ and ϵ is a smoothness parameter. The filter weights can be implemented by determining some linear coefficients a_k and b_k and computing the filter output q:

$$a_k = (\Sigma_k + \epsilon U)^{-1} (\frac{1}{|w|} \sum_{i \in w_k} I_i p_i - \mu_k \bar{p}_k), \tag{3.6}$$

$$b_k = \bar{p}_k - a_k^T I_i \mu_k, \tag{3.7}$$

$$q_i = \bar{a}_k^T + \bar{b}_i. \tag{3.8}$$

Here $\bar{p}_i = \frac{1}{|w|} \Sigma_{i \in w_k} p_i$ is the mean of p in w_k, $\bar{a}_i = \frac{1}{|w|} \Sigma_{i \in w_k} a_k$ and $\bar{b}_i = \frac{1}{|w|} \Sigma_{i \in w_k} b_k$. All summations ($\Sigma_{i \in w_k} f_i$) in these equations are spatio-temporal box filters.

With the guided filter it is possible to transfer structure to the output image q, even if there is no structure in the input image p, but in the guidance image I. With this property the guided filter is used in different applications like in stereo matching [39], alpha matting [37], HDR compression [37] or to denoise a no-flash image under the guidance of its flash version [37].

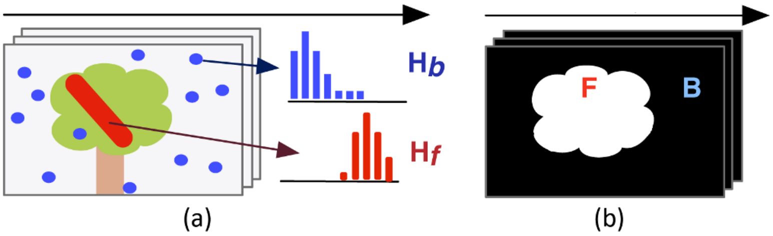


Figure 3.8: Histogram computation by [11]. (a): the foreground histogram H_f based on the user-marked pixels (red scribble) and the background histogram H_b based on randomly chosen background samples (blue dots). (b): the resulting object segmentation, the foreground F (white) and the background B (black). [11]

In Figure 3.3(c) and (d) the authors of [40] smoothed the cost volume of a stereo matching application with various filters to demonstrate the importance of the filter choice. As can be seen in Figure 3.3(b), the unfiltered cost volume slice is noisy and the lowest costs (red points) are not consistent within regions. The cost volume filtered by the box filter is shown in Figure 3.3(c). The resulting lowest costs are smooth, but the edges in the image are blurred. Finally, the authors of [40] applied the guided filter [37] in Figure 3.3(d), which, as mentioned above, has edge-preserving properties.

Finally, after applying a filter to L slices of the cost volume C, the filtered cost volume is denoted as C'. In a last step, the assigned label f_i is chosen as the label l with the lowest cost (e.g., the highest probability) at pixel $i = (x, y)$.

$$f_i = \arg\min_{l} C'_{i,l}. \tag{3.9}$$

In the case of object segmentation, e.g. [11], the result is a segmentation where each pixel is assigned to either the foreground or the background. The video object segmentation algorithm of [11] is discussed in the following in more detail.

Video Object Segmentation via Efficient Cost-Volume Filtering

The framework of [11] performs a fast, interactive object segmentation and allows users to extract objects from a video. Therefore, the user has only to annotate a few foreground scribbles in order to mark pixels which belong to the foreground. Afterwards, a fast optimisation is performed, which is based on spatio-temporal cost volume filtering. At the end, the filtered cost volume is thresholded to get a binary segmentation, which extracts the foreground object from the background.

For simplicity, users only have to mark the foreground object. Thus, a few scribbles are necessary for object segmentation and the amount of user effort stays low. After scribbling on one frame, colour models are built, which are used to compute costs. As colour models colour histograms are used, which sum up to one. Each histogram has K bins and in total two

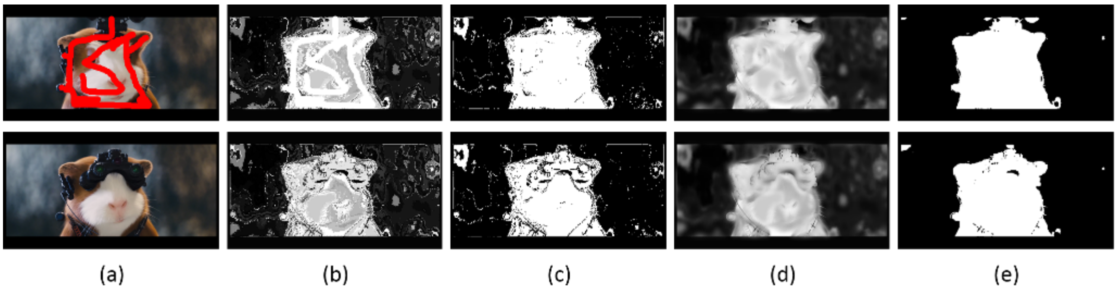


Figure 3.9: Video Object Segmentation of a video with eight frames with the algorithm of [11]. Results in the first row present the first frame of the video, and in the second row results of the last frame are shown. The costs in the cost volume representations ((b) and (d)) are shown in grey values, where black are high costs (1) and white are low costs (zero). (a) shows the input first frame with user-assigned scribbles and the last frame, (b) the unfiltered computed cost volume, (c) the result after thresholding the unfiltered cost volume from (b), (d) the guided video filtered cost volume and (e) the resulting segmentation after thresholding the filtered cost volume from (d). The segmentation is performed without any additional region filling or alpha matting step.

histograms are computed. H_f is the foreground histogram and is built from the user-marked pixels. The background histogram H_b is built from randomly chosen background samples (cf. Figure 3.8). Hence, it is possible that not all background samples truly belong to the background. In [11] the colour models are improved step-by-step, while users interact with the scribble-based user interface (UI). Since this UI, which supports local and progressive editing, is not relevant for the further extension to a depth propagation (cf. Chapter 4), it is not further discussed.

After computing the foreground colour model H_f and the background colour model H_b, the assignment of each pixel to either the foreground F or the background B has to be done (cf. Figure 3.8). Thus, to achieve a spatio-temporal result, cost volume filtering is applied. First of all, a cost volume is built (cf. Figure 3.9(b)), then the cost volume is filtered (cf. Figure 3.9(d)) and one out of the two labels is assigned to each pixel in the video (cf. Figure 3.9(e)). As discussed in Chapter 3.3, a cost volume is built by computing the costs $c_{col}(i)$ of each pixel $i = (x, y, t)$ per label l out of all labels $\mathfrak{L} = (1, ..., L)$. In the context of the binary video object segmentation there are only two labels $\mathfrak{L} = (F, B)$, the foreground F and the background B. Thus, only one cost volume slice is necessary. The cost volume C is three-dimensional (x, y, t), which determines the costs of a pixel (x, y) in frame t to belong to the foreground F. The costs $c_{col}(i)$ are based on the comparison of the frequencies of i's bin in the histograms H_f and H_b:

$$c_{col}(i) = 1 - \frac{H_f(i)}{(H_f(i) + H_b(i))}. \tag{3.10}$$

The costs of all user-marked pixels, which belong to the foreground, are set to 0 (cf. Figure 3.9(b) first row). To achieve a spatio-temporal result, the guided video filter [39], which was discussed in Section 3.3, is applied on the computed cost volume. In that way, pixels are spatially and temporally smoothed while edges are preserved, which results in a spatio-temporally coherent cost volume. A comparison of the cost volume without and with the filtering step is shown in Figure 3.9(b) and (d), respectively. After the filtering step, a threshold is used to assign each pixel either to the foreground (< 0.5) or the background (≥ 0.5) (cf. Figure 3.9(e)). Finally, in [11] an additional, temporally coherent matting step is optionally performed to obtain a soft segmentation for mixed pixels at objects borders.

3.4 Summary

This chapter summarised different aspects of video segmentation and cost volume filtering in the context of their relevance in the field of 2D-to-3D conversion. First, a general overview of segmentation was given including respresentation forms and features of segmentation. Then, several segmentation techniques were presented. On the one hand, these include spatial methods, which work on images, and on the other hand there are spatio-temporal methods, which are applied to videos and result in segments that extend over time. In the case of spatio-temporal methods, the segmentation algorithm can either be based on spatial approaches to detect segments in individual frames and further track them from frame to frame, or can analyse the whole video at once to perform a segmentation. Finally, interactive segmentation described as a labelling problem was discussed. The main idea of cost volume filtering was presented as well. Since the algorithm proposed in this thesis is based on a video object segmentation algorithm that exploits the concept of cost volume filtering, such a algorithm was discussed in more detail in the final section of this chapter.

CHAPTER 4

Cost Volume Filtering-based Depth Propagation

This chapter presents a new semi-automatic 2D-to-3D conversion approach. The proposed approach is motivated by the interactive video object segmentation algorithm from [11]. The interactive segmentation from [11] performs a binary segmentation of a video into foreground objects and background objects (cf. Section 3.3) based on user-performed foreground scribbles. By contrast, our proposed approach focuses on a different and more complex application domain, i.e., 2D-to-3D conversion. Therefore, specific challenges concerning a comfortable 3D viewing experience have to be taken into account additionally to the increased label set in 2D-to-3D conversion, i.e., multiple disparities contrary to only fore- and background. Thus, in Section 4.1 we describe a *naive extension* of [11] to a 2D-to-3D conversion method. Contrary to [11], our extension performs a multi-label segmentation into depth-layers and propagates a user-assigned depth value to each pixel in an image or video. For this purpose, user-assigned depth scribbles are used. In our case of 2D-to-3D conversion, a depth-layer is a part in the video defined by its position in depth. Thus, an object can consist of multiple depth-layers with different assigned depth values, which increases the complexity of the 2D-to-3D conversion compared to the segmentation. The goal of further extensions is to achieve a temporally coherent depth map with additionally identified depth-layers.

The naive extension of [11] to a simple depth propagation framework mentioned above (cf. Section 4.1), is further improved in several steps. The *temporal consistency improvement* (cf. Section 4.2) implements an enhanced filter, the *spatial influence extension* (cf. Section 4.3) adds an additional spatial influence term, and the *temporal depth change extension* (cf. Section 4.4) enables temporal depth changes. Those extensions and improvements are presented in the following.

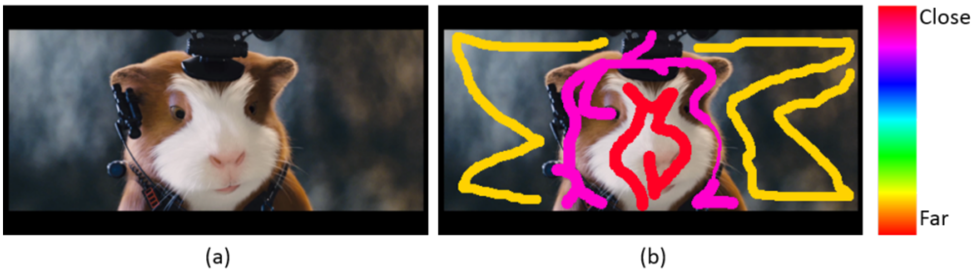


Figure 4.1: (a) the first frame of the video, (b) the first frame annotated with scribbles by the user. The depth values are represented in the form of scribbles that are colour coded according to the hue scale.

4.1 Naive Extension to a Simple Depth Propagation Framework

We first extend the video object segmentation of [11] to a *naive extension* (NE), that is a simple depth propagation method that is based on multiple scribble-based depth annotations instead of the originally used foreground scribbles in [11]. Below, we briefly explain the annotations used and the propagation using the cost-volume filtering framework [40].

User-assigned depth scribbles Prior to the execution of the propagation algorithm, users annotate objects in the first input frame with colour scribbles. The colour of the scribbles encodes their depth, whereby each scribble l has exactly one depth value assigned. Specifically, the hue of the scribble l defines the assigned depth value $d(l)$ (cf. Figure 4.1). Given these sparse depth values, the propagation algorithm's goal is to assign each pixel (x, y) in each frame t to one of the user defined scribbles l from the set of all scribbles $\mathfrak{L} = (1, ..., L)$ and their associated depth values. Analogue to [11], we generate a four-dimensional cost volume $C(x, y, t, l)$, where the costs $c(i, l)$ of each pixel $i = (x, y, t)$ for assigning the depth of scribble l are computed.

Colour models Our main assumption is that similar colours result in similar depth in the final depth propagation. It has been shown (e.g., [8], [11]) that such an assumption can be implemented using colour models. Similar to [11], binned colour histograms that were generated from the user-assigned scribbles are used to perform the cost computation. Contrary to [11] (cf. Section 3.3), we compute fore- and background colour histograms for each scribble l separately. Figure 4.2 gives an example of the cost computation that is performed in the case of 2D-to-3D conversion. Trimaps shown in Figure 4.2 are grey scale images used for the generation of the histograms of each scribble, where white pixels are pixels from the current scribble l, black pixels are pixels from all other scribbles except the current scribble and grey pixels are not covered by any scribble. The foreground histograms $H_{f,l}$ are built from all pixels that are covered by the current scribble l (white pixels in the trimap in Figure 4.2). The background colour histogram $H_{b,l}$ is built from all user-marked pixels without the pixels of the current scribble l (black pixels in the trimap

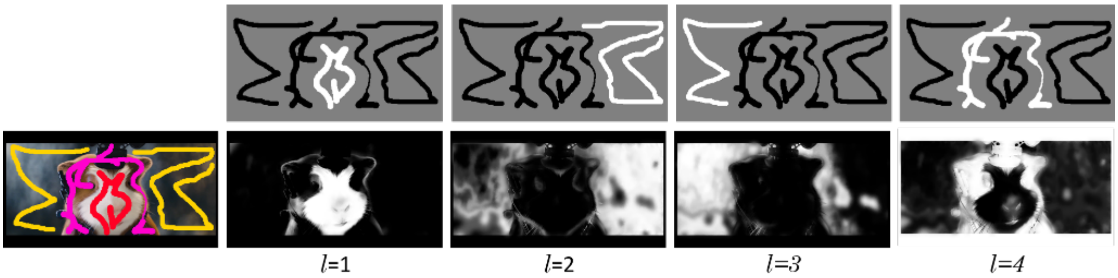


Figure 4.2: In the first row, trimaps of each user defined scribble from Figure 4.1 are shown. White pixels in the trimaps mark the current scribble l and are used for the generation of the foreground histograms $H_{f,l}$. Black pixels mark all other scribbles $\mathfrak{L} \setminus l$ and are used for the generation of the background histograms $H_{b,l}$. Grey pixels are not covered by any user scribble and have no influence on the computation of the histograms. The resulting guided filtered cost volume slices for each scribble l are shown in the second row for the first frame.

in Figure 4.2). In [11] the background histogram was built according to randomly chosen pixels from the background.

Cost volume generation A cost volume slice $C(l)$ is computed for each scribble l. For each pixel $i = (x, y, t)$ the costs for each label l are computed according:

$$c_{col}(i, l) = 1 - \frac{H_{f,l}(i)}{(H_{f,l}(i) + H_{b,l}(i))}. \tag{4.1}$$

Here, $c_{col}(i, l)$ describes the probability that pixel i belongs to scribble l. The last part of Eq. (4.1) is the actual cost computation. Low costs refer to a high probability that pixel i belongs to scribble l and vice versa. In the following, c_{col} is referred to as cost computation and both terms, small costs (i.e., values close to 0) and high probability (i.e., values close to 1) are used synonymously. Subsequently, the l slices, i.e., costs for a specific label l, of the cost volume are filtered with the guided video filter (cf. Section 3.3). Following the CVF framework [40], the costs of all user-marked pixels are set to either 0 or 1, depending on whether the pixel belongs to the current scribble l or to one of the other scribbles. An example of a filtered cost volume is shown in Figure 4.2. It can be seen that those pixels which are similar in colour to pixels covered by the user-assigned scribble l have low costs/high probabilities (white pixels in cost volume in Figure 4.2).

Depth assignment We experimented with two approaches for the final depth propagation $d(i)$ of each pixel $i = (x, y, t)$: (i) the winner-take-all approach (WTA) (cf. Section 3.3) and (ii) depth blending (DB). In the winner-take-all approach, the final propagated depth value of scribble f and therefore the depth value $d(f)$ is chosen from the scribble l and their corresponding depth value $d(l)$ with the lowest cost at pixel $i = (x, y, t)$ (cf. Eq. (3.9)). In that case, the resulting depth map contains only depth values which the user originally assigned with the colour coded scribbles at the beginning (cf. Figure 4.3(b)). Since no variations inside an object are given, this leads to the cardboard effect.

Our newly introduced depth blending approach avoids the above mentioned cardboard effect and enables other depth values as well. The basic idea of this approach is to compute a mean of disparities weighted by their costs. In [89] the authors used this approach in the context of a colourization application. They compute a weighted average of various colours in the provided set of scribbles for the finally assigned colour value. Similarly, the finally assigned depth value at pixel i is an average of all different depth values $d(l)$ which the user assigned at the beginning, weighted by the computed costs at each scribble l:

$$d(i) = \frac{\Sigma_{l \in \mathfrak{L}} c_{i,l} d(l)}{\Sigma_{l \in \mathfrak{L}} c_{i,l}} \tag{4.2}$$

This depth blending approach eliminates the cardboard effect, since there are now depth variations within objects. A further extension is to compute a weighted mean value of the n lowest cost volume slices (out of all cost volumes slices for each scribble l) at each pixel (cf. Figure 4.3). By this means, labels with the highest costs which may indicate outliers do not have any influence on the depth propagation.

Experimental Results

Figure 4.3 gives an example for both depth assignment strategies. Results of the first approach are presented in Figure 4.3(b). It is shown that all three user-assigned depth values are propagated to the final depth map. There are sharp depth changes near object borders. Moreover, the depth borders are well aligned with object borders. However, noise occurs, e.g., at the top left borders of the head. The most noticeable errors are the small holes within the object, e.g., at the bottom (cf. red arrows in Figure 4.3). These holes occur because the colour similarity to one of the background (yellow scribbles in Figure 4.3) scribbles is higher than to the foreground object. Thus, the user-assigned depth of the background scribbles is assigned to those pixels. These holes within the object could easily be corrected with a simple region filling algorithm, e.g., [27]. As mentioned earlier, a disadvantage of the winner-take-all approach is the limitation to the user assigned depth values. Since only the initial depth values are propagated, no depth variations within an object can be modelled, which causes a cardboard effect. After using a rendering algorithm, e.g., [25], to generate the second image and viewing the video in 3D, the objects in the video appear to be flat. The results of depth blending are presented in Figure 4.3(c) and (d). While in Figure 4.3(c) only the $n = 2$ lowest cost volume slices are used to compute depth blending values for the propagation, Figure 4.3(d) shows depth blending of all $n = 4$ cost volume slices. As can be seen, the results in Figure 4.3(d) are smoother than in Figure 4.3(c), which can be attributed to the inclusion of cost volume slices with the largest costs. The difference is especially noticeable in the background, where various depth values are propagated. The depth propagation with this approach enables depth variations within the object. However, the holes that are caused by colour ambiguities between objects and which were noticeable when using the WTA approach for depth propagation in Figure 4.3(b), are also occurring in the results when using the depth blending strategy.

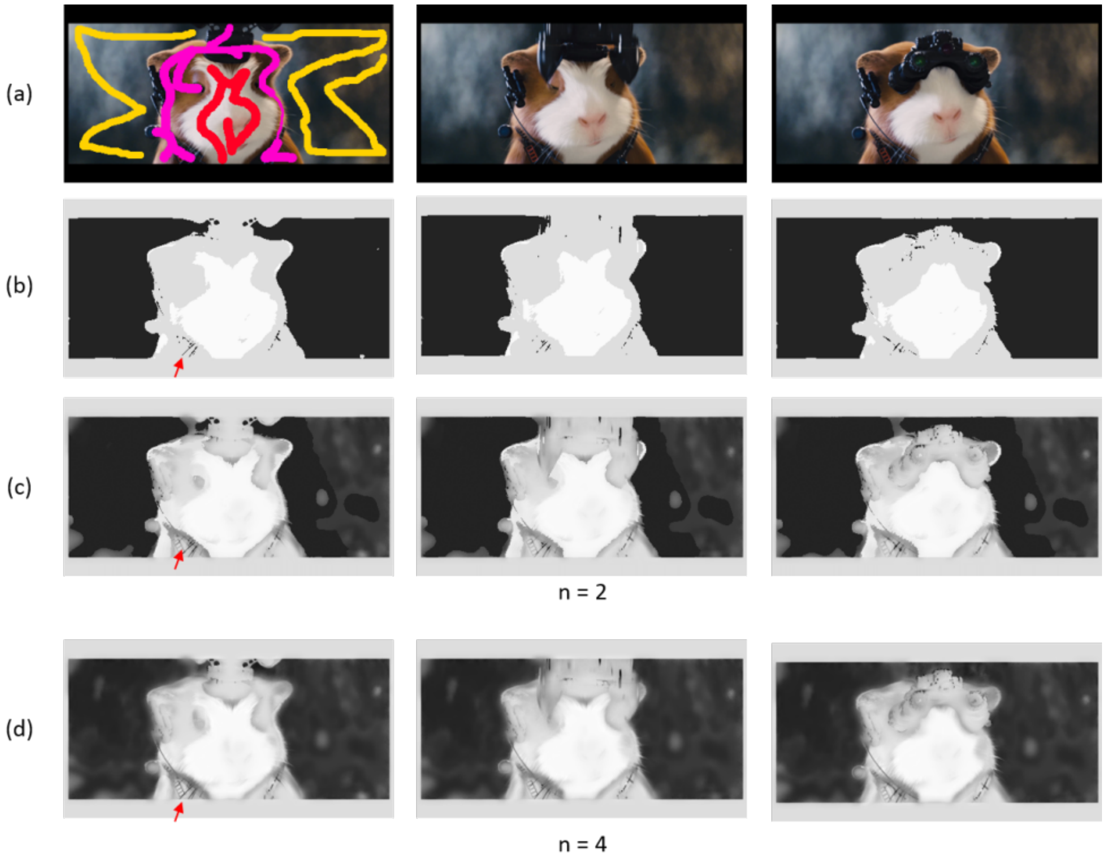


Figure 4.3: Resulting depth propagation. (a) shows frame 1, 5 and 8 of the input video, (b) the depth propagation due to the winner-take-all approach, (c) and (d) show the final depth propagation due to the depth blending approach. (c) shows the depth propagation of $n = 2$ lowest cost volume slices, while in (d) all $n = 4$ cost volume slices are used for the depth blending propagation. Red arrows point to holes within the object caused by colour similarity.

4.2 Temporal Consistency Improvement

In the naive extension of the interactive segmentation algorithm to a 2D-to-3D conversion algorithm, the temporal consistency of the conversion result is limited by the filter radius. Specifically, fast motion can cause an object to move out of the static spatio-temporal filter window of the guided video filter. Since the additional information provided by the neighbouring frames includes information of various objects with different depth values, temporal filtering mixes up depth values of various objects and noise occurs at depth borders (cf. Figure 4.4). The described problem can, depending on the size of the filter window, easily occur in videos that contain dynamic objects.

We address this problem by extending the temporal filtering and improve the temporal consistency of our 2D-to-3D conversion results. Therefore, the *temporal consistency improvement*

(TCI) extends the guided video filter (cf. Section 3.3) used in the CVF process. To this end, the motion-based box filter (cf. Section 3.3), which filters along motion paths, is implemented and used within the guided filter computation instead of the common spatio-temporal box filter as done in [40]. Thus, an optical flow implementation and further the generation of long motion paths have to be performed.

For this purpose, an optical flow estimation provided by OpenCV [62] is used (unless explicitly stated otherwise in this thesis). The estimated motion vectors are used to build motion paths throughout the video. Specifically, motion paths are built by following the estimated flow vectors $w = (u, v)$ of each pixel $i = (x, y, t)$ to the new position in the next frame $i_{t+1} = ((x_{t+1} = x_t + u_t, y_{t+1} = y_t + v_t), t + 1)$. As the flow vectors are floating point numbers, $i_{t+1} = (x_{t+1}, y_{t+1}, t + 1)$ usually ends up between pixels. Thus, after a rounding operation the new pixel position is determined. In Section 3.3, a few constraints for building these motion paths were already discussed: (i) a path has to end, when it leaves the scene, (ii) when multiple paths converge on the same pixel, only one path continues randomly and (iii) when a pixel does not belong to a path in the previous frame, a new path starts at this pixel's position. Now an additional constraint is introduced in order to further improve the paths. Thus, (iv) a path should end, when it gets occluded in order to avoid paths that contain pixels of different objects. We detect occlusions by a tolerant consistency check of the forward and backward optical flow [78]. Hence, flow vectors are computed once from frame t to the next frame $t + 1$, which describes the forward flow vectors $w = (u, v)$, and a second time from frame $t + 1$ back to frame t, which denotes the backward flow vectors $\bar{w} = (\bar{u}, \bar{v})$. Using this notation, the flow vectors in the non-occlusion case equal

$$u_t(x_t, y_t) = -\bar{u}_t(x_t + u_t, y_t + v_t) \tag{4.3}$$

and

$$v_t(x_t, y_t) = -\bar{v}_t(x_t + u_t, y_t + v_t). \tag{4.4}$$

If Eq. (4.3) and Eq. (4.4) are not fulfilled, the vector does not point back and it is either occluded in frame $t + 1$ or an estimation error occurred. Either way, the path should stop at such a point. However, small estimation errors are tolerated by the following constraint:

$$|w - \bar{w}| < 0.5 \tag{4.5}$$

If this condition is not satisfied, the path breaks up and a new one starts at frame $t + 1$. With this additional condition, the long motion paths are built and further used in the motion-based box filter, which is used in the guided video filter implementation from Section 3.3. Thus, the guided video filter uses the long motion paths to filter in the temporal direction.

Experimental Results

Figure 4.4 gives an example of the temporal consistency improvement which was discussed in this section. As described above, the flow vectors are further used to generate the long motion paths. Figure 4.4(a) shows the results of the naive extension of [11] discussed in the previous Section 4.1, and Figure 4.4(b) presents results after applying the above discussed temporal

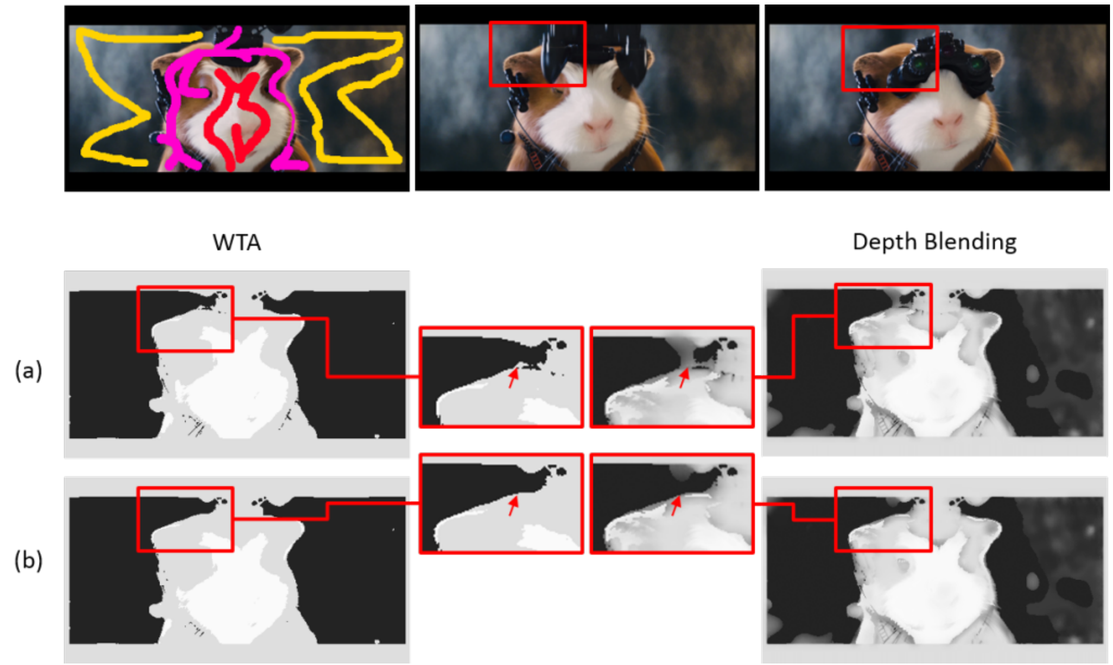


Figure 4.4: Comparison of the naive extension of [11] of Section 4.1 shown in (a) and the results of the temporal consistency improvement shown in (b). To the left the resulting depth propagation using the WTA approach is shown, while to the right the resulting depth propagation using the depth blending is shown. As can be seen, the noise especially at depth borders is being reduced.

consistency improvement. As can be seen in Figure 4.4(b), the before mentioned noise at the object's boundary is corrected in both approaches, the winner-take-all approach (cf. Figure 4.4(a) left hand side) and the depth blending approach (cf. Figure 4.4(b) right hand side). Hence, an improvement, especially at depth boundaries, is achieved, because the enhanced filter uses the long motion paths to filter in the temporal direction. The filter window moves along the built long motion paths and thus, together with the object. Hence, the filter window is centred on the same point of the same object in the previous and next frames. With the common guided video filter implementation (cf. Section 3.3) it might happen that pixels with similar colours in the previous or next frame belong to a different object and thus the filtering process results in artefacts (cf. Figure 4.4(a)).

4.3 Spatial Influence Extension

We believe that users typically prefer local changes. In other words, they assume that a scribble on an object influences only the scribbled object and not similarly coloured objects that are far away from the scribble. Thus, the assumption is that pixels close to a scribble have lower costs (i.e., high probability to belong to the depth of the scribbled object) than pixels that are far away

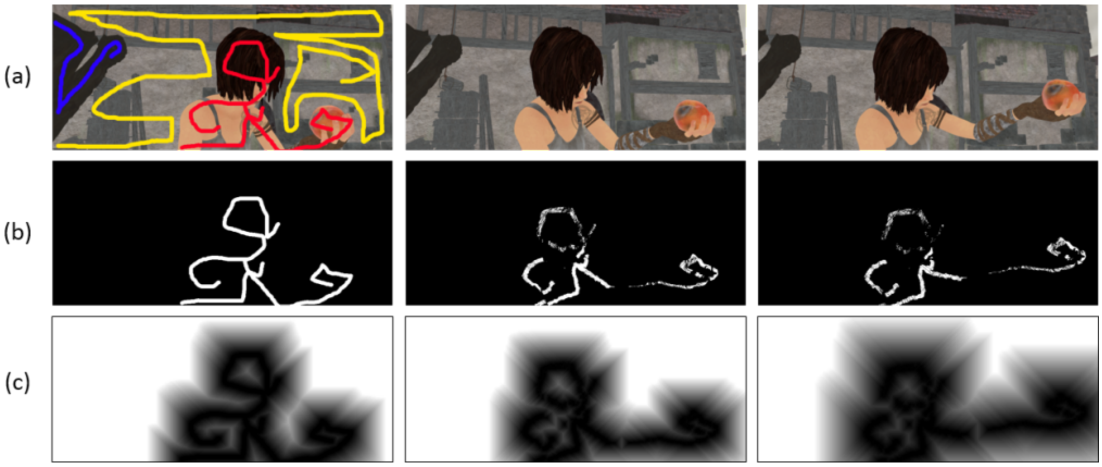


Figure 4.5: (a) shows frame 1, 13 and 20 (from left to right) of the input video. (b) presents the movement of points belonging to the red coloured scribble in the first frame by showing their long motion path pixels in frame 1, 13 and 20. (c) visualizes the distance maps with a spatial threshold $t_{spatial} = 100$. Black pixels are the tracked scribble points of scribble l, while white pixels result in total costs of 0. Grey values in between are reducing linearly the colour probability $c_{col}(i, l)$ and result in higher total costs.

from the current scribble. This assumption was not included in the naive extension of [11]. Since costs are calculated due to colour similarity, *wrong pixels* with low costs can appear. Such wrong pixels are occurring when pixels have a similar colour as the scribbled object, while actually belonging to another object. This case is shown in Figure 4.7, where wrongly computed costs cause a mis-assignment of depth in the final depth propagation (cf. Figure 4.7(b) and (c) first row).

Therefore, the idea of this *spatial influence extension* (SIE) is to update the calculation of the total costs $c(i, l)$ for each pixel i belonging to scribble l, by adding an additional spatial influence term $c_{sp}(i, l)$. This spatial influence term $c_{sp}(i, l)$ describes the influence of pixel i on the cost computation based on its spatial distance to scribble l. The basic idea is that the spatial influence term $c_{sp}(i, l)$ increases the total costs of wrong pixels, which are far from the current scribble l. Below, we describe the computation of these spatial costs and the accordingly performed update of the cost volume in detail.

The computation of spatial costs in the first frame is straightforward. The distance of each pixel i to the user-assigned scribble l in the first frame is determined in order to calculate the spatial influence. To this end, a distance map for each scribble l is computed, which establishes the distance to the closest pixel from scribble l. Therefore, a distance transform method using the algorithm from [7] is implemented. The result of the distance transformation method for each scribble l is an image where all non-scribble pixels have a distance value, which is the distance to the nearest scribble pixel of scribble l [7]. Commonly, metrics like the Euclidean distance or the Manhattan distance are used for the distance calculation. However, these approaches are complex and therefore in [7] the author introduced a faster and local approach, where the function consists of basic shifts in horizontal, vertical and diagonal direction and the distance is calculated as a sum of these basic distances. The resulting distance map of scribble l defines the distance, given in pixels, to the closest scribble point of l.

In a next step, a spatial threshold $t_{spatial}$ is defined, which determines the maximal distance to scribble points, where costs are weighted due to their distance to the scribble points. This threshold $t_{spatial}$ is set in pixels. In the following $dst(i, l)$ is the distance of pixel $i = (x, y, t)$ to scribble l in frame t in pixels. $c_{sp}(i, l)$ are the spatial costs at pixel i regarding scribble l:

$$c_{sp}(i,l) = \begin{cases} 1.0, & \text{if } dst(i,l) \geq t_{spatial} \\ dst(i,l)/t_{spatial}, & \text{otherwise} \end{cases} \tag{4.6}$$

As a result, scribble points from scribble l have a distance value 0.0 (black pixels in Figure 4.5(c)) and pixels with a distance $\geq t_{spatial}$ are set to 1.0 (white pixels in Figure 4.5(c)). Thus, the resulting spatial costs $c_{sp}(i, l)$ are in a range between 0.0 and 1.0.

To calculate spatial costs $c_{sp}(i, l)$ in the following frames, the position of the user-assigned scribbles in the first frame have to be identified in each of the following frames t. To this end, we use the long motion paths that were introduced in Section 4.2 to track scribble points in each frame. Paths which start at scribble points of l in the first frame are followed throughout the video. Thus, in each frame t the position of the scribble points of l can be determined and a distance map can be computed.

Finally, the computed spatial costs $c_{sp}(i, l)$ are combined with the costs based on colour similarity $c_{col}(i, l)$ in order to determine total costs $c(i, l)$, i.e., the probability of pixel i belonging to scribble l:

$$c(i,l) = c_{col}(i,l) * (1 - c_{sp}(i,l)) \tag{4.7}$$

The right term of this equation (Eq. (4.7)), determines the probability that a pixel i is close to a point of scribble l. By combining this probability with the colour probability (cf. Eq. (3.10)), the cost of a pixel close to scribble l hardly changes, while the costs of pixels further away increase until the threshold $t_{spatial}$ is reached, i.e., the probability belonging to scribble l decreases. Pixels further than $t_{spatial}$ have costs of 1.0 and are excluded of belonging to scribble l. By this means, wrong pixels with similar colour are eliminated.

Since users typically prefer local editing over global editing, we additionally allow the definition of the spatial costs influence specific for each scribble. This means we allow expert users to define separate $t_{spatial}(l)$ for each scribble l. To allow an easier choice of the area of increased influence for a specific scribble l, the respective distance map (e.g., as in Figure 4.5) can be visualised.

Experimental Results

Figures 4.6 to 4.11 show the effect of the spatial influence extension. In these examples, ground truth optical flow [16] was used. In the following, different aspects of the spatial influence extension are discussed: (i) the general impact of the additional spatial costs term, (ii) the influence of the spatial threshold $t_{spatial}$, (iii) the additional feature of separated spatial thresholds, and (iv) the combination with the temporal consistency improvement from Section 4.2.

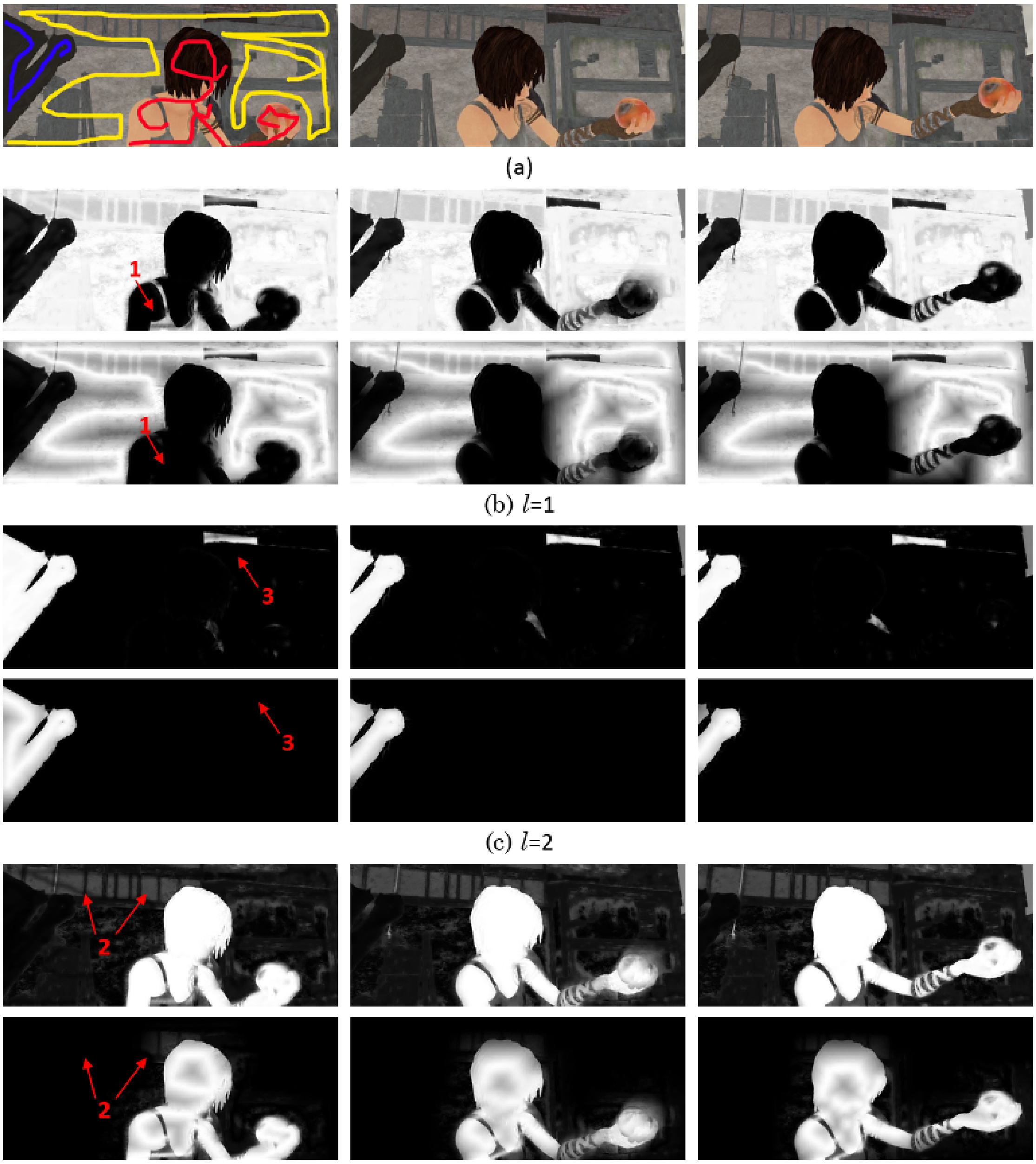
Impact of the additional spatial costs term The influence of the additional cost term can be seen in the cost volumes shown in Figure 4.6 and the corresponding resulting depth maps shown in Figure 4.7. In the following discussion, the resulting cost volume with simple calculated costs as presented in the naive extension of [11] (cf. Section 4.1) is called the *basic cost volume*, while the *spatial cost volume* refers to costs calculated with the additional spatial influence term as introduced in this spatial influence extension. For better understanding, details discussed in the following passages are highlighted and numbered in Figure 4.6 and Figure 4.7.

The basic cost volume slice of scribble $l = 1$ (the yellow scribble shown in Figure 4.6(a) with the corresponding basic cost volume slice in (a) first row), which marks the background region, clearly shows wrong cost values in the region of the person's tank top, and thus, propagates a wrong depth value to the final depth map (cf. arrow nr. 1 in Figure 4.6 and Figure 4.7). These wrong costs are occurring due to the colour similarity of the tank top and the marked background. In the corresponding spatial cost volume slice of the scribble $l = 1$ (cf. Figure 4.6(b) second row), those costs are eliminated because those pixels are too far from scribble $l = 1$.

The contrary case can be seen in the basic cost volume slice of scribble $l = 3$ (the red scribble shown in Figure 4.6(a) with the corresponding basic cost volume slice shown in (d) first row), which marks the person in the foreground. Here, the structure in the background shows low costs (cf. arrow nr. 2 in Figure 4.6 and Figure 4.7), because of the colour similarity to the marked tank top. These wrong costs have no effect on the final depth propagation of the WTA approach as can be seen in the final depth propagation shown in Figure 4.7(b) in the first row. Due to the fact that the background region has already low costs in the cost volume slice of scribble $l = 1$, the depth value of scribble $l = 1$ is assigned. However, due to the weighted mean computation of all cost volume slices, these wrong costs effect the final depth propagation of the DB approach as shown in Figure 4.7. However, the wrongly computed low costs in the top right part of the frame (cf. arrow nr. 3 in Figure 4.6 and Figure 4.7) in the basic cost volume slice of scribble $l = 2$ (the blue scribble shown in Figure 4.6(a) with the corresponding basic cost volumes shown in (c) first row), are resulting in erroneous depth propagation (cf. Figure 4.7(a)).

Influence of the threshold $t_{spatial}$ By looking at the person's head in the second row of Figure 4.7(b), one recognizes that the depth borders of the depth propagation using the spatial cost volume show more artefacts than in the first row. This is the result of a too small spatial threshold, e.g., visible in the middle frame of Figure 4.6(b).

In Figure 4.9 the effect of various spatial threshold values $t_{spatial}$ is demonstrated. A higher threshold value leads to less noise in the head's border region, but also induces erroneous depth propagation in the tank top region. The result of a depth propagation with a large depth range and more detailed scribble input is presented in Figure 4.8. There, the effect of the spatial influence term is visible at the boxes at the left hand side of the scene



(a)

(b) l=1

(c) l=2

(d) l=3

Figure 4.6: Visualization of the effect of the spatial influence extension with the additional spatial costs $c_{sp}(i, l)$. (a) shows the user-assigned scribbles and frames 1, 13 and 20. (b)-(d) show the cost volume slice entries for each of the input scribbles. The first row per scribble l presents the result from the cost computation without spatial costs as in the naive extension of [11] (cf. Section 4.1), while the second row shows the results from the cost volume slices of scribble l with spatial costs $c_{sp}(i, l)$. (b) corresponds to the cost volume slice entries of the yellow user-assigned scribble marking the background. (c) corresponds to the cost volume slice entries of the blue user-assigned scribble marking the object at the right hand side. (d) corresponds to the cost volume slice entries of the red user-assigned scribble marking the person in the foreground. Highlighted and numbered regions refer to details discussed in the *Experimental Results* section.

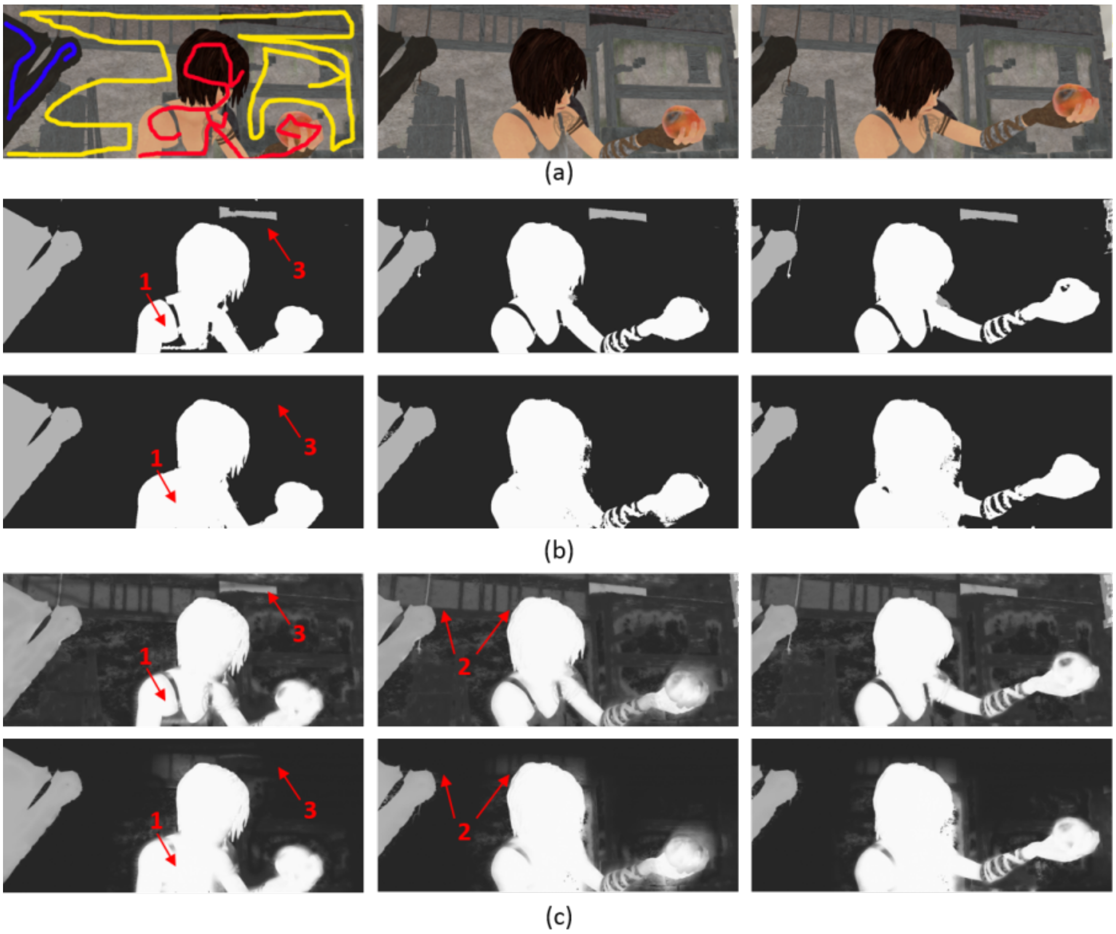
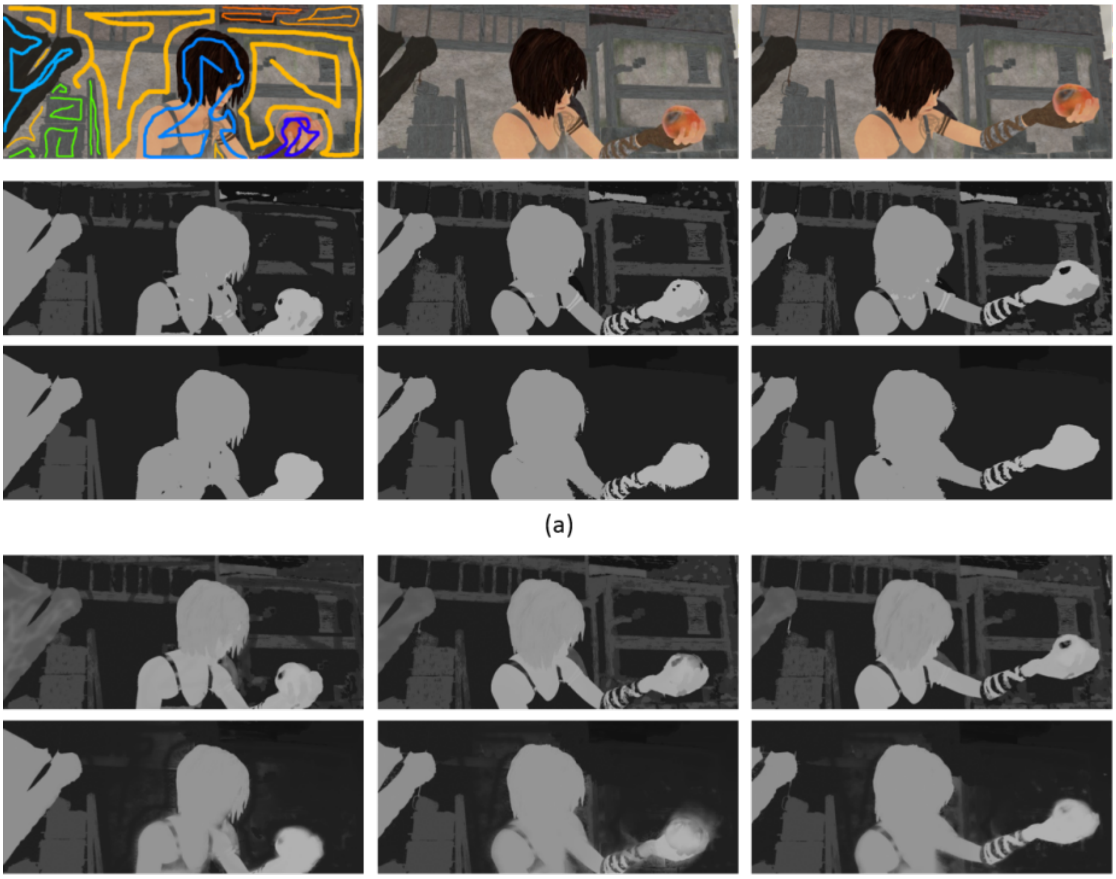


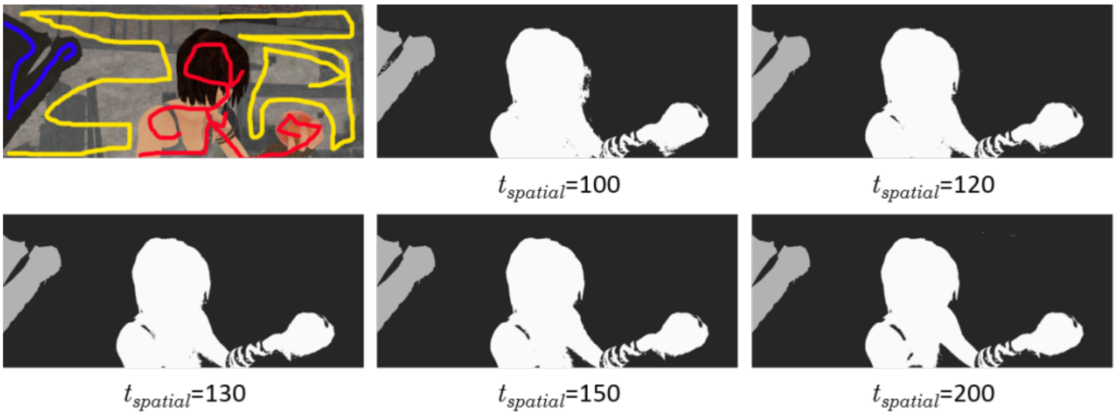
Figure 4.7: Final depth propagation. (a) shows frames 1, 13 and 20 of the input video, (b) shows results of the winner-take-all approach and (c) results of the depth blending approach. The first row in (b) and (c) shows the results of the depth propagation with a cost computation without spatial costs as in the naive extension of [11] (cf. Section 4.1). The second row in (b) and (c) shows the results of propagation with a cost computation with spatial costs. Highlighted and numbered regions refer to details discussed in the *Experimental Results* section.



(a)

(b)

Figure 4.8: Final depth propagation of a more advanced scribbling with 16 scribbles and 7 different depth values. (a) shows results of the winner-take-all approach and (b) results of the depth blending approach. The first row in (a) and (b) shows depth propagation with the common cost computation as presented in Section 4.1, while the second row in (a) and (b) shows propagation after adding the additional spatial influence term with $t_{spatial} = 130$ to the cost computation.



$t_{spatial}$=100

$t_{spatial}$=120

$t_{spatial}$=130

$t_{spatial}$=150

$t_{spatial}$=200

Figure 4.9: Final depth propagation after the winner-take-all approach with various spatial threshold $t_{spatial}$ values.

and the structure at the façade in the background. The similar colours of the boxes cause wrong costs at the façade and, therefore, the final depth propagation looks erroneous (cf. Figure 4.8(a) and (b) first rows). Using the spatial cost volume the propagation result improves (cf. Figure 4.8(a) and (b) second rows).

Separate spatial thresholds It is difficult to use a spatial threshold $t_{spatial}$ globally for all scribbles. For example, in the case of the background in Figure 4.10, a global threshold $t_{spatial} = 100$ results in an erroneous segmentation. Therefore a separate setting of $t_{spatial}(l)$ is introduced and illustrated in Figure 4.10.

Combination with the Temporal Consistency Improvement Finally, the results of both introduced improvements shall be discussed. On the one hand, there is the temporal consistency improvement from Section 4.2 and, on the other hand, the spatial influence extension presented in this section. In Figure 4.11, the combination of both is presented. The spatial influence extension can eliminate wrong pixels. However, it may still lead to noisy pixels around the object borders (cf. Figure 4.11 second row, red framed regions). By applying the temporal consistency improvement from Section 4.2, some noisy pixels can be removed, as can be seen in the last row of Figure 4.11 (green framed regions). Hence, making use of both introduced extensions can result in enhanced depth propagation.

4.4 Temporal Depth Change Extension

Dynamic objects in videos are typically not fixed at a single depth. For example, objects can move towards the camera or the camera zooms in. Our final extension enables temporal depth changes of objects. For this purpose, our 2D-to-3D conversion algorithm is extended by the *temporal depth change extension* (TDCE) by the following steps: (i) users mark the last frame with colour coded scribbles additionally to the first frame, (ii) matching scribble pairs from the

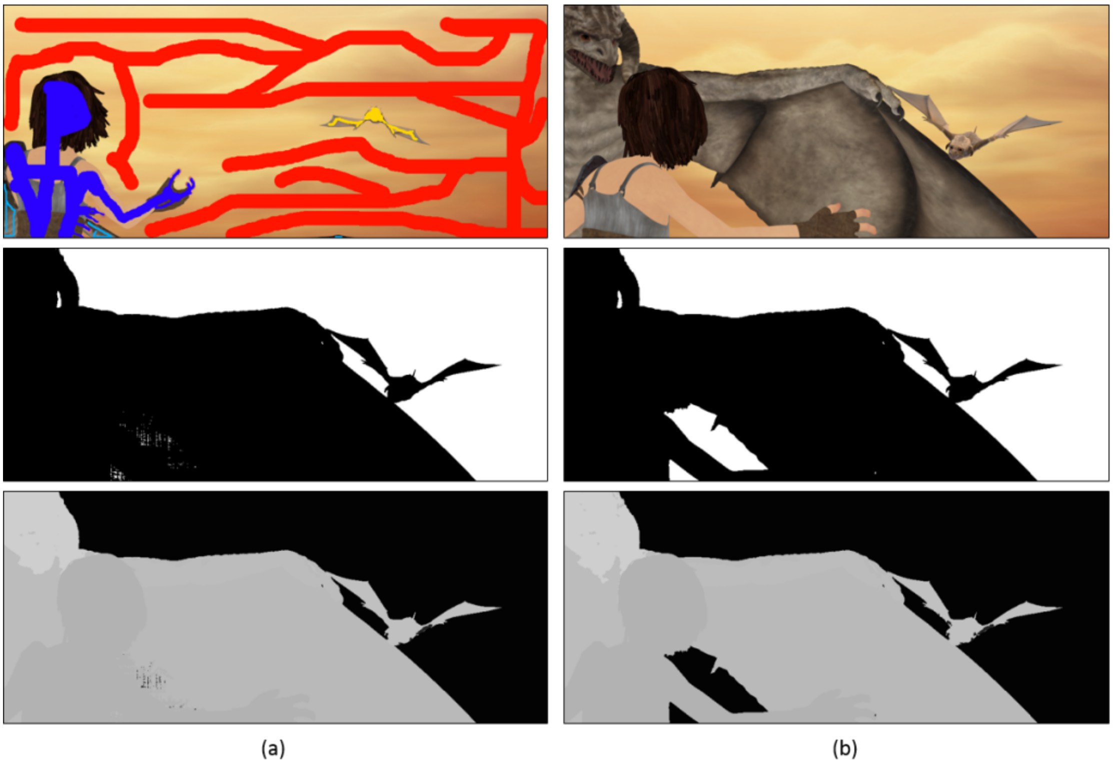


Figure 4.10: Comparison of the global threshold for all scribbles, shown in (a), versus a separate threshold for each scribble, shown in (b). In the first row of (a) and (b), the input scribbles on the first frame, and input frame 16 are shown, respectively. (a) shows in the second row the resulting CVF segment of frame 16 of the background (red scribble), by using a global threshold $t_{spatial} = 100$ for all scribbles in the video. In the third row, the resulting depth map (WTA approach) is shown. (b) shows results by using separate thresholds for each scribble. For the background (red scribble) no spatial influence term is added at all, the thresholds $t_{spatial}$ of the other objects in the video are chosen between 100 and 300 to achieve the final depth map (WTA approach) in the third row. In the second row of (b), the resulting CVF segment of the background (red scribble) without any spatial threshold for this scribble is shown.

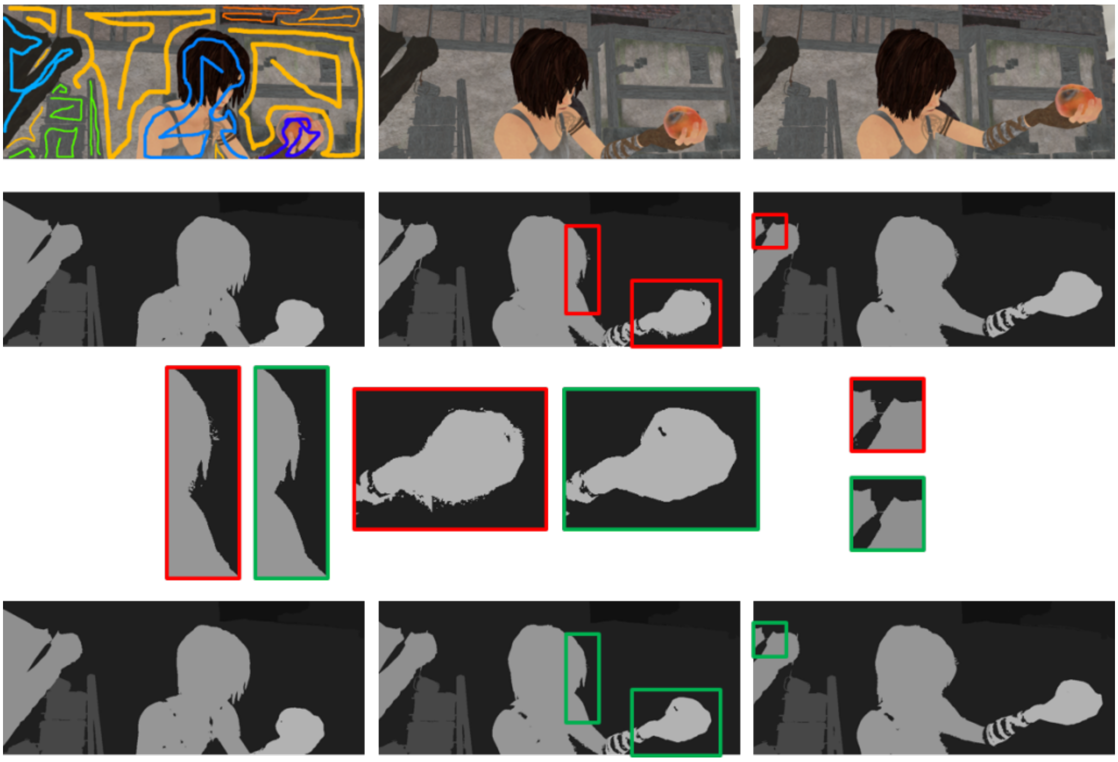


Figure 4.11: Effect of the temporal consistency improvement (cf. Section 4.2) in combination with the spatial influence extension. The first row shows frame 1, 13 and 20 of the input video. The second row presents results of CVF (winner-take-all) with the spatial influence extension (highlighted regions in red), while the last row shows CVF (winner-take-all) with the temporal consistency improvement and the spatial influence extension (highlighted regions in green). In the third, some regions are compared with each other. As can be seen, most of the noisy pixels at borders are removed by using the spatial influence extension.

first and last frame are identified, (iii) the cost computation is adapted, (iv) a depth order of the objects is determined to ensure a perceptual consistent depth change, and (v) based on the depth order, a depth change model for each scribble pair is determined. These steps are discussed in detail in the following.

(i) User Interface

In order to define the depth change of an object, the user has to add scribbles on the last frame as well (cf. Figure 4.12). A positive side effect of additionally adding scribbles in the last frame is that objects that enter the scene at a later point can be assigned with scribbles, e.g., the wing of the big dragon in the last frame in Figure 4.12.

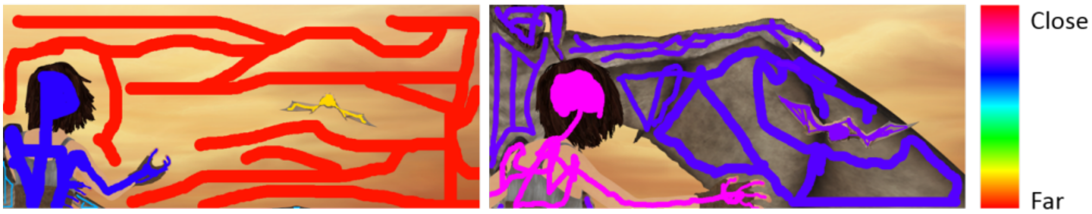


Figure 4.12: First (left) and last (right) frame with annotated scribbles of a video with 18 frames. By assigning differently coloured scribbles in the first and last frame, e.g., on the person or the small dragon, the user defines a depth change between the first and the last frame.

(ii) Identify matching scribble pairs

Subsequent to the user annotation, the propagation algorithm determines scribble pairs, i.e., one scribble in the first and one scribble in the last frame that cover the same object (e.g., the yellow and purple scribble of the small dragon in Figure 4.12). By assigning different depth values to an object in the first and last frame, users define a depth change over time. Based on these different depths, models for smooth depth changes can be defined at a later point of the algorithm. To identify the above mentioned matching scribble pairs, we adapt the motion segmentation algorithm of [30]. This algorithm merges long motion paths with similar motion to one segment. The basic idea is that matching scribbles of the same object will contain the same segment. These matching scribbles are merged to a single scribble pair.

In the following, we briefly review the segmentation algorithm of [30]. It works by merging similar (in terms of motion) neighbour paths to segments according to two adaptive thresholds (see [30] for details). Only a few adoptions are made to use the algorithm in the course of this thesis. The motion segmentation algorithm uses the built long motion paths for the temporal consistency improvement (cf. Section 4.2). To achieve a segmentation into segments of similar moving paths, the efficient graph-based segmentation algorithm from [26] is implemented (see [26] for more details). The long motion paths are grouped due to a similarity measure by taking the pixel positions of the paths and their movement, i.e., their flow vectors, into account[1]. Contrary to [30], we add an additional step: Initially, each node presents a region of its own. We further join all paths which belong to a user-assigned scribble l, independently from their edge weights. By this means, it is made sure that the user-assigned scribble belongs to exactly one segment of the video (cf. Figure 4.13). The following grouping of similar paths in segments is applied as in [30]. Finally, the graph-based segmentation of similar moving paths results in segments as can be seen in Figure 4.13.

The resulting segments (cf. Figure 4.13(b) and Figure 4.14(c) and (d)) are further used to identify matching scribble pairs. In this context, a scribble from the first or the last frame belongs to exactly one segment. However, it is worth to note that a segment can contain none, one or more than one scribble. The goal is to group scribbles so that each scribble belongs to one of the following definitions:

---

[1] In [30], an additional depth component is added, which is not used in this implementation.

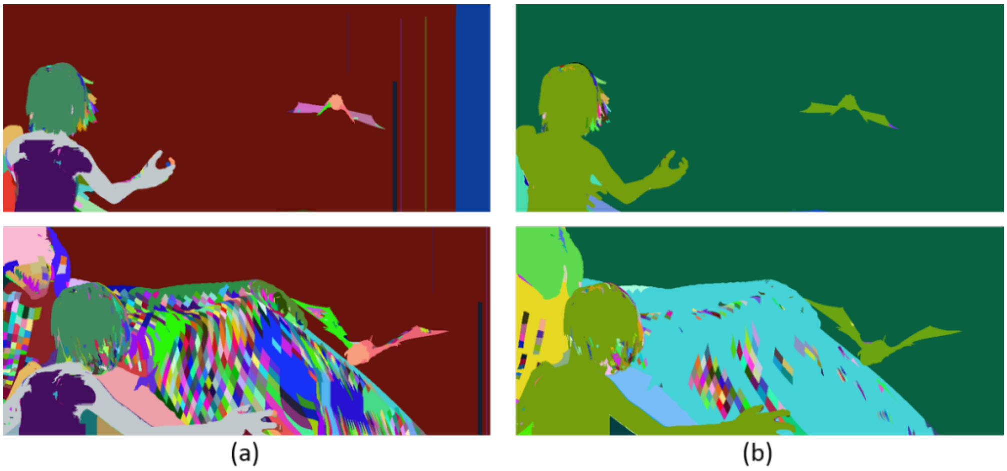


Figure 4.13: Illustration of resulting segments (a) without merging segments belonging to the user-assigned scribble l, and (b) with merging segments belonging to the user-assigned scribble l. User-assigned scribbles are shown in Figure 4.12.

Single start scribble One scribble from the first frame, e.g., yellow framed scribbles in Figure 4.14(f). This scribble belongs to exactly one segment and has a user-assigned depth value. No other scribble from the first frame with the same user-assigned depth value is in the same segment.

Grouped start scribble More than one scribble from the first frame. All start scribbles in this group belong to exactly one segment and have the same user-assigned depth value. Note that other scribbles from the first frame in the same segment but with a different user-assigned depth value might exist in the first frame. However, if they are assigned to a different depth value, they are not part of this grouped start scribble.

Single end scribble One scribble from the last frame, e.g., blue framed in Figure 4.14(f). This scribble belongs to exactly one segment and has a user-assigned depth value. No other scribble from the last frame with the same user-assigned depth value is in the same segment.

Grouped end scribble More than one scribble from the last frame. All end scribbles in this group belong to exactly one segment and have the same user-assigned depth value. Note that other scribbles from the last frame in the same segment but with a different user-assigned depth value might exist in the last frame. However, if they are assigned to a different depth value, they are not part of this grouped end scribble.

Scribble pair A pair of a (grouped) start scribble from the first and a (grouped) end scribble from the last frame, e.g., green framed in Figure 4.14(f). All scribbles in a scribble pair belong to exactly one segment, where the (grouped) start scribble(s) have a different user-assigned depth value than the (grouped) end scribble(s).

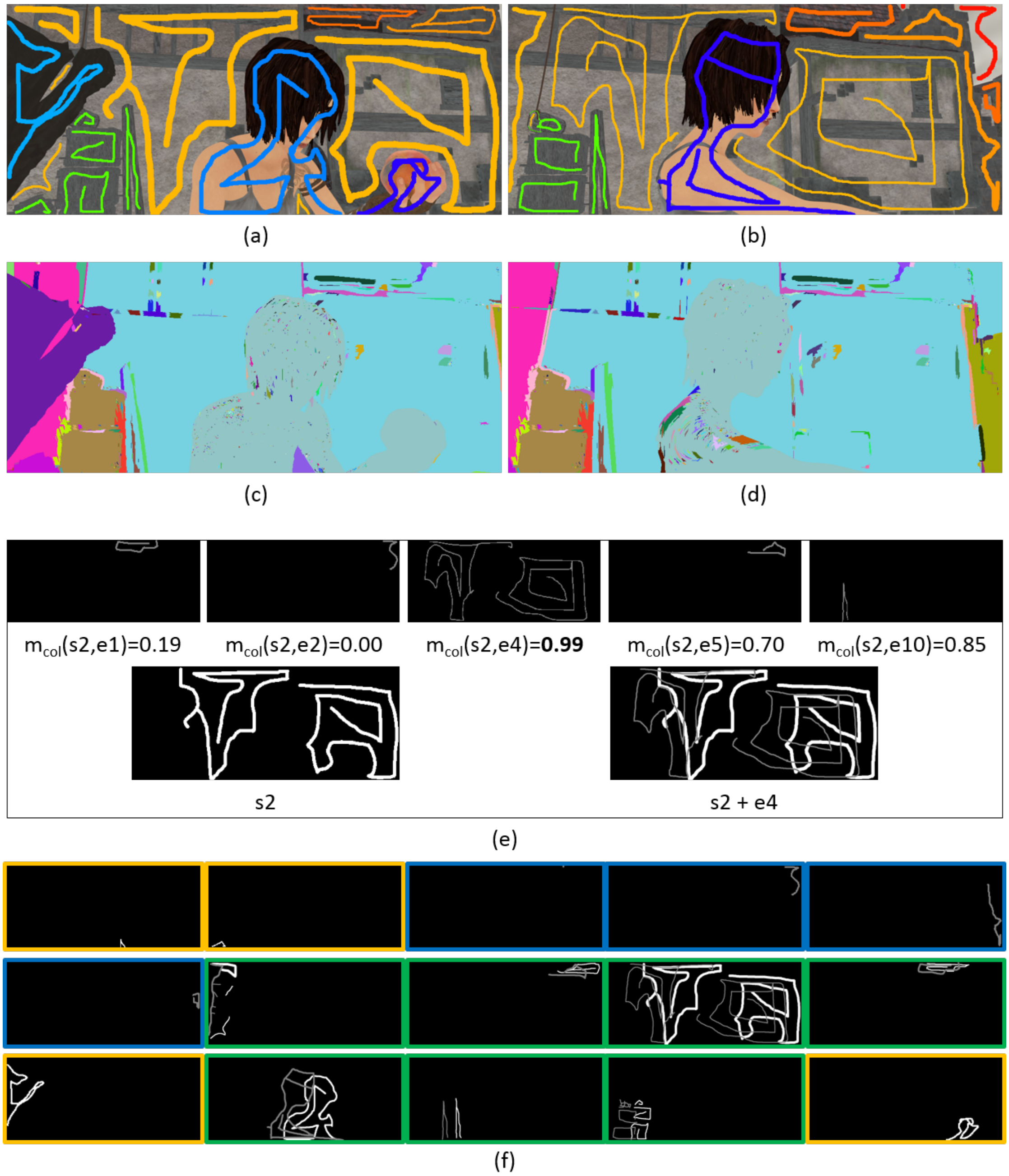


Figure 4.14: Identifying matching scribble pairs. (a) shows the first frame with user-assigned start scribbles and (b) shows the last frame with end scribbles. A depth change of the person is noticeable between the first and last frame scribbles. (c) shows the resulting segments in the first frame, while (d) shows segments in the last frame. (e) presents the scribble matching process. The grouped start scribble s_2 has five possible matches with (grouped) end scribbles e_1, e_2, e_4, e_5 and e_{10}, where the grouped end scribble e_4 achieves the highest correlation and is chosen for a scribble pair match. (f) shows all resulting single start scribbles (yellow framed), single end scribbles (blue framed) and joined scribble pairs (green framed)

.

Depth changes are indicated by corresponding (grouped) start and (grouped) end scribbles that exhibit different depths. To identify pairs (i.e., scribble pairs) of corresponding (grouped) start and (grouped) end scribbles we perform a matching step. Each of the (grouped) start scribbles is examined according to corresponding (grouped) end scribbles in the same segment. It is possible that there is more than one possible match (cf. Figure 4.14(e)). Hence, a matching probability is computed for each candidate scribble pair. In particular, the colour correlation is calculated by comparing the colour histograms of points belonging to the (grouped) start scribble s_i and to the (grouped) end scribble e_i:

$$m_{col}(H_s, H_e) = \frac{\Sigma_n (H_{s_i}(n) - \bar{H}_{s_i})(H_{e_i}(n) - \bar{H}_{e_i})}{\sqrt{\Sigma_n (H_{s_i}(n) - \bar{H}_{s_i})^2 \Sigma (H_{e_i}(n) - \bar{H}_{e_i})^2}} \tag{4.8}$$

Here, H_{s_i} is the colour histogram with a total number of N bins of the (grouped) start scribble s_i and H_{e_i} the colour histogram of the (grouped) end scribble e_i. Then n is the n^{th} bin of the start or end scribble histogram and $\bar{H}_k = \frac{1}{N} \Sigma_n H_k(n)$. The resulting metric values of $m_{col}(H_s, H_e)$ range between 0.0 and 1.0. The larger the value, the larger is the colour similarity between the two (grouped) scribbles. Because the start and end scribbles of a scribble pair belong to the same object, the colour similarity should be large and the (grouped) start and end scribble with the largest match are joined to a scribble pair (cf. Figure 4.14(e)). However, it is possible that scribbles which are not match are in the same segment. Therefore, a threshold t_{colour} is defined, and only pairs with a correlation higher than t_{colour} are joined to a scribble pair. The threshold is set to $t_{colour} = 0.6$ by default. Following this matching procedure, single start scribbles (cf. Figure 4.14(f) yellow framed), grouped start scribbles, single end scribbles (cf. Figure 4.14(f) blue framed), grouped end scribbles and scribble pairs with scribbles from the first and last frame (cf. Figure 4.14(f) green framed) can be identified. Objects represented by such scribble pairs with different depth values in the first and last frame are defined by the user to change depth over the time.

(iii) Adapt cost computation

Since there are now (grouped) start scribbles, (grouped) end scribbles and scribble pairs, the cost computation has to be adapted. In particular, the pixels used to built the foreground histogram $H_{f,l}$ and the background histogram $H_{b,l}$ are newly defined. Additional to the colour information of the user-assigned scribbles on the first frame, the colour information of the user-assigned scribbles on the last frame is used as well. The foreground histogram $H_{f,l}$ is built from all pixels covered by the (grouped) start scribble, (grouped) end scribble or the scribble pair l. In the case of a scribble pair, pixels from the first and last frame are used to build the foreground histogram. Similarly, the background histogram $H_{b,l}$ is built, i.e., all pixels from scribbles on the first and last frame not belonging to the current scribble l are used.

(iv) Determining a depth order

In order to define the depth change of an object corresponding to a scribble pair, a depth order in each frame of the video is determined by analysing the motion in the video, i.e., occlusions that are caused by moving objects. For example, the occlusion of an object A by another object

B is identified, i.e., B is in front of A. Thus, *depth restrictions* for the depth change can be defined, i.e., $B > A$, the depth value of object A is not allowed to be larger than the depth value of object B, because it is occluded by B. Those depth restrictions are important for achieving a perceptually consistent depth change.

As mentioned above, a rough depth order of a frame can be established by analysing the motion in the video. Specifically, motion-caused occlusions and disocclusions carry these depth order restrictions. The depth order and depth restrictions are defined for each frame separately in three steps: (i) a pairwise depth order is estimated by finding occlusions and disocclusions between two segments corresponding to a (grouped) scribble or scribble pair, (ii) the relative depth orders of a frame are stored in a directed acyclic graph (DAG) in order to achieve a global depth order for each frame, and (iii) depth restrictions are defined due to the order in the DAG. Below, these three steps are explained in detail.

In order to identify occlusions/disocclusions between two segments, those segments were obtained by previous steps in the 2D-to-3D conversion. Thus, the segments used for the pairwise occlusion/disocclusion detection are either from the CVF or from the motion segmentation. Their difference and characteristics are explained below:

CVF segments A segment corresponds to a (grouped) scribble or scribble pair, which results from CVF. In particular, a segment resulting from CVF is based on the user-assigned scribbles and covers pixels which are similar in colour and close to the user-assigned scribble. Hence, a pixel in frame $I(t)$ belongs to exactly one segment corresponding to scribble l, which enables finer depth ordering, e.g., in front of slanted surfaces, since partial occlusions of a moving object in front of a slanted surface can be detected. However, if the resulting segmentation from CVF is noisy, the depth order can be erroneous. Figure 4.15(e)-(h) shows resulting CVF segments of the brown slanted surface due to the input scribbles shown in (a) and (d). Due to of the separated segments, partial occlusions and disocclusions of the brown slanted surface that were caused by blue rectangle can be detected.

Motion segments These segments result from the motion segmentation. Contrary to the CVF segments, motion segments are based on similar motion and not on colour. Since motion segments covered by a user-assigned scribble are joined during the motion segmentation procedure, the user can influence these segments with the scribbles as well. Hence, as mentioned above, a motion segment can correspond to one or more (grouped) scribbles or scribble pairs. Thus, a slanted surface typically is represented by one motion segment, since the whole surface is moving and has the same motion, with more than one scribble with different assigned depth values, e.g., as shown in Figure 4.15(i). However, each of the scribbles on the slanted surface corresponds to the same motion segment. When an object A moves in front of a slanted surface B and a partial occlusion is detected, the resulting depth order would be $A > B$, even if A occludes only the furthest part of B, e.g., Figure 4.15(i). Since the brown slanted surface in Figure 4.15 is used as shown in (i) for the occlusion/disocclusion detection, it is assumed that the blue rectangle occludes the whole slanted object, even if only the furthest part is occluded.

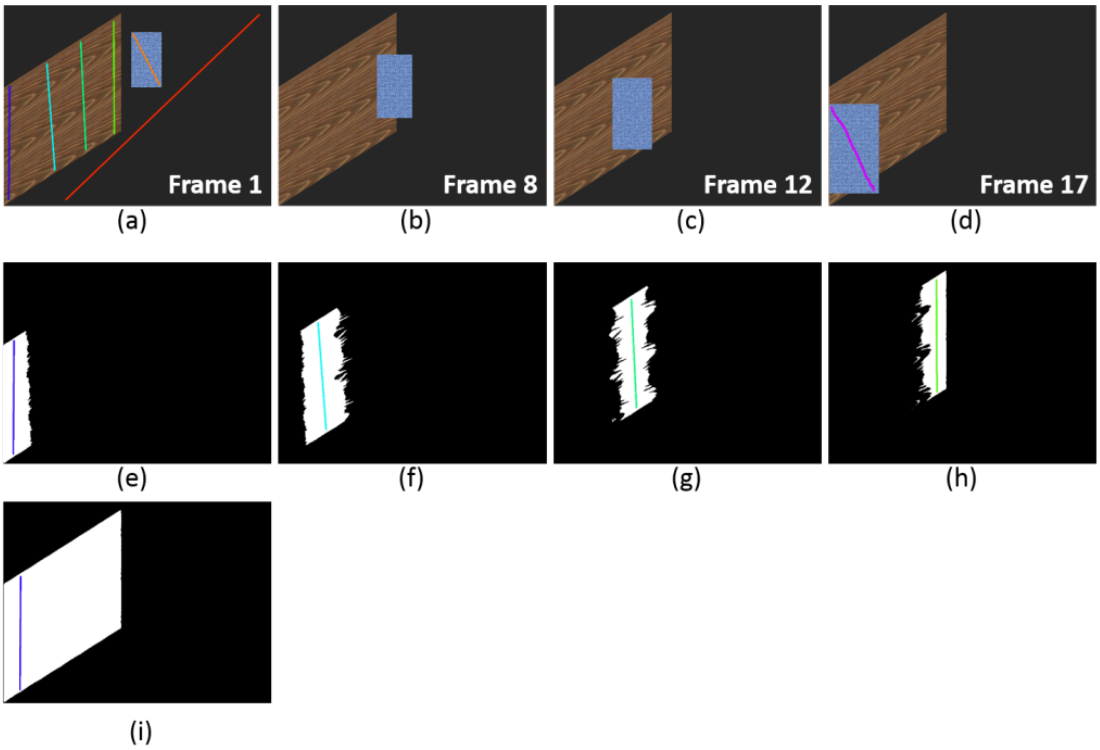


Figure 4.15: A video with a slanted surface (brown object). (a)-(d) show the first frame with user-assigned scribbles (coloured lines), frame 8, frame 12 and the last frame with one user-assigned scribble (coloured line) of a video with 17 frames. As can be seen, the blue rectangle moves in front of the brown slanted surface closer to the camera. (e)-(h) show the separated segments of frame 1 resulting from CVF of the brown slanted surface with a fix assigned depth value, which is used to determine depth restrictions. Since the brown slanted surface is separated in multiple segments, each with different assigned depth values (coloured lines), a step-by-step occlusion/disocclusion detection of each separate segment can be performed, which results in a smooth depth change of the blue moving rectangle. (i) shows the resulting motion segment of the brown slanted surface in frame 1 with a fix assigned depth value, which is used to determine depth restrictions. The fix assigned depth value is the largest user-assigned depth value on the slanted surface, since the maximum occluded depth value is used to determine the minimum restriction of the blue moving rectangle. Furthermore, no partial occlusions can be detected by using the motion segment. Thus, no matter which part of the slanted surface is occluded by the blue rectangle, it is supposed that the whole brown slanted surface is occluded.

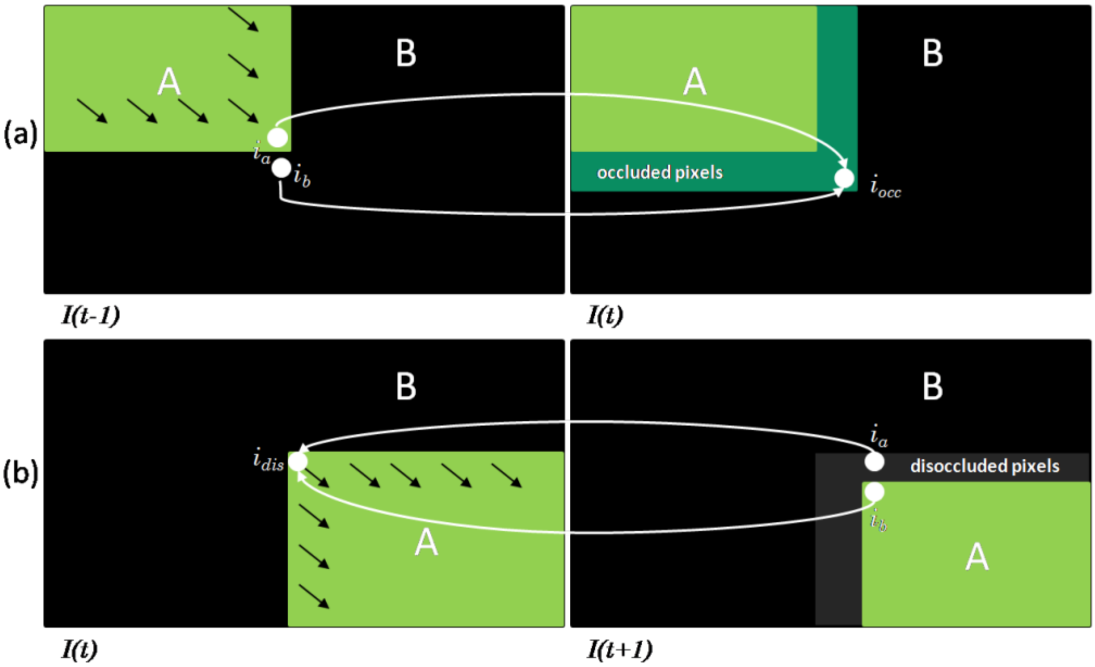


Figure 4.16: Illustration of occlusions and disocclusions. In both representations, i.e., (a) and (b), A is an object moving in front of the background B to the right bottom direction. Forward flow vectors in (a) from $I(t-1)$ to $I(t)$, and backward flow vectors in (b) from $I(t+1)$ to $I(t)$, of pixels i_a and i_b are illustrated by white arrows. (a) shows the occlusion of pixels from frame $I(t-1)$ to $I(t)$. Both pixels, $i_a = (x1, y1, t-1)$ and $i_b = (x2, y2, t-1)$, move to the same pixel $i_{occ} = (x3, y3, t)$ in frame $I(t)$. The dark green region in $I(t)$ illustrates occluded pixels, i.e., pixels which were visible in $I(t-1)$ and are not visible in $I(t)$. The occluding segment can be identified at pixel i_{occ}. Thus, $A > B$. (b) shows the disocclusion of pixels from frame $I(t)$ to $I(t+1)$. Both pixels, $i_a = (x1, y1, t+1)$ and $i_b = (x2, y2, t+1)$, move to the same pixel $i_{dis} = (x3, y3, t)$ in frame $I(t)$. The grey region in $I(+1)$ illustrates disoccluded pixels, i.e., pixels which were not visible in $I(t)$ and are visible in $I(t+1)$. The occluding segment can be identified at pixel i_{dis}. Thus, $A > B$.

Those segments are further used to determine the depth order by performing an occlusion/disocclusion detection. Several algorithms in the literature (e.g., [63], [82]) propose depth ordering systems by finding occlusions between a pair of segments. To identify occlusions, the algorithms of [63] and [82] use an optical flow estimation. As described in Section 4.2, occlusions can be detected by a consistency check of the forward and backward flow. However, to identify which segment is occluded and, thus, is behind the occluding segment, more information is needed. Similar to the approach in [63], we use three frames $I(t-1)$, $I(t)$ and $I(t+1)$ to detect occlusions and disocclusions. *Occluded* pixels are pixels which become invisible from $I(t-1)$ to $I(t)$, and *disoccluded* pixels are pixels which become invisible from $I(t+1)$ to $I(t)$. In Figure 4.16(a), the occlusion of pixels from frame $I(t-1)$ to frame $I(t)$ is illustrated.

Object A moves diagonally to the right bottom of the image and thus, occludes pixels of object B in $I(t)$, which are at the right and bottom edge (cf. Figure 4.16 dark green region). Similarly, Figure 4.16(b) shows the disocclusion of pixels from object B from frame $I(t)$ to $I(t+1)$ (cf. Figure 4.16 grey region). As suggested by the authors of [63], we use the bijection property of the optical flow w_t to detect occluded pixels. Thus, in the non-occlusion case, the optical flow w_{t-1} creates a bijection between $I(t-1)$ to $I(t)$, i.e., the flow vector of exactly one pixel from $I(t-1)$ points to a pixel i_x in $I(t)$. However, when there is an occlusion, there are two pixels $i_a = (x1, y2, t-1)$ and $i_b = (x2, y2, t-1)$ pointing to the same pixel $i_{occ} = (x3, y3, t)$ in frame $I(t)$. This case is shown in Figure 4.16(a). Similarly, the backward optical flow $\bar{w}_{t+1}$ from the frame $I(t+1)$ to $I(t)$ is used to detect disoccluded pixels (cf. Figure 4.16(b)). According to [63], an occlusion is identified, if

$$i_a + w_{i_a,t-1} = i_b + w_{i_b,t-1} = i_{occ}, i_a \neq i_b, \tag{4.9}$$

and a disocclusion, if

$$i_a + \bar{w}_{i_a,t+1} = i_b + \bar{w}_{i_b,t+1} = i_{dis}, i_a \neq i_b. \tag{4.10}$$

After detecting occluded and disoccluded pixels between two segments m and n, the occluding segment k can be identified, which is the segment at pixel i_{occ} in the occlusion case, or pixel i_{dis} in the disocclusion case in frame $I(t)$. Therefore, segment k is in front of the other segment. Due to estimation errors of the optical flow estimation, it might occur that there are misleading occlusion or disocclusion detections. Therefore, N_m is the number of pixels i_{occ} or i_{dis}, that determine segment m to be in front and vice versa N_n. To establish occlusion/disocclusion relations between segments, the occluding segment k is chosen as follows:

$$k = \begin{cases} m, & N_m > N_n. \\ n, & N_n > N_m. \end{cases} \tag{4.11}$$

Vice versa the segment l, which is behind the segment k, is defined. Additionally, a *confidence value* d_{conf} is defined to determine the strength of the depth order relation. This value is used to achieve a frame-wide depth order and to eliminate contradicting depth order relations. If such contradicting depth order relations occur, the confidence value is used to decide for the strongest depth order relation to be true. Thus, the depth order relation with the smaller confidence value d_{conf} is not taken into account for the frame-wide depth order. The confidence value d_{conf} is defined as follows:

$$d_{conf} = abs(N_m - N_n). \tag{4.12}$$

The occlusion/disocclusion detection described above is used for spatial neighbouring segments in a frame to determine their relative depth order. Therefore, detection is only performed for segments which are in the same frame and occlude each other.

Subsequent to the pairwise occlusions/disocclusions detection, the resulting relative depth orders of a single frame t are stored in the DAG of frame t in order to achieve a global depth order per frame, similar to the depth ordering algorithms of, e.g., [63] or [82]. In this graph, nodes represent segments and directed edges between nodes represent the detected depth order

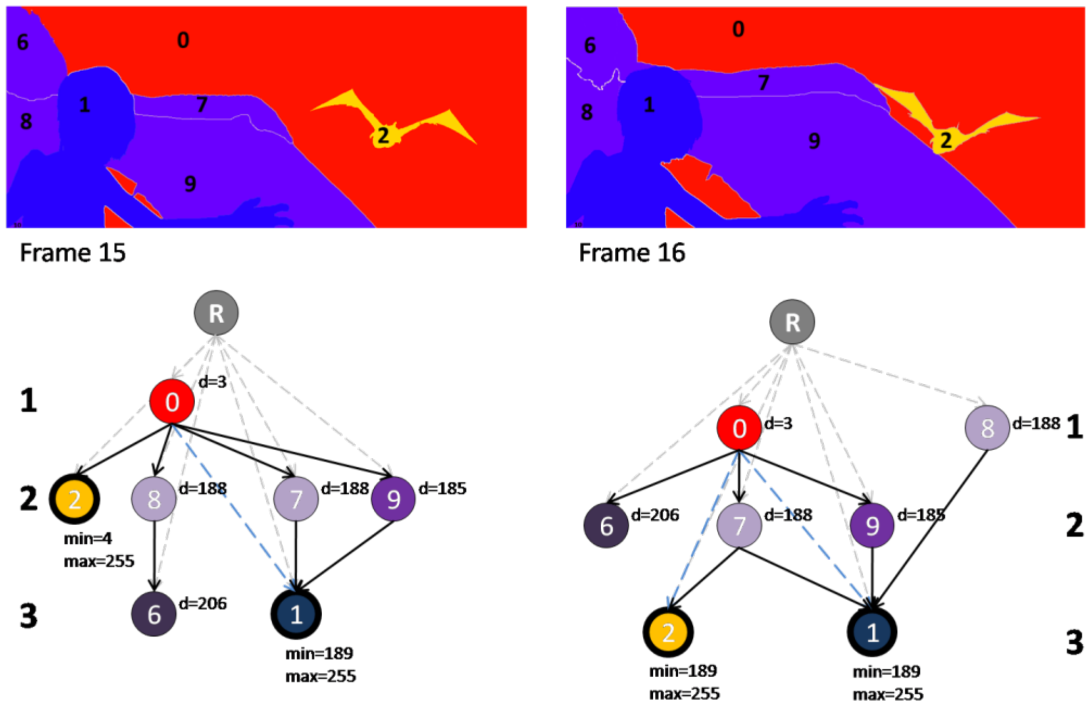


Figure 4.17: Illustration of pairwise checked segments and the resulting depth order graphs of frames 15 and 16 of the video shown in Figure 4.12. In the first row the resulting motion segments of each (grouped) scribble and scribble pair at frame 15 (left) and frame 16 (right) are shown. The corresponding resulting depth order graphs are shown below. Nodes at top are occluded by their child nodes, i.e., are further away from the camera. While each (grouped) scribble has a fixed assigned depth value (nodes 0, 6, 7, 8 and 9), scribble pairs (nodes 1 and 2) have minimum r_{min} and maximum r_{max} restrictions assigned resulting from their parent and child nodes depth values. Additionally, the depth order graphs are built regarding the determined depth levels of each node, i.e., the longest path from the artificial root node R to each node (grey dotted paths). Blue dotted paths also show detected depth order relations, but are not used to determine the depth level. Considering scribble pair/node 2, the change of the minimum restriction can be seen. Since the object corresponding to node 2 is moving in front of the object corresponding to node 9, its depth level and minimum depth value restriction r_{min} changes. Thus, while the object can have a minimum depth value of 4 in frame 15, i.e., it is only moving in front of the background, its depth value in frame 16 has to be at least 189, i.e., it is now moving in front of the big dragon's wing, which has a fixed assigned depth value of 188.

relations between two segments (e.g., Figure 4.17). Thus, parent nodes have a smaller depth value than their child nodes, i.e., they have a larger distance from the camera than their child nodes, also child nodes occlude their parent nodes. In order to avoid ambiguous depth order relations, cyclic edges are removed similar to [82]. Thus, each time a depth order relation is added to the graph, a *cycle check* is performed. Therefore, a depth-first-search strategy [80] is used to identify cyclic edges and the edges with the smallest confidence value d_{conf}, which was described above, is removed from the graph. Thus, a global depth order graph for each frame is established.

Finally, after adding all depth order relations of a single frame to the DAG of a frame, the *depth level* of each node in the DAG is determined by searching for the longest path from an artificial root node to each node in the graph. The artificial root node is additionally added and has a directed edge to all nodes in the DAG of a single frame. Since there is not always an explicit root node, the artificial root node is added in order to determine the longest path to each node in the graph. The depth level determines the maximal number of nodes one has to pass to reach a specific node, i.e., the hierarchy level in the graph. Moreover, the depth level of each node in a single frame is used together with the parent's and child's depth value to determine final depth restrictions in order that nodes at different depth levels differ at least by one in their assigned depth. To determine the depth level and thus, solve the longest path problem, the Bellman-Ford algorithm, [3], [28] is used. For this purpose, all edges have the same negative edge weight, which is set to -1. Thus, the longest path from the artificial root node to all nodes in the DAG of a frame can be determined. The distance from the artificial root node to each node agrees with the depth level (cf. Figure 4.17). The depth level $\lambda(i, t)$ of a segment (node) i in frame t is further used to define depth restrictions, which are identified in the next step. The incorporation of the depth level into the determination of the restrictions makes sure that the depth value of a scribble pair i is at least one depth value larger than the maximal parent node of i, and at least one depth value smaller than the minimal child node of i. This is especially important in the advanced case, that multiple moving scribble pairs occlude each other, which is described in the following in more detail.

Depth restrictions for scribble pairs in a single frame are identified by traversing the established DAG. Each of the (grouped) start and end scribbles, which do no change depth over time, have a fixed assigned depth value, e.g., scribbles (nodes) 0, 6, 7, 8 and 9 shown in the depth order graph in Figure 4.17. Therefore, these depth values are used to determine depth restrictions for scribble pairs, e.g., scribbles (nodes) 1 and 2 in the depth order graph in Figure 4.17. Thus, for each scribble pair i in frame t, restrictions to a minimal $r_{min}(i, t)$ depth value and a maximal $r_{max}(i, t)$ depth value are determined. These minimum and maximum depth restrictions define a range for each scribble pair in each frame. The depth values of the object change so that the assigned depth value $d(i, t)$ in frame t is in between the range of the determined minimum and maximum depth restrictions, i.e., $r_{min}(i, t) \leq d(i, t) \leq r_{max}(i, t)$. Considering these minimum and maximum restrictions, a depth change without causing perceptual conflicts can be established.

Since a scribble pair i occludes all its parent nodes, i has to have a depth value larger than the parent node, which is the closest to the camera, i.e., has the largest depth value. Thus, the minimum restriction is established by taking the maximum value of all parent nodes, i.e., all nodes corresponding to segments which are occluded by the current node i. Basically, the restriction of a scribble pair i in frame t to a minimal depth value of $r_{min}(i,t)$, i.e., the smallest depth value the scribble pair i is allowed to have at frame t, results from the maximal assigned depth value of all n parent nodes. Following, n is the total number of parent nodes of scribble pair i, and j is the parent node with the maximal assigned depth value. The minimal restriction value is determined by the maximal assigned depth value of parent node j plus the difference between the depth levels of the current node i and the parent node j, i.e., $|\lambda(i,l) - \lambda(parent(i,j,t))|$, where $0 < j < n$.

$$r_{min}(i,t) = \max_j (d(parent(i,j,t)) + |\lambda(i,t) - \lambda(parent(i,j,t))|) \tag{4.13}$$

Contrary, since a scribble pair i is occluded by all its child nodes, i has to have a depth value smaller than the child node, which is the farthest away from the camera, i.e., has the smallest depth value. Hence, the maximum restriction is determined by taking the minimum value of all child nodes, i.e., all nodes corresponding to segments, which are occluding the current node i. Thus, the restriction of a scribble pair i in frame t to a maximal depth value of $r_{max}(i,t)$, i.e., the largest depth value the scribble pair i is allowed to have at frame t, results from the minimal assigned depth value of all n child nodes. Following, n is the total number of child nodes of scribble pair i, and j is the child node with the minimal assigned depth value. The maximal restriction is determined by the minimal assigned depth value of child node j minus the difference between the depth levels of the current node i and the child node j, i.e., $|\lambda(i,t) - \lambda(child(i,j,t))|$, where $0 < j < n$.

$$r_{max}(i,t) = \min_j (d(child(i,j,t)) - |\lambda(i,t) - \lambda(child(i,j,t))|) \tag{4.14}$$

In order to handle the more advanced case that multiple moving scribble pairs occlude each other, a special conditioning is necessary. In case that a scribble pair is occluded by another scribble pair, it is neither possible to determine a maximum restriction for the occluded scribble pair nor to determine a minimum restriction for the occluding scribble pair. To this end, an order of examination has to be defined in order to determine and update temporal restrictions. Thus, nodes are traversed and restrictions are determined regarding their depth levels, in an ascending order. Considering the DAG, this order is chosen so that parent nodes are processed before their child nodes, i.e., segments which are farther away from the camera are examined before segments which are closer. Let us consider two scribble pairs A and B, where scribble pair A is occluded by scribble pair B in frame t, i.e., B is a child node of A in the DAG of frame t. Since B has no fixed depth value, a temporal restriction of the maximal depth value for A has to be defined. For this purpose, all child nodes of B are explored until a node F with a fixed assigned depth value is found. The maximum restriction of A is set to $r_{max}(A) = d(F) - |\lambda(A) - l(F)|$. The difference between the depth levels guarantees that all scribble pairs between A and the found node F have at least a range of one possible depth value, since the difference between the restriction values is always exactly 1. Moreover, a temporal restriction of the minimal depth value of B is set to $r_{min}(B) = r_{min}(A) + 1$. Considering the examination order mentioned above, depth values of the segment corresponding to node A are established before B. After the assignment A has a fixed assigned depth value at frame t and the temporally set restriction to the minimal depth value for B can be updated. Note that with these temporal restrictions and the update procedure, it is also possible to handle a video where all objects are moving in depth, i.e., all nodes in the DAG are scribble pairs. In that case, the root node R in the DAG starts with a restriction of the minimal depth value of $r_{min}(R) = 0$ and the node Z with the largest depth level has a restriction of the maximal depth value of $r_{max}(Z) = 255$. The minimum restrictions of all other nodes increase by one per depth level distance to R and vice versa, the maximum restrictions of all other nodes decrease by one per depth level distance to Z. Hence, at the end Z has a temporal restriction of the minimal depth value of $r_{min}(Z) = \lambda(Z)$. Subsequently, the depth change models are established by starting at node R. After each computed depth change model, the temporal restrictions of the remaining nodes are updated.

Thus, a possible depth range for interpolating the depth of each scribble pair is determined. The depth ranges are further used to establish a perceptually consistent depth change of objects in a video. The following section introduces depth change models, which incorporate depth order information and restrictions in order to model a depth change over time.

(v) Depth change model generation

After adding scribbles to the end frame, identifying matching scribble pairs, adapting the cost computation and determining depth orders of objects in the video, the actual computation of the depth change over time can be accomplished. Therefore, two models are presented: (i) the *basic depth change model* interpolates depth values between the user-assigned depth values at the first and last frame, and (ii) the *advanced depth change model* interpolates depth values by using the object's size to determine depth between the user-assigned depth values at the first and last frame. Both models incorporate the above established depth order information and the resulting depth restrictions in order to model perceptually consistent depth changes. In the following, the initialisation of both models is first described separately without considering the depth information. Then, the incorporation of the additional depth restrictions is presented. Please note, that objects closer to the camera have larger depth values than objects further away from the camera.

Basic depth change model The basic depth change model simply uses the user-assigned depth values of start and end frame of each scribble pair to interpolate depth values linearly between start and end frame. Thus, the depth value of scribble pair i in frame t is computed as follows:

$$d(i,t) = d(i,0) + \frac{d(i,n) - d(i,0)}{n} * t. \tag{4.15}$$

Here, n is the total number of frames in the video, $d(i,0)$ is the user-assigned depth value of scribble pair i in the first frame, and vice versa $d(i,n)$ defines the depth value in the last frame. Thus, a linear depth change is modelled, i.e., the depth value of the object changes by the same amount between each frame of the video. Hence, the object appears to come closer or go further away with the same velocity throughout the video, which is reasonable for a person, e.g., walking continuously away from the camera. However, if the person stops walking, pauses and then continues walking the basic depth change model produces

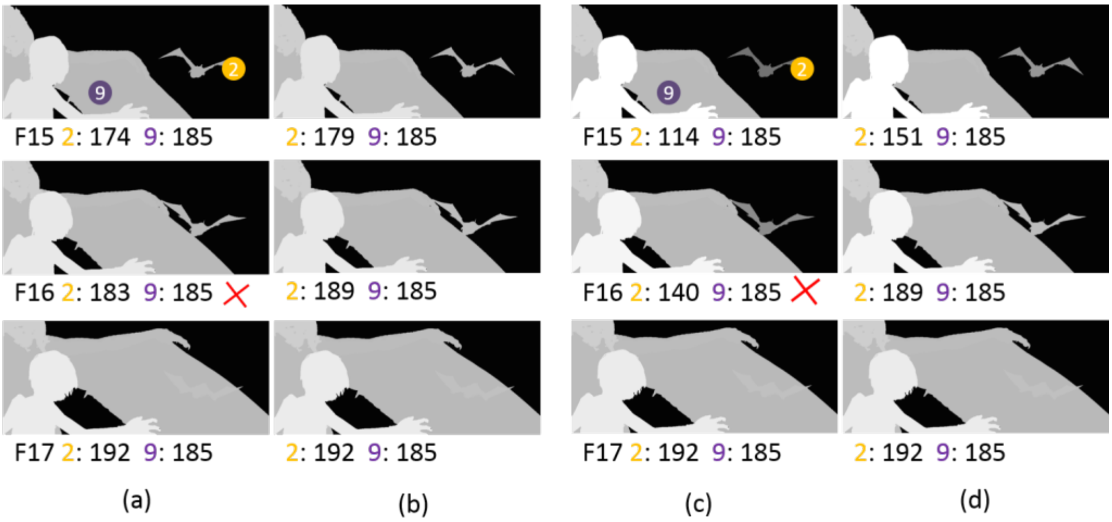


Figure 4.18: Results of the basic ((a), (b)) and advanced ((c), (d)) depth change model. Here, the depth values of the two objects corresponding to segments 2 and 9 are compared in frames 15, 16 and 17. (a) shows the resulting depth maps (WTA approach) using the basic depth change model without the incorporation of the depth order information. As can be seen in frame 16, this leads to conflicting depth values. The object corresponding to segment 2 (cf. Figure 4.17) has a smaller depth value propagated than the object corresponding to segment 9, although segment 2 is in front of segment 9. (b) shows the resulting depth maps (WTA approach) using the basic depth change model with the incorporation of the depth order information. As can be seen in frame 16, the previous conflict is eliminated and both objects have perceptually consistent depth values propagated. (c) shows the resulting depth maps (WTA approach) using the advanced depth change model without the incorporation of the depth order information. As can be seen, propagated depth values differ to values in (a), since an object's size is considered to interpolate depth. However, resulting depth values in frame 16 induce a conflict again. (d) shows the resulting depth maps (WTA approach) using the advanced depth change model with the incorporation of the depth order information. The previous conflict is eliminated.

unrealistic depth changes, since the person's depth values keep changing even when the person stopped walking. To additionally handle the second case, we further propose an advanced depth change model computation.

Advanced depth change model The idea of the advanced depth change model is to consider the case of an irregularly moving object as described above. To this end, the size of the object is incorporated in the interpolation. The object size is one of several pictorial depth cues in a 2D image [91]. Since the size of an object can be determined in each frame and, additionally, the depth value of the object's size in the first and last frame is known, it is possible to interpolate depth values corresponding to a determined object size between the object's start size and the object's end size. Since an object becomes larger the closer it moves to the camera, the depth values of an object coming closer become larger as well. Thus, the depth value of scribble pair i with an object size of $s(i,t)$ in frame t is computed as follows:

$$d(i, s(i,t)) = d(i, s(i,0)) + \frac{d(i, s(i,n)) - d(i, s(i,0))}{s(i,n) - s(i,0)} * (s(i,t) - s(i,0)). \quad (4.16)$$

Here, n is the total number of frames in the video, $d(i, s(i,0))$ is the user-assigned depth value of scribble pair i with the start size $s(i,0)$ in the first frame 0, while $d(i, s(i,n))$ is the user-assigned depth value of scribble pair i with the end size $s(i,n)$ in the last frame n.

Hence, the depth values between two frames only change when the object's size changes, i.e., the object is moving closer or further. Additionally, the depth change depends on the observed object motion. Moreover, when an object moves horizontally, i.e., its size does not change, or moves in another way without changing its size, e.g., a ball falling down in front of the camera, the depth values do not change between two frames. Additionally, it is also possible to model a movement that under- or overruns the start or end size of an object. For example: Object A starts with size $s(0) = 10$ and depth value $d(A, s(A,0)) = 100$ in the first frame. A moves closer to the camera until it has a size of $s(A,10) = 20$ in frame 10, and then moves away again. The size in end frame $n = 15$ is $s(A,15) = 17$ and the known depth in the last frame is $d(A, s(A,15)) = 180$. Therefore, the depth value of A in frame 10 can be calculated and is $d(A, s(A,10)) = 214$. With the basic depth change model, such a movement can not be modelled because only values between the start and end depth are possible. Thus, the advanced depth change model enables a depth change based on the object's movement structure. Note that if the depth change indicated by the user does not agree with the depth change indicated by the object size, i.e., the object is coming closer regarding the depth values in the first and last frame, but becomes smaller regarding the object size in the first and last frame, the advanced depth change model is not applied for this object. Instead, the basic depth change models that does not take the object size into account is used.

To determine the object size, two approaches are implemented: (i) by using the resulting CVF segment of the scribble corresponding to the object, or (ii) by analysing paths of the user-assigned scribbles corresponding to the object throughout the video. In both cases, the object size corresponds to the height of the object, because typically growth in vertical direction delivers the necessary information about the movement in depth. Since an object can become larger in horizontal direction without coming closer to the camera, e.g., a person stretching out the arm to the left side, growth in X-direction is not usable to determine motion in depth[2].

CVF segments This method basically examines the resulting CVF segment of scribble pair i in frame t. As described above, a segment resulting from CVF is based on colour regarding the user input. In order to determine the object size of a CVF segment, the largest $y_{max}(i,t)$ and lowest $y_{min}(i,t)$ points of the segment are detected. Thus, the object size $s(i,t)$ of scribble pair i in frame t is defined as follows:

$$s(i,t) = y_{max}(i,t) - y_{min}(i,t). \tag{4.17}$$

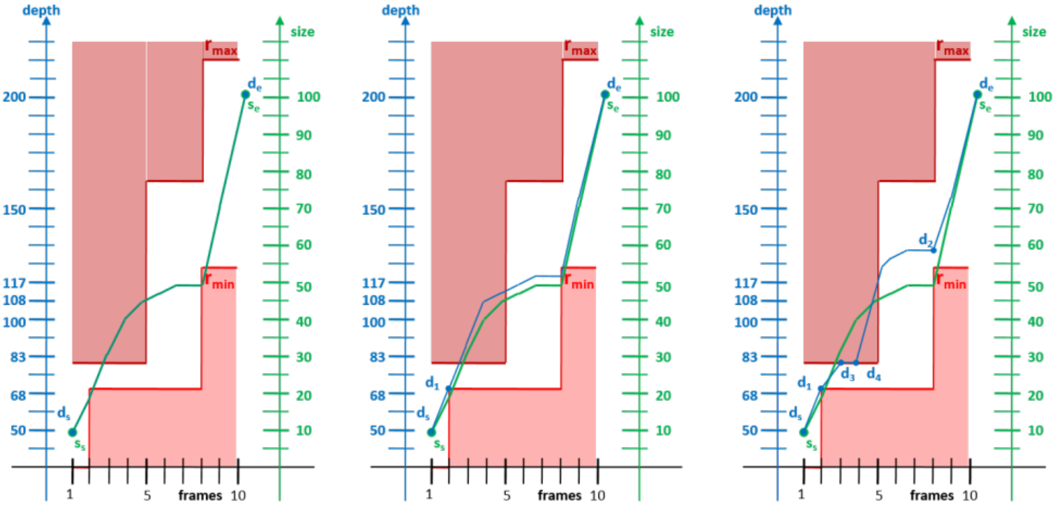
Thus, the size of fixed sized objects visible throughout the whole video can be determined. It is worth to note that this approach is limited by partial occlusions of objects. If an object is only partly visible, e.g., until frame 5, and then becomes fully visible in frame 6, there is a big jump between the object size between frame 5 and 6. Moreover, the resulting depth values can cause an unrealistic depth change.

Path object size This approach makes use of the long motion paths. Therefore, all long motion paths belonging to pixels of the user-assigned scribbles going throughout the whole video, are chosen to determine the size, i.e., those paths have the same length as the video has frames. Next, the highest $y_{max}(i,t)$ and lowest $y_{min}(i,t)$ points are detected. This time, points from the chosen paths of scribble pair i in frame t are examined to detect those points. Contrary to the approach above, this method determines size by only considering points which are visible throughout the whole video. Thus, the size of, e.g., a partly occluded object, which becomes fully visible after some frames, can be determined. However, the scribble has to cover parts of the corresponding segment which are visible throughout the whole video, and to achieve a good result, at least 30 long motion paths belonging to the scribble pair and having the video's length are necessary. Note that if no long motion paths are found, the basic depth change model is generated instead.

In the following, we discuss the effect of the approaches described above in more detail using examples. With the advanced depth change model it is possible to model a more realistic object movement in depth. However, if the object occludes or is occluded by another object, it is important that the depth values are consistent with the depth order, i.e., the front object's depth has to be larger than the back object's depth. Otherwise a perceptual conflict is the result, e.g., Figure 4.18(a) and (c). As can be seen in Figure 4.18, the small dragon is moving closer and occludes from frame 16 on the wing of the big dragon. Thus, the depth value of the small dragon has to be larger than the depth value of the wing of the big dragon starting from frame 16. In Figures 4.18(a) and (c) it can be seen that this requirement is not fulfilled, neither with

---

[2] Similar cases, e.g., a person stretching its arm upwards, can occur as well. However, such a case appears less often.



(a) (b) (c)

Figure 4.19: Illustration of the frame-wise restrictions check for the advanced depth change model of an object. Minimum r_{min} and maximum r_{max} restrictions are presented. Light and dark red shaped areas showing the range of depth values, which results in conflicts. Thus, the interpolated values are not allowed to be in any red shaped area. The green graph with start size point s_s and end size point s_e illustrates the object size at each frame. As can be seen, regarding the object's size the object starts moving constantly (frames 1 to 5), then slows down (frames 5 to 7), stops a moment (frames 7 and 8) and continues with a fast movement (frames 8 to 10). The blue graph with start depth point d_s and end depth point d_e illustrates the assigned depth values at each frame. (a) shows the interpolation graph after initialising depth values using the advanced depth change model. Conflicting depth values can be identified at frame 2 and frame 8 caused by minimum restrictions and at frames 3 to 5 caused by maximum restrictions. (b) shows the updated interpolation graph after adding an interpolation point d_1 at frame 2, because of an identified conflict regarding the minimum restriction at this frame. (c) shows the final interpolation graph after adding in total four new interpolation points and updating after each identified conflict. The interpolation points are numbered due to their adding and updating the interpolation graph. Thus, interpolation points d_1 and d_2 were added due to a found conflict with the minimum restriction at frame 2 and frame 8, respectively. Interpolation points d_3 and d_4 were added due to a found conflict with the maximum restrictions at frames 3 and 4. The final interpolation graph does not show any violations any more.

the basic depth change (cf. Figure 4.18(a)) model nor with the advanced depth change model (cf. Figure 4.18(c)) without considering the depth order. Therefore, the above established depth order information and depth restrictions are incorporated in both depth change models in a next step.

Incorporating depth restrictions In a first step depth values are initalised as described above for the chosen deph change model, without condidering depth restrictions. In a next step, a frame-wise restriciton check is accomplished. Basically, this restriction check is a recursive method, alternating between checking violations against the minimum and maximum restrictions in a bounded range after a conflict is found until all frames are verified to have valid depth values. If a conflict is found a new fix assigned depth value is added and the depth change model is updated. A demonstration of the step-by-step incorporation for the advanced depth change model is presented in Figure 4.19. In Figure 4.19, the movement of the object is illustrated by the green graph and the initalised depth values by the blue graph. According to the size of the object, it moves irregular. As can be seen in Figure 4.19 the object moves constantly in the beginning (frames 1 to 5), then slows down (frames 5 to 7), stops (frames 7 and 8), and moves fast (frames 8 to 10) in the last frames. Additionally, the minimum and maximum restrictions are illustrated in Figure 4.19, i.e., r_{min} and r_{max}. Figure 4.19(c) shows a resulting interpolation graph after the frame-wise restriction check. As can be seen, four conflicts were found in total (points d_1 to d_4), where the points are numbered by the order of detection of conflicts. In Figure 4.19(c) no more violations can be identified, and, thus, a perceptual consistent depth change is established.

As mentioned above, if a scribble pair A is occluded by another scribble pair B the minimum restriction value is set to a temporally value and has to be updated after the determination of the depth values of A. Therefore, the depth change models for each scribble pair are examined in the depth order of the first frame and after each examination the restriction values of all child nodes are updated. Thus, scribble pair B has valid minimum and maximum restrictions, when its depth change model is established.

Experimental Results

Figure 4.18 gives an example of the effect of both established depth change models. In this example, ground truth optical flow [16] was used. Segments used to detect pairwise occlusions/disocclusions result from the motion segmentation. To determine the object's size for the advanced depth change model, the path object size approach, which analyses paths of the user-assigned scribbles throughout the video, is used. This approach was chosen because the small dragon moves its wings up and down throughout the video, which would result in large and small alternating object size values, when the CVF segment size would be used. As already discussed above, it can be seen in Figure 4.18(a) and (c) that without incorporating the depth order in the depth change models, a perceptual inconsistent depth change is the result.

Another example of the effect of the temporal depth change extension is given in Figure 4.20, a video with two objects, a brown slanted object and a blue rectangle. In this example optical flow is estimated with the method of [51]. Segments used for the pairwise occlusions/disocclusions detection result from CVF. Since the rectangle's form is fixed and not partly occluded throughout the video, the CVF segment object size approach is used to determine the object's size for the advanced depth change model. The blue rectangle is moving in front of the brown slanted surface closer to the camera. As can be seen in Table 4.1 the blue rectangle's

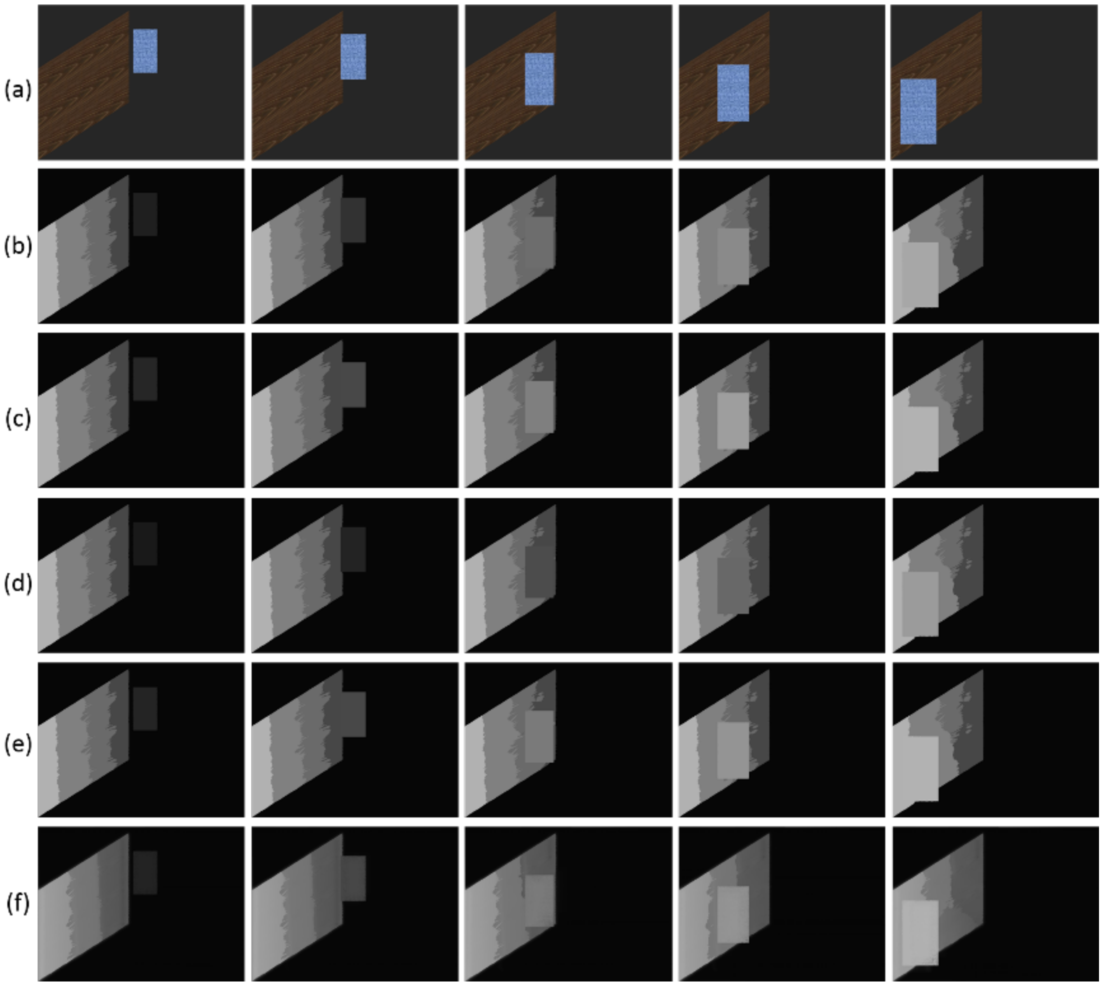


Figure 4.20: Depth change results of a video with 17 frames and a slanted surface. (a) shows frames 2, 4, 10, 13 and 16. (b) WTA approach results from the basic change model without incorporation of the depth order, (c) WTA approach results of the basic change model with incorporation of the depth order, (d) WTA approach results from the advanced change model without incorporation of the depth order, (e) WTA approach results from the advanced change model with incorporation of the depth order and (f) depth blending approach results with the $n = 2$ lowest cost volume slices averaging the propagated depth value from the advanced change model with incorporation of the depth order.

frame (size)	B,N	B,Y	A,N	A,Y	r_{min}	r_{max}
1 (123)	21	21	21	21	-	-
2 (125)	31	38	25	36	7	255
3 (128)	40	55	31	57	7	255
4 (130)	**50**	72	**35**	72	72	255
5 (134)	**60**	81	**43**	84	72	255
6 (136)	**70**	90	**47**	90	72	255
7 (138)	80	98	**51**	95	72	255
8 (142)	**89**	107	**59**	107	107	255
9 (145)	**99**	109	**66**	113	107	255
10 (150)	109	127	**76**	122	107	255
11 (154)	**119**	137	**84**	129	129	255
12 (159)	**128**	148	**94**	138	129	255
13 (164)	138	158	**104**	148	129	255
14 (169)	148	168	**114**	157	129	255
15 (179)	**158**	178	**136**	178	178	255
16 (188)	**167**	178	**155**	178	178	255
17 (200)	178	178	178	178	-	-

Table 4.1: Detailed illustration of the established depth change model of the blue rectangle shown in Figure 4.20. The first column shows the frame and the object's size per frame. Columns annotated with **B** show the results of the basic depth change model without (N) and with (Y) incorporated depth order. Columns annotated with **A** show the results of the advanced depth change model without (N) and with (Y) incorporated depth order. r_{min} and r_{max} show the determined depth order restrictions per frame. Conflicting depth values are bold.

size changes irregularly between frames. The user-assigned depth values of the brown slanted surface are from right to left, i.e., from back to front, 71, 106, 128 and 177. The propagated depth values per frame of the blue rectangle can be seen in detail in Table 4.1. Bold printed depth values show perceptually conflicting depth values due to the determined depth restrictions r_{min} and r_{max}.

As shown in Table 4.1, without incorporating the determined depth order conflicting depth values are propagated. Thus, the blue rectangle has at, e.g., frame 4 a lower depth value assigned than the brown slanted surface, which is occluded by the blue rectangle. The resulting 3D view would induce a perceptual inconsistency and the blue rectangle would appear to be further away than the brown slanted surface, although occlusions cues induce the opposite case. Furthermore, the difference between the basic (b) and advanced (A) depth change model can be seen in Table 4.1 in columns B,N and A,N. While the basic depth change model increases the propagated depth value equally from one frame to the next, i.e., the interval is about 9 due to rounding, the advanced basic depth change model increases the propagated depth values from one frame to the next according to the object's size at each frame. Thus, it changes irregularly, since the size of the blue rectangle increases in irregular intervals as well. Regarding the size, the depth of the blue rectangle increases more slowly than by using the basic depth change model. However, both models induce conflicting depth values without incorporating depth order. As shown in columns B,Y and A,Y those conflicts are eliminated after the depth order is incorporated in both depth models. Thus, a perceptually consistent depth change of the blue rectangle is established and the resulting 3D view would show a smooth movement of the blue rectangle towards the camera.

4.5 Summary

The 2D-to-3D conversion algorithm proposed in this chapter performs a semi-automatic depth propagation based on only little user input. By annotating scribbles on the first and last frame of the video, the user assigns depth values which are propagated to the video based on a cost volume filtering-based video object segmentation (cf. Section 4.1). For each user-assigned scribble, costs are computed for each pixel regarding their similarity to pixels covered by the scribble. Thus, the final depth of the scribble with the lowest costs is propagated. The proposed algorithm focuses on several problems by applying the above introduced extensions. The temporal consistency improvement (TCI) addresses the problem of edge-sharpness and temporal consistency by improving depth propagation at depth boundaries in videos with fast moving objects by filtering along the objects' movement throughout the video (cf. Section 4.2). The spatial influence extension (SIE) addresses wrong propagations caused by segmentation issues. The SIE reduces wrong depth assignments by adding a spatial influence term to the cost computation. Thus, pixels which are similar in colour but different in depth have less influence on the final depth propagation (cf. Section 4.3). Finally, the temporal depth change extension (TDCE) addresses temporal depth changes and enables perceptually consistent depth changes of objects throughout the video by generating depth change models that achieve a significant improvement in videos with movement in depth.

CHAPTER 5

Evaluation

In this chapter we evaluate our proposed temporally coherent cost volume filtering-based depth propagation algorithm using ground truth depth maps. The evaluations are performed systematically for all described extensions (cf. Chapter 4) of the proposed algorithm. Additionally, a comparison with a similar depth propagation algorithm [42] is performed. Section 5.1 discusses our evaluation strategy, including the dataset and the error metrics that are used in our evaluations. In Section 5.2 the impact of each extension is quantitatively evaluated using the evaluation strategy mentioned before. Furthermore, we show and discuss various depth maps that were generated with the 2D-to-3D conversion algorithm that is proposed in this thesis. In Section 5.3 we quantitatively and visually compare our proposed depth propagation algorithm to a related 2D-to-3D conversion algorithm, i.e., [42]. Depending on the characteristics of the video, we show that each of the extensions can achieve a significant improvement. Moreover, high-quality results can be achieved by using a robust optical flow estimation and a robust scribble annotation by the user.

5.1 Evaluation Methodology

In total, the algorithm is applied to 19 test videos. The results of all 19 test videos were obtained using user-assigned scribbles on the first and last frame to propagate depth. Generally, the used test videos can be separated in two sets: (i) eight test videos from the MPI-Sintel data set [16] used to quantitatively evaluate the impact of each extension in Section 5.2, and (ii) 11 test videos from the data set used in [42] to quantitatively compare our resulting depth maps with results of [42]. In the following, data of both sets are presented in more detail.

MPI-Sintel Dataset [16]

Sintel [68] is an animated open source short film created by Blender [1]. [16] modified the film in order to achieve a useful data set for optical flow evaluation with naturalistic video sequences addressing challenges such as large motion, long sequences or motion blur. The MPI-Sintel data

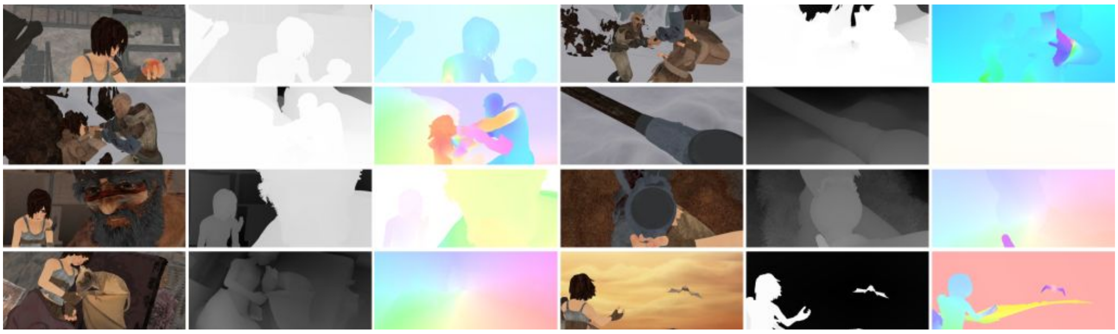


Figure 5.1: Test videos from the MPI-Sintel data set [16]. The first frame, the corresponding depth ground truth and corresponding optical flow ground truth of the eight test videos used in the evaluation are shown. Test videos (from left to right, two test videos in a row): *alley1*, *ambush2*, *ambush5*, *ambush7*, *shaman2*, *shaman3*, *sleeping1* and *temple3*.

Nr.	Name	Frames	Resolution	$t_{spatial}$	c	min	OS
1	*alley1*	50	1024×436	256	450	10	Cvseg
2	*ambush2*	5	1024×436	256	250	10	Cvseg
3	*ambush5*	31	1024×436	256	250	10	path
4	*ambush7*	38	1024×436	256	250	10	path
5	*shaman2*	50	1024×436	256	250	40	path
6	*shaman3*	50	1024×436	256	250	10	path
7	*sleeping1*	48	1024×436	256	250	10	path
8	*temple3*	18	1024×436	256	300	40	path

Table 5.1: Test videos from the MPI-Sintel data set [16] using the *albedo* training set, which were used for the evaluation in Section 5.2. The table shows the name of the test video, the total number of frames of the test video, the resolution, the spatial threshold $t_{spatial}$ used for the spatial influence extension, the maximal segment size c and the minimal segment size min used for the motion segmentation, and the used approach to determine the object size (OS) used for the advanced depth change model of the temporal depth change extension.

set [16] contains optical flow ground truth and depth ground truth (cf. Figure 5.1) for each test video, which allows us to use the data set in the context of our evaluation. Both the optical flow and depth ground truth are used to evaluate the results in Section 5.2. The MPI-Sintel data set provides different level passes for each scene that gradually increase complexity. In the context of the evaluation, we use the *albedo* pass which is the simplest rendering pass and renders flat, unshaded surfaces that exhibit constant albedo over time [16]. However, since the MPI-Sintel data set [16] provides only forward optical flow vectors, in our algorithm only those are used to detect occlusions and further establish a depth order. Since optical flow ground truth is used, the forward flow vectors are sufficient to achieve a valid depth order. However, we also want to validate the impact of the optical flow in our proposed depth propagation algorithm. Therefore,

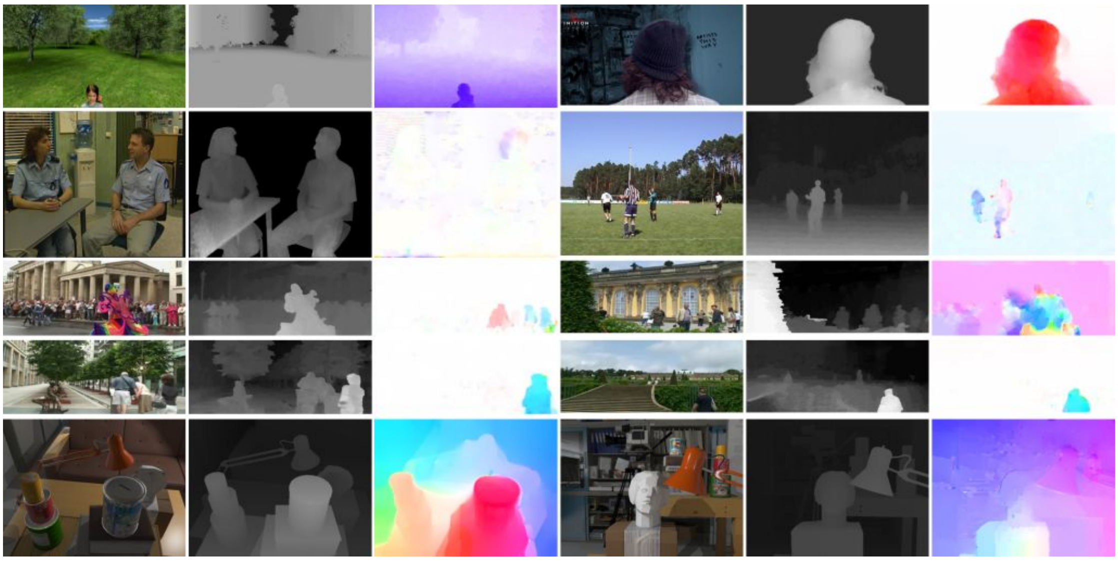


Figure 5.2: 10 of 11 test videos used to compare resulting depth maps with [42]. The first frame, the corresponding depth ground truth/reference depth and corresponding optical flow ground truth/estimated optical flow of the test videos used in the evaluation are shown. Test videos (from left to right, two test videos in a row): *Child* [10], *Head* [10], *Interview* [10], *Soccer* [12], *Parade* [12], *Palace* [12], *City* [12], *Staircase* [12], *Tsukuba380-397* [56] and *Tsukuba50-66* [56].

we compute an estimated optical flow by using the method of [51] and additionally compute depth maps by using the estimated forward and backward optical flow fields in the proposed depth propagation algorithm. The parameters set for the estimation of the optical flow stayed the same for all test videos. Ground truth depth values are resulting from the available depth video that is normalised by its maximal depth value. Thus, depth values can be compared with the results of the proposed depth propagation algorithm.

Figure 5.1 presents the eight test videos from the *albedo* training set of the MPI-Sintel data set [16] used to evaluate the impact of each extension in Section 5.2. Table 5.1 gives an overview of the attributes and parameters of the eight test videos. Generally, the MPI-Sintel data set provides video sequences with 50 frames per sequence. For the evaluation we used subsequences of test videos *ambush2* (frames 1-5), *ambush5* (frames 1-31), *ambush7* (frames 1-38), *sleeping1* (frames 1-48) and *temple3* (frames 5-22). Other test videos listed in Table 5.1 use the provided 50 frames of the video sequences. The user-assigned scribbles were annotated by the author and stayed unchanged for all test results in this chapter.

Comparison Dataset

Figure 5.2 presents the 11 test videos used to compare our proposed algorithm with the depth propagation algorithm from [42]. Table 5.2 gives an overview of the attributes and parameters of the 11 test videos. All test videos shown in Table 5.2 were used in [42] for evaluation as well. Test videos 1 to 5 are test videos from [12]. Reference depth maps for these test videos were

Nr.	**Name**	**Frames**	**Resolution**	$t_{spatial}$	**c**	**min**	**OS**
1	*City*	18	699 × 282	175	130	20	CVseg
2	*Parade*	11	689 × 282	172	130	20	CVseg
3	*Palace*	10	702 × 278	176	130	20	CVseg
4	*Staircase*	20	702 × 278	176	130	20	CVseg
5	*Soccer*	21	669 × 282	167	130	20	path
6	*Child*	21	600 × 338	150	130	20	Cvseg
7	*Head*	81	600 × 330	150	130	20	CVseg
8	*Interview*	101	600 × 480	150	130	20	CVseg
9	*Tsukuba50-66*	21	640 × 480	160	100	20	CVseg
10	*Tsukuba380-397*	18	640 × 480	160	100	20	path
11	*Tsukuba1-100*	100	640 × 480	160	100	20	path

Table 5.2: Test videos, which are used for the evaluation and comparison with the depth propagation algorithm from [42] in Section 5.3. The table shows the name of the test video, the total number of frames, the resolution, the spatial threshold $t_{spatial}$ used for the spatial influence extension, the maximal segment size c and the minimal segment size min used for the motion segmentation, and the used approach to determine the object size (OS) used for the advanced depth change model of the temporal depth change extension.

generated in [42] with a stereo matching algorithm from [6] for all frames. Test videos 6 to 8 are from [10]. For these test videos ground truth depth maps for some frames are available, but not for all frames as for the other test videos. Therefore, results of test videos 6 to 8 are only using the available ground truth depth maps to compute the difference between the results and the ground truth. Test videos 9 to 11 are subsequences of the Tsukuba stereo data set [56], [65]. Ground truth depth maps are available for all frames of the Tsukuba test videos. Contrary to [42], the method of [51] is used to compute forward and backward optical flow vectors, because the occlusion/disocclusion check requires flow vectors in both directions in order to determine the depth order. The same user-assigned scribbles as in [42] are used on the first and last input frame. Contrary to [42], no additional user-assigned scribble input is given on key frames between the first and the last frame.

Evaluation Strategy

In order to compute depth maps for the evaluation, some constant parameters are used for all test videos to achieve the results shown in this chapter: the Guided Filter parameters are set to $r = 11$, $r_t = 5$, $\varepsilon = 0.0016$, the colour matching threshold to avoid wrongly joined scribble pairs is set to $t_{colour} = 0.6$, and for all test videos resulting CVF segments are used to identify a frame-wide depth order. Furthermore, the spatial threshold of the spatial influence extension is set to the quarter of each test video's width $t_{spatial} = width/4$. All results showing the depth blending (DB) approach used the $n = 2$ lowest cost volume slices for the depth propagation, i.e. the $n = 2$ cost volumes slices with the lowest costs are averaged to determine the depth value.

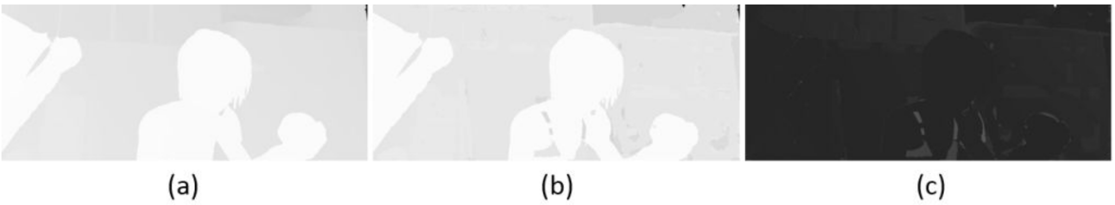


Figure 5.3: (a) Depth ground truth, (b) resulting depth map using our proposed algorithm, and (c) difference to depth ground truth. The lighter a pixel is, the higher is the difference at this pixel.

Additional parameters for the motion segmentation and the determination of the object size for the advanced depth change model (aM) of the temporal depth change extension (TDCE) are set individually per test video and can be seen in Table 5.1 and Table 5.2.

Our resulting depth maps are compared with depth ground truth or, if this is not available, reference depth maps computed by a stereo matcher [6]. As already mentioned, video scribbles for each test video are annotated on the first and last frame of the video. To enable a comparison of our depth propagation results with the depth ground truth, we propagate depth values at scribble positions instead of user-assigned depth values at scribble positions. Since our proposed approach is restricted to the assignment of a single depth value per scribble l, we compute the mean depth value of all depth values which the user-assigned scribble l covers in the depth ground truth:

$$d(l) = \frac{1}{n}\sum_{i=1}^{n} \hat{d}(i) \tag{5.1}$$

Here, n is the total number of pixels belonging to scribble l, i is one of the pixels belonging to scribble l, and $\hat{d}(i)$ is the depth value at pixel i in the ground truth depth map. Thus, each user-assigned scribble uses the mean value of the ground truth for the propagation.

Error Metric

We quantitatively compare the ground truth depth maps and our resulting depth maps by calculating the difference to the depth ground truth with the mean squared error e_{mse} [58]:

$$e_{mse} = \frac{1}{n}\sum_{i=1}^{n} (\hat{d}(i) - d(i))^2 \tag{5.2}$$

Here, n is the total number of pixels in the video, $d(i)$ is the depth value at pixel $i = (x, y, t)$ of the resulting depth map applying the proposed algorithm, and $\widehat{d(i)}$ is the depth value at pixel i in the ground truth depth map. In the following evaluation, for better readability the mean squared error e_{mse} is multiplied by 100. Moreover, e_{mse} is illustrated by grey scale error images. The lighter a pixel is, the higher is the error at this pixel, and thus, the difference to the ground truth at this position (cf. Figure 5.3). To better recognise the difference, the contrast and the brightness of the error images are increased by 30%.

5.2 Evaluation with Depth Ground Truth

In this section, the introduced extensions (cf. Chapter 4) are evaluated and their impact on results is discussed, i.e., the naive extension of [11] (NE) (cf. Section 4.1), the temporal consistency improvement (TCI) (cf. Section 4.2), the spatial influence extension (SIE) (cf. Section 4.3), and the temporal depth change extension (TDCE) (cf. Section 4.4) with the basic depth change model (bM) and the advanced depth change model (aM). Therefore, for each of the test videos shown in Table 5.1, depth is propagated by using one or a combination of the proposed extensions. Thus, in total 16 resulting depth maps of eight variants for both final depth propagation approaches are computed, i.e., eight resulting depth maps by using the winner-take-all (WTA) approach (cf. Table 5.3) and eight resulting depth maps by using the depth blending (DB) approach (cf. Table 5.4). In order to evaluate the impact of the optical flow in our proposed algorithm, we computed these 16 depth maps by using the provided optical flow ground truth from the MPI-Sintel data set (cf. Table 5.3 and Table 5.4), and additionally with an estimated optical flow (cf. Table 5.5 and Table 5.4). Below, we give an overview over the tested variants of our depth propagation algorithm. Their results are shown in Tables 5.3- 5.6, where bold printed values show the best resulting variant of each test video.

NE Results of the naive extension. No additional spatial influence term is added to the cost computation. Results are shown without the TCI (left column), and with the TCI (right column).

TCI The temporal consistency improvement is added to each of the extensions in order to additionally improve the results with the enhanced guided filter.

SIE Results of the spatial influence extension. An additional spatial influence term using a spatial threshold of $t_{spatial} = width/4$ is added to the cost computation. Results are shown without TCI (left column), and with the TCI (right column).

TDCE - bM Results of the temporal depth change extension using the basic depth change model. An additional spatial influence term using a spatial threshold of $t_{spatial} = width/4$ is added to the cost computation. Results are shown without TCI (left column), and with the TCI (right column).

TDCE - aM Results of the temporal depth change extension using the advanced depth change model. An additional spatial influence term using a spatial threshold of $t_{spatial} = width/4$ is added to the cost computation. Results are shown without TCI (left column), and with the TCI (right column). Note that, as mentioned before, if the start and end depth values contradict with the start and end object size values or there are not enough paths to determine the object's size, the basic depth change model is computed instead for this specific scribble pair with contradicting values.

Following, results of the variants shown in Tables 5.3- 5.6 are represented and discussed. As can be seen in Tables 5.3- 5.6, there is no variant which shows distinctly the best result, neither with the optical flow ground truth (cf. Table 5.3 and Table 5.4) nor with the estimated optical flow (cf. Table 5.5 and Table 5.6). This indicates that for both depth propagation approaches, i.e., the WTA and the DB approach, the impact of the extensions depends on the test video and its details.

	WTA - $e_{mse} \times 100$ (optical flow ground truth)							
	NE		**SIE**		**TDCE - bM**		**TDCE - aM**	
Video		**TCI**		**TCI**		**TCI**		**TCI**
alley1	0.05	0.05	0.03	0.03	**0.03**	0.03	0.03	0.03
ambush2	1.09	0.28	1.02	0.22	1.00	**0.20**	1.01	0.25
ambush5	0.99	1.00	0.43	0.44	**0.43**	0.43	0.43	0.43
ambush7	0.69	0.69	**0.20**	0.20	0.20	0.23	0.20	0.23
shaman2	0.67	0.70	0.26	0.25	0.26	**0.25**	0.26	0.25
shaman3	1.91	1.91	1.86	1.86	**0.37**	0.39	0.41	0.42
sleeping1	3.65	3.66	3.56	3.57	0.50	**0.49**	0.55	0.54
temple3	0.17	**0.12**	0.51	0.34	0.51	0.34	0.51	0.34

Table 5.3: Quantitative evaluation of the MPI-Sintel data set test videos. The difference $e_{mse} \times 100$ of resulting depth maps using optical flow ground truth and applying the WTA approach to the depth ground truth is computed. Variants of the introduced extensions are evaluated using the test videos shown in Table 5.1 from the MPI-Sintel data set [16]. Results of each extension are evaluated once without and once with the temporal consistency improvement (TCI), i.e., the naive extension (NE), the spatial influence extension (SIE), and the temporal depth change extension (TDCE) with the basic depth change model (bM) and the advanced depth change model (aM). The best variant of each test video is marked bold.

By comparing results using the optical flow ground truth of the WTA approach in Table 5.3 with results of the DB approach in Table 5.4, it can be seen that the WTA and the DB approach act similarly, i.e., the best variant of each test video is identical for the WTA and the DB approach. Moreover, a similar performance between the different variants can be observed, e.g., the error decreases for the WTA and the DB approach in the case of test video *ambush2* by additionally applying the TCI to the NE. Although the difference between the decreased error values differs, a similar behaviour by applying the different variants can be noticed. However, neither the WTA approach nor the DB approach achieves the best result for all test videos. In a following discussion, both approaches are compared with each other and the causes for achieving a better result than the other approach are explored.

Resulting depth maps using an estimated optical flow (cf. Table 5.5 and Table 5.6) behave similar to the above discussed results using optical flow ground truth. Moreover, comparing results using the optical flow ground truth with results using the estimated optical flow, it can be seen that the best variant of each test video is not always identical. This indicates that the optical flow has a significant impact on the resulting depth maps. The amount of the impact are part of the following discussions. Furthermore, it can be recognised that in the case of some videos and variants, e.g., all variants of test video *alley1*, results using the estimated optical flow are slightly better, i.e., the error decreases by >0.07 using the estimated optical flow. This behaviour

	DB - $e_{mse} \times 100$ (optical flow ground truth)							
	NE		**SIE**		**TDCE - bM**		**TDCE - aM**	
Video		**TCI**		**TCI**		**TCI**		**TCI**
alley1	0.07	0.07	0.03	0.02	**0.02**	0.02	0.02	0.02
ambush2	0.94	0.32	0.83	0.20	0.83	**0.19**	0.83	0.22
ambush5	0.88	0.88	0.58	0.58	**0.58**	0.58	0.58	0.58
ambush7	0.63	0.63	**0.20**	0.20	0.21	0.23	0.21	0.23
shaman2	0.93	0.95	0.41	0.40	0.41	**0.40**	0.41	0.40
shaman3	1.85	1.85	1.82	1.83	**0.29**	0.30	0.32	0.33
sleeping1	3.67	3.67	3.53	3.54	0.42	**0.42**	0.47	0.47
temple3	0.31	**0.20**	0.56	0.35	0.56	0.35	0.56	0.35

Table 5.4: Quantitative evaluation of the MPI-Sintel data set test videos. The difference $e_{mse} \times 100$ of resulting depth maps using optical flow ground truth and applying the DB approach to the depth ground truth is computed. Various variants of the introduced extensions are evaluated using test videos shown in Table 5.1 from the MPI-Sintel data set [16]. Results of each extension are evaluated once without and once with the temporal consistency improvement (TCI), i.e., the naive extension (NE), the spatial influence extension (SIE), and the temporal depth change extension (TDCE) with the basic depth change model (bM) and the advanced depth change model (aM). The best variant of each test video is marked bold.

	WTA - $e_{mse} \times 100$ (estimated optical flow)							
	NE		SIE		TDCE - bM		TDCE - aM	
Video		**TCI**		**TCI**		**TCI**		**TCI**
alley1	0.04	0.04	0.02	0.03	0.02	**0.02**	0.02	0.02
ambush2	1.05	**0.29**	1.05	0.68	1.04	0.67	1.06	0.78
ambush5	1.61	1.62	1.52	1.53	**1.05**	1.13	3.27	2.93
ambush7	0.71	0.72	**0.21**	0.21	0.24	0.23	0.23	0.23
shaman2	0.75	0.78	0.25	**0.24**	0.44	0.43	0.38	0.37
shaman3	1.91	1.91	1.87	1.87	**0.27**	0.31	0.37	0.41
sleeping1	3.71	3.72	3.61	3.62	**0.54**	0.54	0.65	0.65
temple 3	**0.49**	0.62	1.88	4.25	9.81	11.23	11.86	16.72

Table 5.5: Quantitative evaluation of the MPI-Sintel data set test videos. The difference $e_{mse} \times 100$ of resulting depth maps using an estimated optical flow and applying the WTA approach to the depth ground truth is computed. Variants of the introduced extensions are evaluated using the test videos shown in Table 5.1 from the MPI-Sintel data set [16]. Results of each extension are evaluated once without and once with the temporal consistency improvement (TCI), i.e., the naive extension (NE), the spatial influence extension (SIE), and the temporal depth change extension (TDCE) with the basic depth change model (bM) and the advanced depth change model (aM). The best variant of each test video is marked bold.

	DB - $e_{mse} \times 100$ (estimated optical flow)							
	NE		**SIE**		**TDCE - bM**		**TDCE - aM**	
Video		**TCI**		**TCI**		**TCI**		**TCI**
alley 1	0.05	0.05	0.02	0.02	0.02	**0.02**	0.02	0.02
ambush 2	0.92	**0.32**	0.86	0.64	0.86	0.65	0.87	0.74
ambush 5	1.89	1.88	1.54	1.63	**1.02**	1.13	2.70	2.69
ambush 7	0.63	0.63	**0.21**	0.21	0.23	0.23	0.23	0.23
shaman 2	0.86	0.87	0.38	**0.38**	0.50	0.49	0.46	0.46
shaman 3	1.89	1.89	1.83	1.83	**0.21**	0.25	0.30	0.34
sleeping 1	4.18	4.18	3.60	3.61	**0.48**	0.48	0.59	0.59
temple 3	**0.55**	0.56	1.75	3.97	7.94	10.13	10.25	14.75

Table 5.6: Quantitative evaluation of the MPI-Sintel data set test videos. The difference $e_{mse} \times 100$ of resulting depth maps using an estimated optical flow and applying the DB approach to the depth ground truth is computed. Various variants of the introduced extensions are evaluated using test videos shown in Table 5.1 from the MPI-Sintel data set [16]. Results of each extension are evaluated once without and once with the temporal consistency improvement (TCI), i.e., the naive extension (NE), the spatial influence extension (SIE), and the temporal depth change extension (TDCE) with the basic depth change model (bM) and the advanced depth change model (aM). The best variant of each test video is marked bold.

can be explained by rounding errors in the context of the long motion path generation in favour of the estimated optical flow. Since optical flow fields are floating point numbers, the new pixel position of the long motion path is determined by a rounding operation. Thus, it might happen that long motion paths using the estimated optical flow are built in a way that the error slightly decreases compared to results using the ground truth optical flow. In the following, first the impact of each extension is discussed separately regarding the optical flow ground truth and the estimated optical flow. Since the behaviour of both depth propagation approaches, i.e., the WTA and the DB approach, is similar, discussed issues are valid for both approaches unless otherwise noted. Subsequently, results of the two depth propagation approaches, i.e., the WTA and the DB approach, are compared with each other.

Temporal Consistency Improvement (TCI)

Examining results using the optical flow ground truth in Table 5.3 and Table 5.4, it can be seen that resulting depth maps either significantly improve, i.e., the improvement is > 0.1, or that there is no or only a slight improvement, i.e., the improvement is < 0.03. Exploring test videos with a significant improvement, i.e., test videos *ambush2* and *temple3*, one can recognise that both test videos contain large object movements. That is, objects are moving fast in front of other objects or the background. Thus, through the cost volume filtering process without the TCI, costs of different objects are filtered and costs are wrongly propagated. Finally, a wrong depth value is propagated. Figure 5.4 shows an example of the discussed issue. Figure 5.4 presents the results

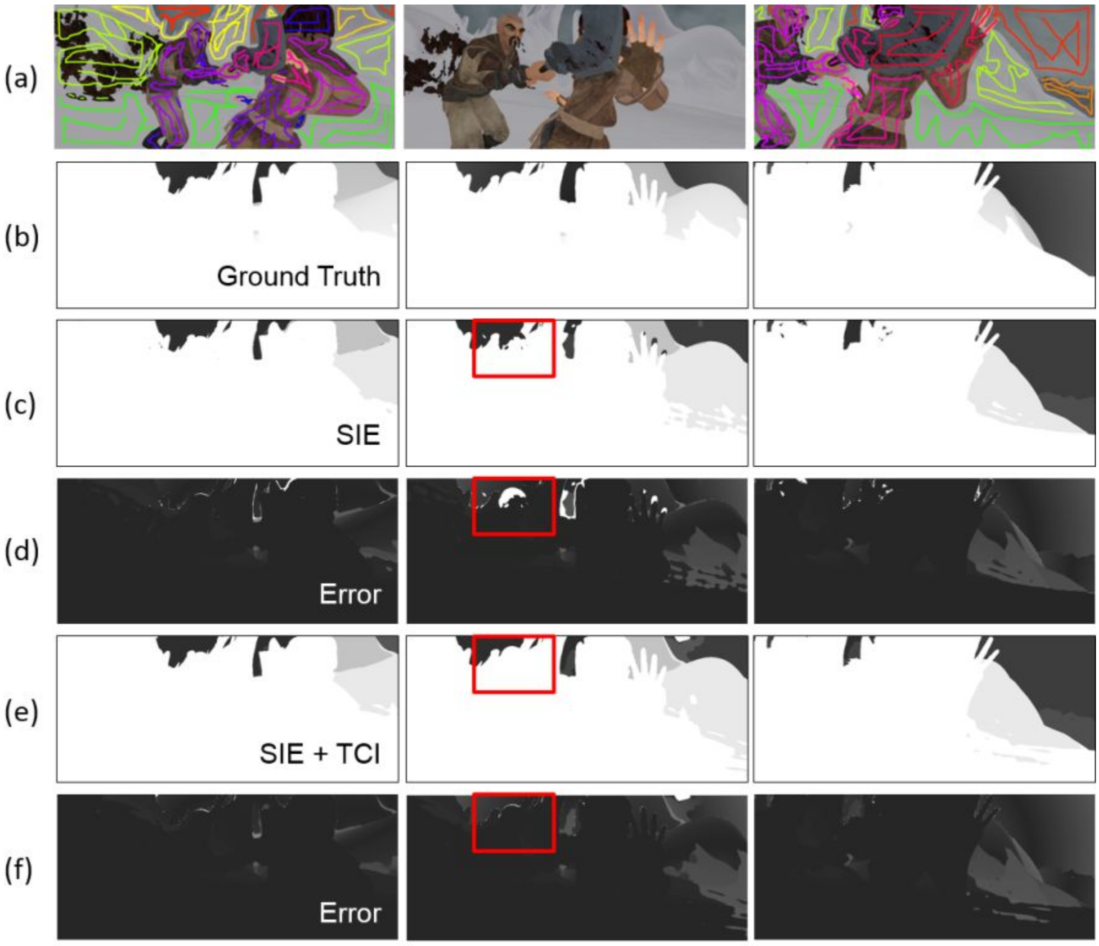


Figure 5.4: Comparison of results of test video *ambush2* with and without the TCI by using the WTA approach for the final depth propagation. (a) shows the input frames 1, 3 and 5 of the test video with user-assigned scribbles on the first and last frame. (b) shows the ground truth depth maps. (c) shows results of the SIE and a threshold of $t_{spatial} = 256$. (d) shows the corresponding error image of (c) compared to the ground truth ($e_{mse} \times 100 = 1.02$). (e) shows results of the SIE with a threshold of $t_{spatial} = 256$ combined with the TCI. (f) shows the corresponding error image of (e) compared to the ground truth ($e_{mse} \times 100 = 0.22$). The highlighted area shows the improved assignment of depth caused by the TCI.

of test video *ambush2* without and with the TCI, i.e., without and with the enhanced guided filtering of the cost volumes, using the SIE in order to eliminate wrong pixels. The wrong depth propagation caused by a filtering over different objects can be seen in Figure 5.4(c) and (d). The head of the bold person is moving fast in front of the background. Thus, a part of the head is wrongly assigned to the background depth value.

Contrary, looking at results using the estimated optical flow in Table 5.5 and Table 5.6, it can be seen that the impact of the TCI is similar with the exception of test videos *ambush5* and *temple3*. In both cases the TCI degrades the resulting depth maps. Since the test videos contain large object movements, the method of [51] fails in estimating robust optical flow fields. Due to the inaccurate optical flow, long motion paths are built containing different objects and filtering along the long motion paths increases the error. An example is shown in Figure 5.5 where the optical flow ground truth is compared with the estimated optical flow by [51]. As can be seen, the estimated optical flow fields are oversmoothed at borders (cf. Figure 5.5(c)) which results in long motion paths containing different objects. Thus, a robust optical flow estimation which achieves robust results for videos with large movements is necessary.

Additionally applying the TCI when using a robust optical flow estimation, the results increase significantly for videos with large object movement as can be seen in the case of test video *ambush2* (cf. Figure 5.4(e) and (f)). However, if there are no large object movements in the video, the additional computational effort of the TCI can be hold back.

Spatial Influence Extension (SIE)

As can be seen in Tables 5.3- 5.6, the results improve, i.e., the error e_{mse} decreases, for almost all test videos (cf. Table 5.1) by applying the SIE with a threshold of $t_{spatial} = 256$. Test video *temple3* is the only exception. This applies to results using the optical flow ground truth (cf. Table 5.3 and Table 5.4) and to results using the estimated optical flow (cf. Table 5.5 and Table 5.6). Figure 5.6 and Figure 5.7 show results of two test videos when applying the NE and when applying the SIE, i.e., without and with reducing the influence of wrong pixels by adding spatial costs to the cost computation, respectively. In the case of test video *temple3*, the SIE degrades the results as can be seen in Figure 5.6(e)-(f). The increasing error results from the wrong assignment of a background region to a foreground depth. This error is caused by the occlusions of the background throughout the video. When tracking the scribbles from frame to frame these occlusions causing the loss of scribble points of user-assigned background scribble points (red scribble in Figure 5.6(a)) and the SIE increases costs in this region. By annotating additional scribbles covering the background on the last frame, the error could be decreased. However, the user effort would increase as well.

Figure 5.7 gives an example of a test video where the error decreases by 0.04 when applying the SIE, i.e., the results improve. Since the shown test video *alley1* contains regions with similar colours (e.g., the background and the person's tank top) but different depth positions, the SIE reduces the impact of wrongly assigned pixels, and thus, improves the results. For example, it can be seen that the background noise in Figure 5.7(c) is reduced by applying the SIE.

The impact of the optical flow in this extension is small. Since the SIE uses the optical flow vector fields only to track scribble points throughout the video, no large error occurs. Moreover, since every test video presented in Table 5.1, except test video *temple3*, contains various regions

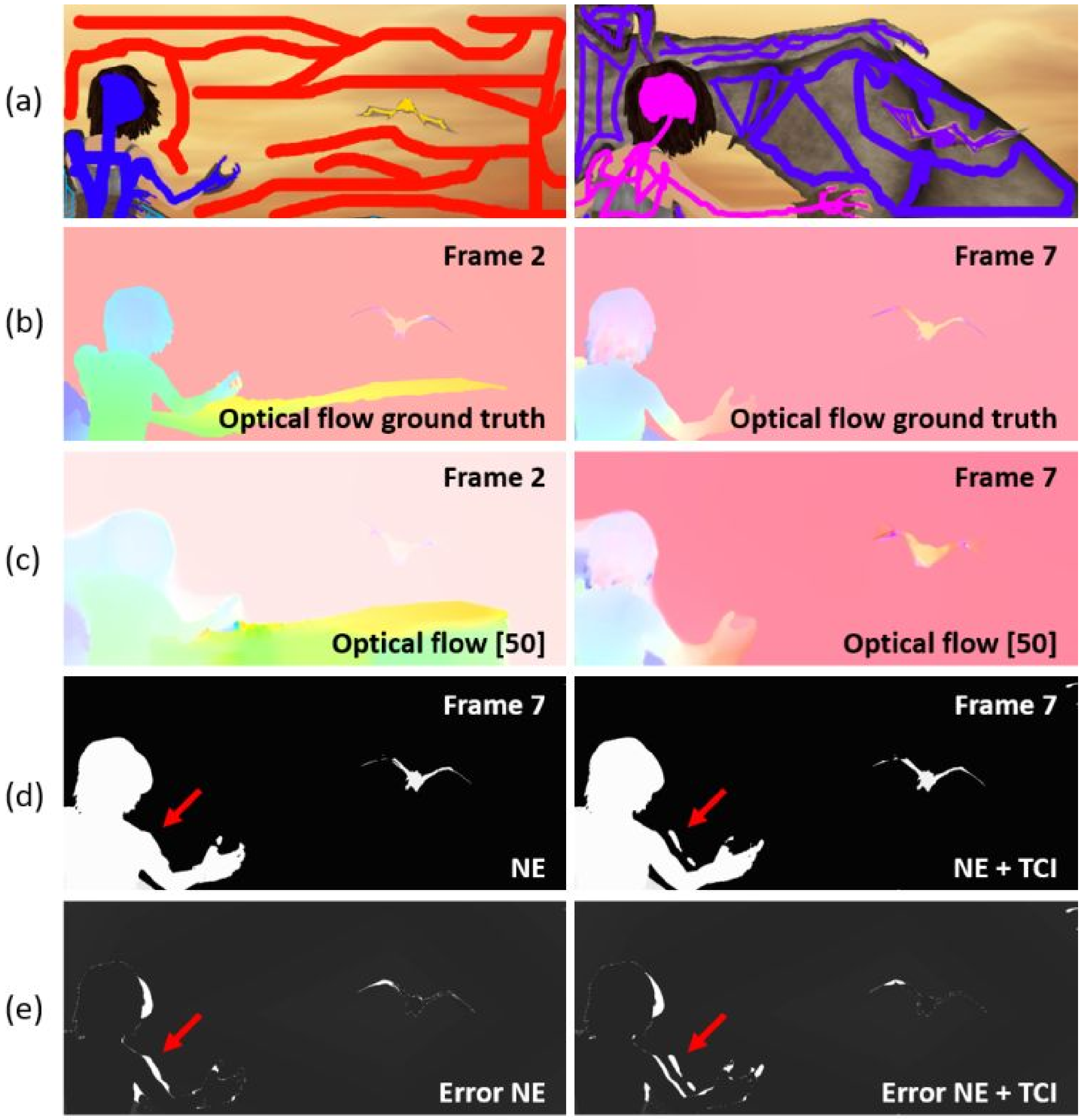


Figure 5.5: Comparing results of test video *temple3* using the estimated optical flow when applying the NE and when additionally applying the TCI. (a) shows the first and last input frame with user-assigned scribbles. (b) shows the optical flow ground truth of frames 2 and 7. (c) shows the estimated optical flow by [51] of frames 2 and 7. (d) shows the resulting depth map of frame 7 when applying the NE (left) and when additionally applying the TCI (right). (e) shows the corresponding error images when applying the NE (left) and when additionally applying the TCI (right).

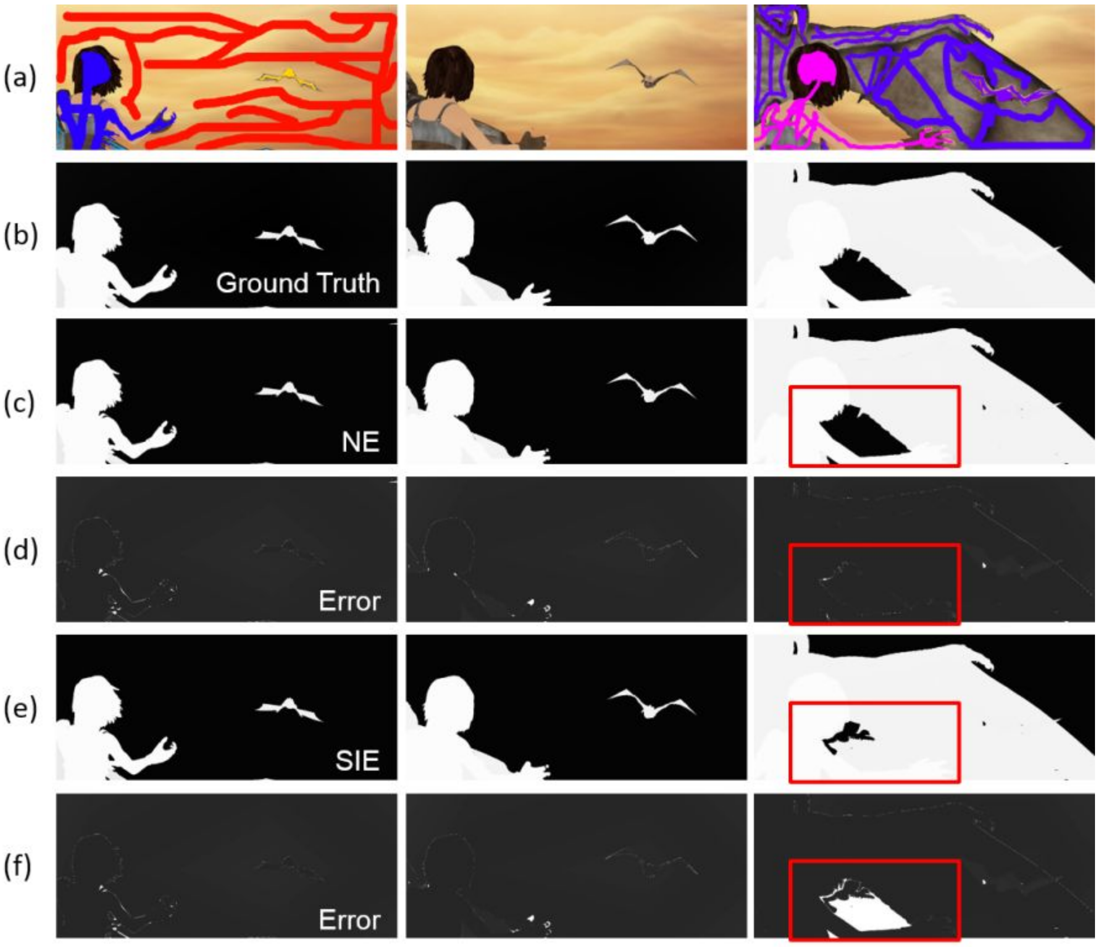


Figure 5.6: Comparison of results of test video *temple3* with and without the SIE by using the WTA approach for the final depth propagation. (a) shows the input frames 5, 14 and 22 of the test video with user-assigned scribbles on the first and last frame. (b) shows the ground truth depth maps. (c) shows results of the NE without the SIE. (d) shows the corresponding error images of (c) compared to the ground truth ($e_{mse} \times 100 = 0.17$). (e) shows results of the SIE and a threshold of $t_{spatial} = 256$. (f) shows the corresponding error images of (e) compared to the ground truth ($e_{mse} \times 100 = 0.51$). The highlighted scope shows the wrong assignment of a region belonging to the background caused by the SIE.

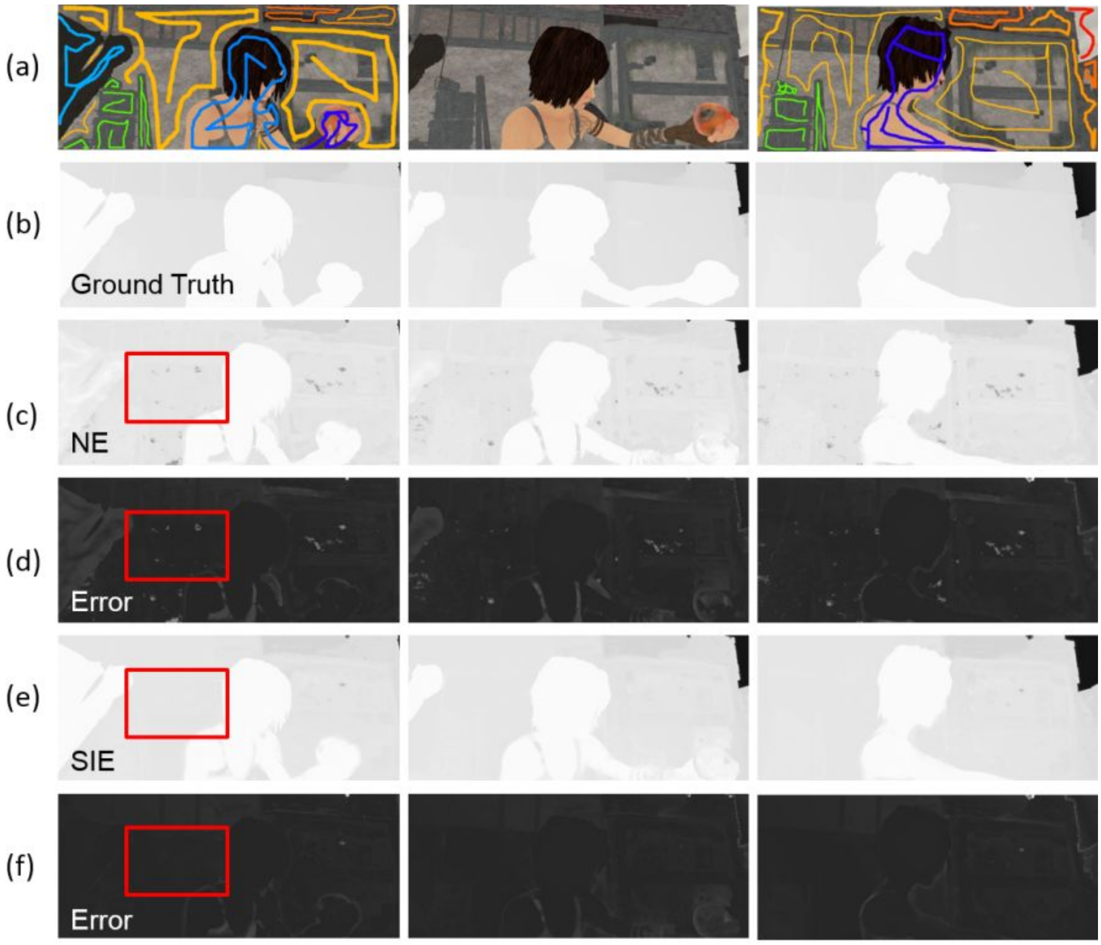


Figure 5.7: Comparison of results of test video *alley1* with and without the SIE by using the DB approach for the final depth propagation. (a) shows the input frames 1, 25 and 50 of the test video with user-assigned scribbles on the first and last frame. (b) shows the ground truth depth maps. (c) shows results of the NE without the SIE. (d) shows the corresponding error images of (c) compared to the ground truth ($e_{mse} \times 100 = 0.07$). (e) shows results of the SIE and a threshold of $t_{spatial} = 256$. (f) shows the corresponding error images of (e) compared to the ground truth ($e_{mse} \times 100 = 0.03$). The highlighted scope shows the reduction of errors caused by the SIE.

with similar colours but different positions in depth, the SIE improves the results by reducing the impact of wrong pixels as described in Section 4.3. Therefore, if a video contains objects unique in their colour and depth position, the additional computation of spatial costs is not necessary and could degrade the results (e.g., in the case of test video *temple3*).

Temporal Depth Change Extension (TDCE)

Examining results using the optical flow ground truth in Table 5.3 and Table 5.4 shows that only in the case of two test videos, *shaman3* and *sleeping1*, a significant improvement is achieved when applying the TDCE. A characteristic of both test videos, *shaman3* and *sleeping1*, is a large camera zoom-in. The zoom causes all objects in the video coming closer to the camera which changes their depth distinctly and visibly. Regarding the other test videos, no significant achievement is recognisable in the results. Although some of them contain objects moving closer to the camera, they contain smaller movements in depth than test videos *shaman3* and *sleeping1*. Figure 5.8 shows results of test video *sleeping1* without and with the TDCE, i.e., without and with enabling depth changes. As can be seen in Figure 5.8, the error for both variants, i.e., without the TDCE (cf. Figure 5.8(c) and with the TDCE (cf. Figure 5.8(f)), in the first frame is identical, since in both cases the user-assigned depth values are propagated to the first frame. However, in Figure 5.8(c) and (d) it can be seen that those depth values do not change throughout the video, since no depth change is performed. Thus, the error increases with the number of frames. Contrary to that, Figure 5.8(e) and (f) show results of the performed depth change by the TDCE.

Looking at the results using the estimated optical flow in Table 5.5 and Table 5.6, the same behaviour of test videos *shaman3* and *sleeping1* can be observed, i.e., a significant improvement is achieved when applying the TDCE. However, in the case of two test videos, i.e., *shaman2* and *temple3*, the error increases by applying the TDCE. In the case of test video *temple3* (cf. Figure 5.9) the large error can be explained by scribble matching and depth ordering issues. As already mentioned above, the method of [51] has problems in estimating optical flow fields in videos with large object movements. Due to that, the optical flow fields are oversmoothed at borders and wrong depth order relations are detected which further result in a wrong interpolation of depth values (cf. arrows nr. 1 and nr. 2 in Figure 5.9). Moreover, the performed motion segmentation in order to identify matching scribble pairs is inaccurate and scribbles from the first frame are matched with a wrong scribble from the last frame (cf. Figure 5.9(b) and arrow nr. 3 in Figure 5.9). Further, wrong depth change model values are computed according to wrong matched scribble pairs which leads to such a large error when applying the TDCE, i.e., $e_{mse} > 7.9$. In the case of test video *shaman2* the increasing error can be explained by a depth ordering issue due to the above mentioned problems with the estimated optical flow.

Therefore, the TDCE can significantly improve the results of videos with large depth changes. However, in order to correctly identify matching scribbles, an optical flow estimation algorithm which can deal with large movements is needed. Additionally, the estimated optical flow should contain sharp borders in order to detect correct depth order relations and compute the corresponding depth change model. Generally, the additional effort of the TDCE can be hold back for videos without large depth changes.

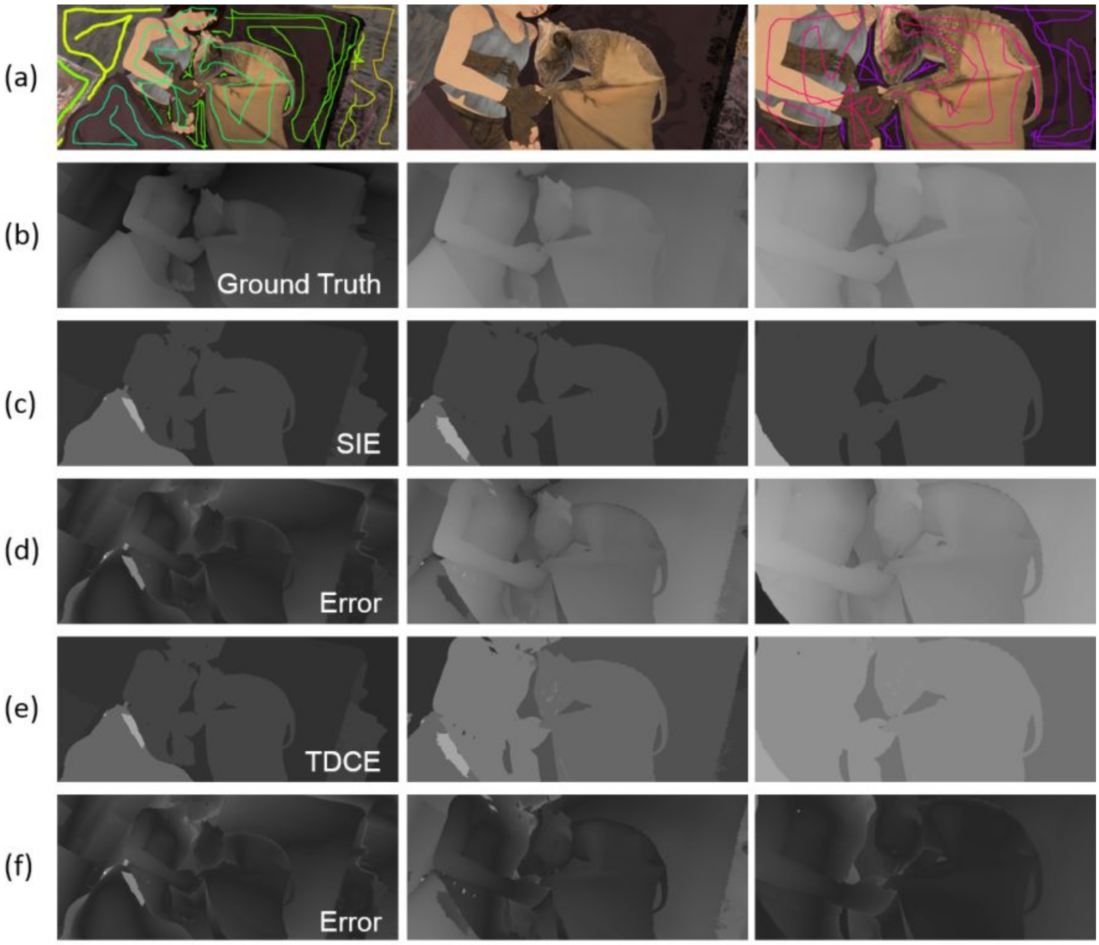


Figure 5.8: Comparison of results of test video *sleeping1* with and without the TDCE by using the WTA approach for the final depth propagation. (a) shows the input frames 1, 25 and 48 of the test video with user-assigned scribbles on the first and last frame. (b) shows the ground truth depth maps. (c) shows results of the SIE and a threshold of $t_{spatial} = 256$. (d) shows the corresponding error images of (c) compared to the ground truth ($e_{mse} \times 100 = 3.56$). (e) shows results of the SIE with a threshold of $t_{spatial} = 256$ combined with the TDCE using the basic depth change model. (f) shows the corresponding error images of (e) compared to the ground truth ($e_{mse} \times 100 = 0.50$).

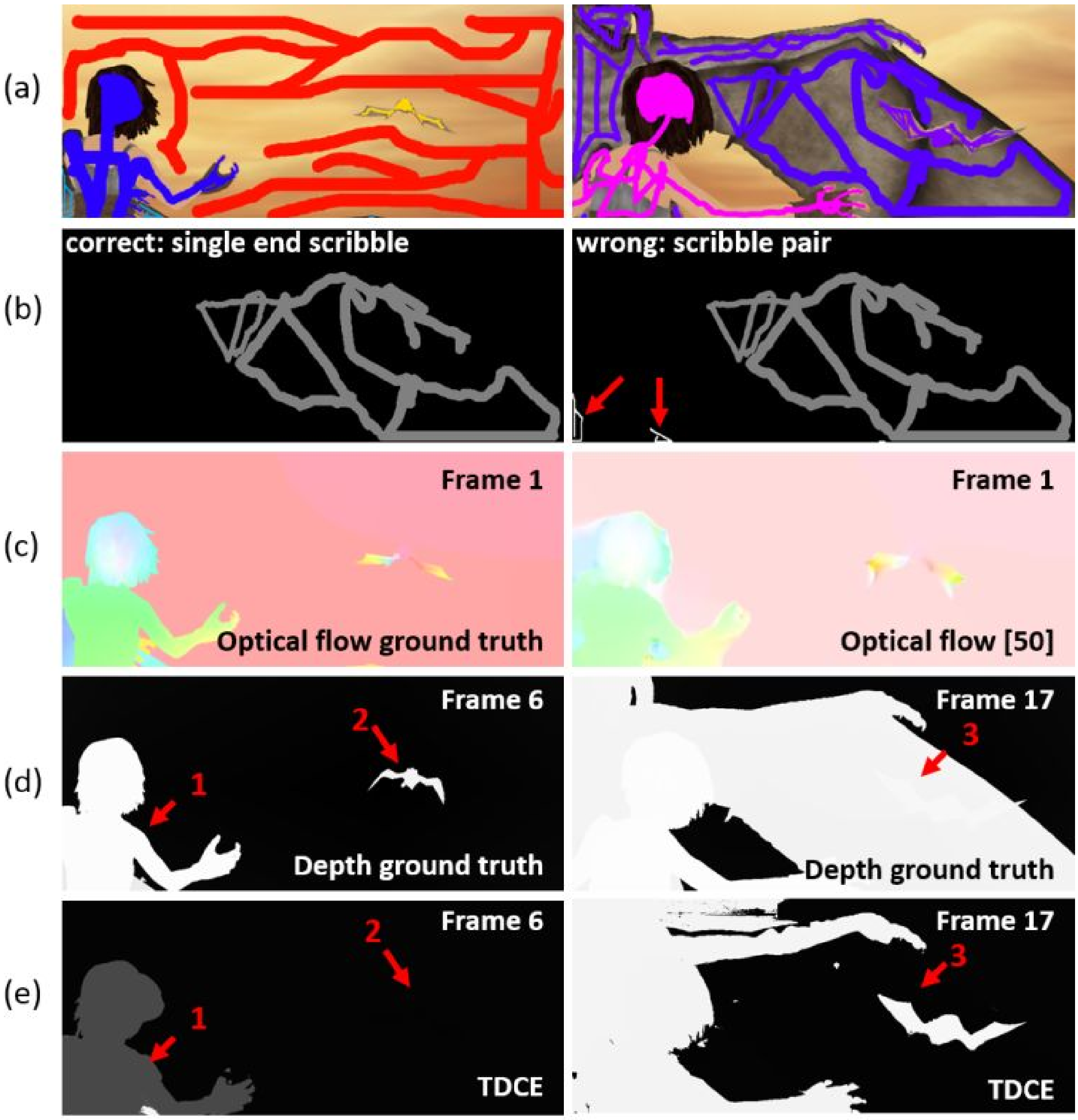


Figure 5.9: Comparing results of test video *temple3* using the estimated optical flow when using optical flow ground truth and when using an estimated optical flow by [51]. (a) shows the first and last frame with user-assigned scribbles. (b) shows the correct identification of a single end scribble (left), i.e., the big wing of the dragon in the last frame, and the wrong match of a scribble pair containing two scribbles from the first frame (white scribbles) and a scribble from the last frame (grey scribble). (c) shows the optical flow ground truth of frame 1 (left) and the estimated optical flow by [51] (right). (d) shows the depth ground truth of frames 6 and 17. (e) shows the resulting depth maps applying the TDCE of frames 6 and 17. Arrows nr. 1 and nr. 2 show wrong propagated depth values due to a depth ordering issue. Arrow nr. 3 shows wrong propagated depth value due to a scribble matching issue, i.e., shown in (b).

Basic (bM) vs. Advanced (aM) Depth Change Model

In the case of results using the optical flow ground truth there is no significant difference between results using the basic depth change model and results using the advanced depth change model as can be seen in Table 5.3 and Table 5.4. Since in all test videos (cf. Table 5.1), objects are moving in depth continuously, e.g., in the case of camera zoom-ins in test videos *shaman3* and *sleeping1*, the advantages of the aM does not appear, i.e., the recognition when an objects stops moving in depth. Moreover, it happened that the start and end depth value contradicted with the start and end object size, or that not enough paths go throughout the whole video to determine the object's size. As mentioned above, in such cases the basic depth change model was computed for the scribble pair. This instance occurred for test videos 1-4 (cf. Table 5.1). The aM depends on a valid determination of the object size in each frame.

Contrary, examining the results using the estimated optical flow shown in Table 5.5 and Table 5.6, some significant differences can be seen. In the case of three test videos, i.e., *ambush5*, *sleeping1* and *temple3*, the aM degrades the results significantly, i.e., the difference is > 0.1. This can be explained by wrongly identified scribble pairs for reasons already discussed above, i.e., inaccurate motion segmentation due to the optical flow estimation. Since all of the three test videos are using paths to identify the object's size per frame, wrongly matched scribbles can influence the depth change and lead to wrong depth propagation.

Therefore, the extensive aM can be hold back, when objects in the video are changing without any interruptions their position in depth. In such cases, the basic depth change model achieves good and perceptually consistent results.

Comparing our approaches: Winner-take-all and Depth Blending

When examining results in Tables 5.3-5.6, it can be seen that neither the WTA approach nor the DB approach achieves the best results for all test videos. However, comparing results using optical flow ground truth (cf. Table 5.3 and Table 5.4) with results using an estimated optical flow (cf. Table 5.5 and Table 5.6), it can be seen that for almost each test video the same approach achieves the best results, e.g., in the case of test video *shaman2* the WTA approach achieves the best results using the optical flow ground truth and using an estimated optical flow. This indicates, that the impact of the two approaches depends on the test video and its details.

Thus, analysing test videos where the WTA approach significantly (i.e., difference between WTA and DB approach > 0.06) achieves the best results, i.e., test videos *ambush5*, *shaman2* and *temple3*, it can be seen that all these test videos contain little depth variations inside objects (cf. Figure 5.10 (a)) compared to test videos where the DB approach clearly achieves the best results, i.e., test videos *shaman3* and *sleeping1* (cf. Figure 5.10 (b)). Moreover, objects are far from each other regarding their position in depth in the case of test videos where the WTA approach wins.

However, in the case of test video *ambush5*, the WTA approach achieves better results when using the ground truth optical flow, but the DB approach when using the estimated optical flow. Test video *ambush5* contains large movements but also little depth variations inside objects. Moreover, the objects are close to each other regarding their position in depth. As mentioned above, the estimated optical flow fields computed by [51] cannot handle large movements and

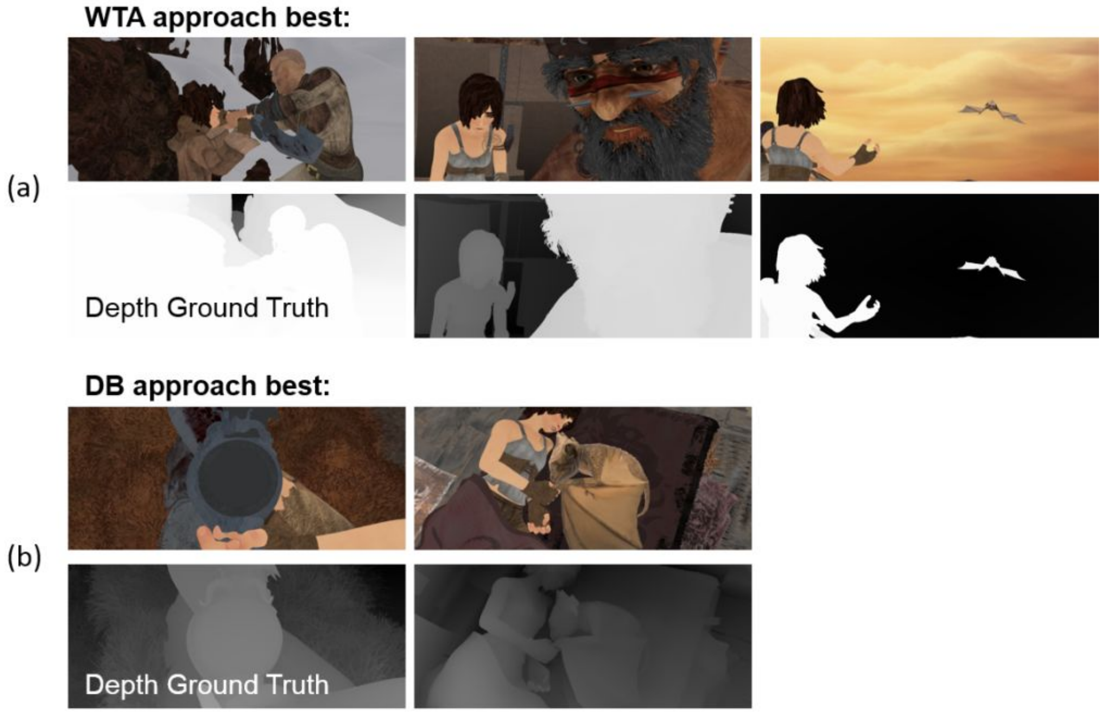


Figure 5.10: Comparing our approaches: Winner-take-all and Depth Blending. In the first row of (a) and (b) the first frame of each test video is shown. In the second row of (a) and (b) the corresponding depth ground truth is shown. (a) shows test videos *ambush5*, *shaman2* and *temple3* (from left to right), where the WTA approach achieves the best results. (b) shows test videos *shaman3* and *sleeping1* (from left to right), where the DB approach achieves the best results.

depth is wrongly propagated. Thus, the DB approach improves the result of wrongly propagated depth values by computing a weighted mean and smoothing the result.

Figure 5.11 shows results of test video *shaman2*, where the WTA approach achieves better results. As can be seen in the highlighted area, the DB approach wrongly increases the depth values of the background object. Since the foreground object which occludes the background object is similar in colour to the background, pixels close to the border of both objects have low costs assigned in two cost volume slices, e.g., the cost volume corresponding to the background scribble (orange in Figure 5.11(a)) and the cost volume corresponding to the foreground object (light blue in Figure 5.11(a)). Due to the computation of the weighted mean of those two cost volume slices, wrong depth values are assigned. Since there is a huge difference in depth between the two objects and DB increases the depth values of the back object, the resulting depth value degrades the result.

Contrary, Figure 5.12 shows results of test video *sleeping1*, where the DB approach achieves better results. As can be seen in the highlighted area, the DB approach corrects a misassignment of a region, where two cost volume slices achieve low costs. Due to the computation

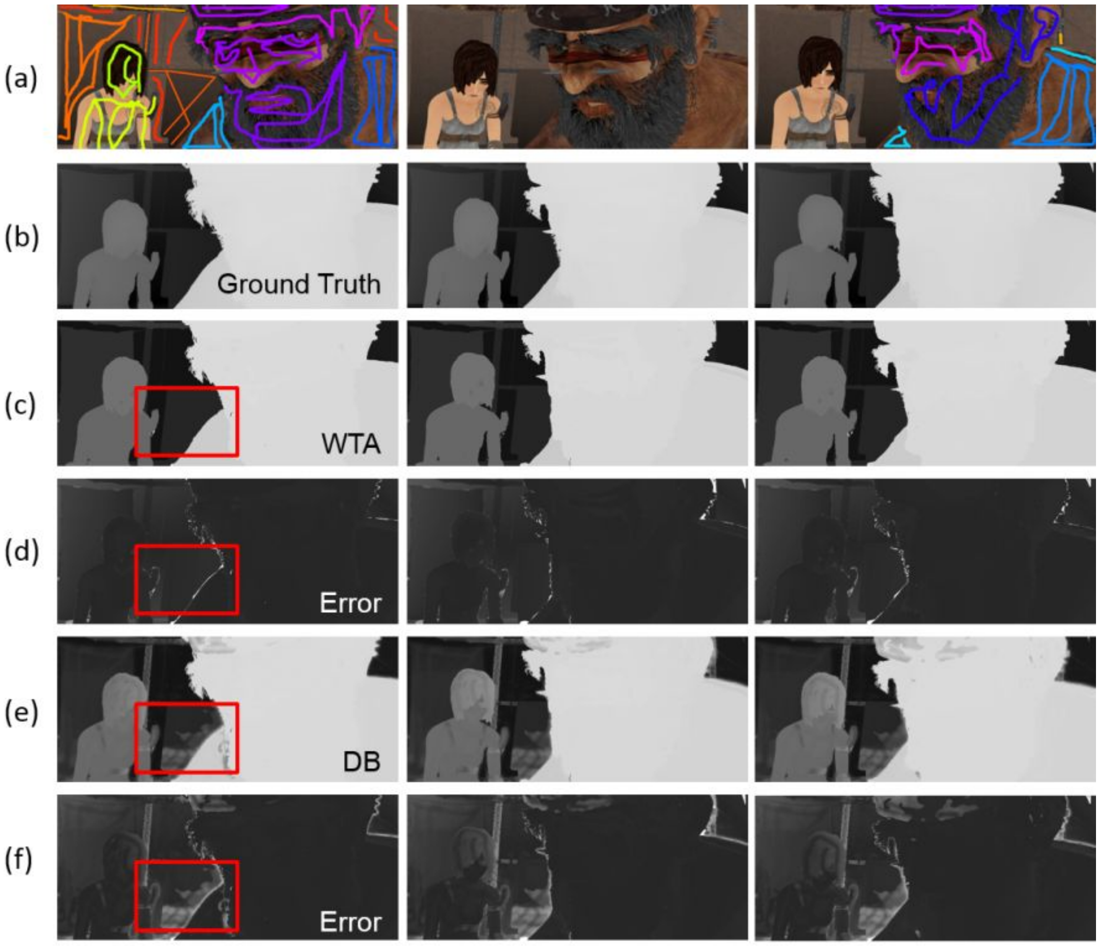


Figure 5.11: Comparing the two depth propagation approaches: winner-take-all and depth blending. Results of test video *shaman2*. The TDCE using the basic depth change model and a threshold of $t_{spatial} = 256$ combined with the TCI is applied in order to show the difference between the WTA and the DB approach. (a) shows the input frames 1, 25 and 50 of the test video with user-assigned scribbles on the first and last frame. (b) shows the corresponding ground truth depth maps. (c) shows results using the WTA approach. (d) shows the corresponding error images with an error of $e_{mse} \times 100 = 0.25$. (e) shows results using the DB approach. (f) shows the corresponding error images with an error of $e_{mse} \times 100 = 0.40$. The highlighted area shows the region which is degraded in the result caused by the DB approach.

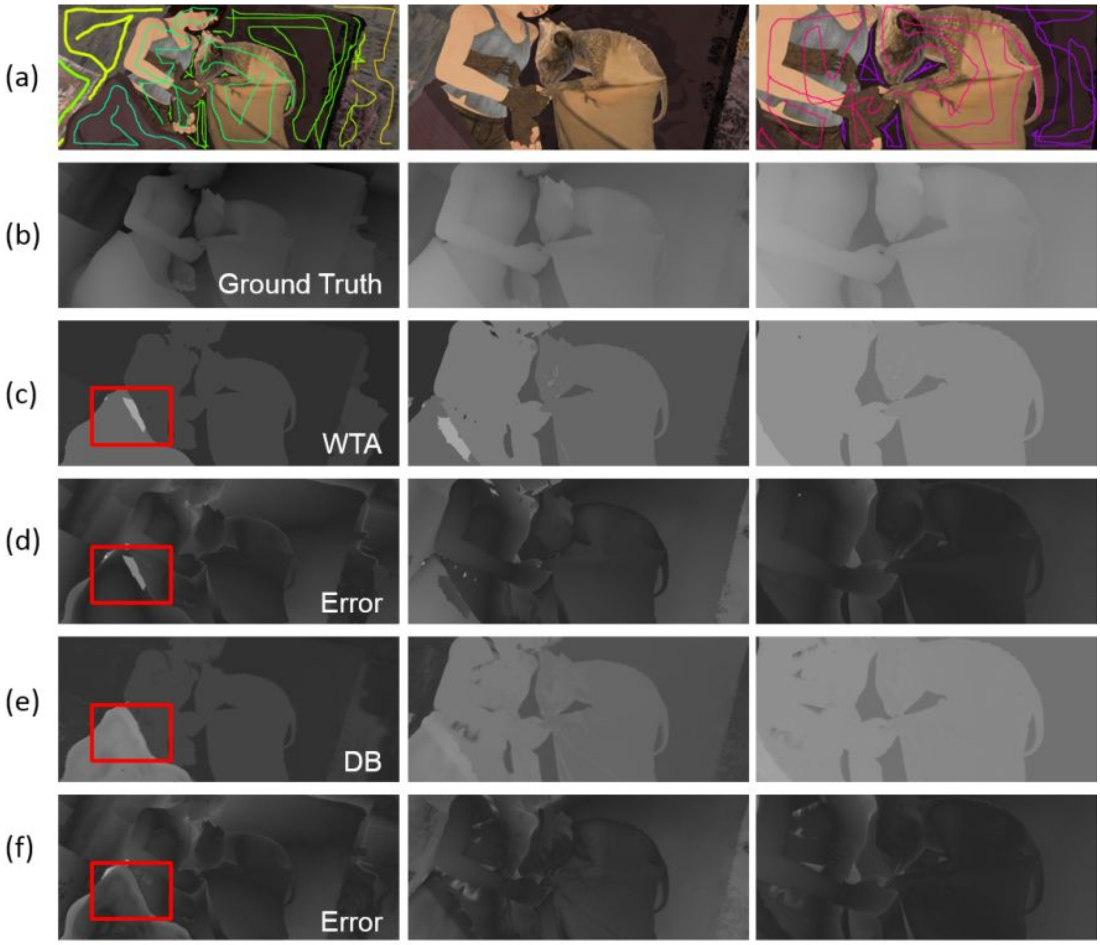


Figure 5.12: Comparing the two depth propagation approaches: winner-take-all and depth blending. Results of test video *sleeping1*. The TDCE using the basic depth change model and a threshold of $t_{spatial} = 256$ is applied in order to show the difference between the WTA and the DB approach. (a) shows the input frames 1, 25 and 48 of the test video with user-assigned scribbles on the first and last frame. (b) shows the corresponding ground truth depth maps. (c) shows results using the WTA approach. (d) shows the corresponding error images with an error of $e_{mse} \times 100 = 0.49$. (e) shows results using the DB approach. (f) shows the corresponding error images with an error of $e_{mse} \times 100 = 0.42$. The highlighted scope shows the region which is smoothed by the DB approach.

of a weighted mean of the two cost volume slices with the lowest costs, this misassignment is smoothed and reduces the error. Moreover, the DB approach can improve the result of regions containing slanted or rounded surfaces. In such cases, depth values are smoothed between the change of two segments of the slanted or rounded surface.

Thus, the WTA approach achieves better results in videos with a large depth range and little depth variations inside objects. Contrary, DB improves the result within an object by smoothing various depth values (cf. Figure 5.12) and thus, corrects misassignments between two scribbles with low costs by smoothing. But the DB approach can also degrade the result, especially at borders of two objects with similar colour but a huge difference in depth (cf. Figure 5.11).

Conclusions

Summing up, each of the extensions achieves an improvement depending on the content of the video: The TCI achieves significant improvements for videos with large object movements by filtering along the object movements if a robust optical flow estimation is used and long motion paths contain only the movement of one object. The SIE improves results for videos with objects similar in colour but different positions in depth by reducing the influence of wrong pixels. The accuracy of the optical flow estimation has less impact when applying the SIE. Finally, the TDCE improves results for videos with a significant depth change of either the whole scene of the video (e.g., camera zooms), or of objects moving in depth by establishing depth change models.

Note that results depend on a robust optical flow estimation as well as on a robust scribble annotation by the user. We notice limitations concerning the scribble placement. Since the user effort is kept low, i.e., only two keyframes with user-assigned scribbles are necessary, the user has to keep in mind colour similarities between objects, the depth position of each object and the movement throughout the video. In order to achieve a valid segmentation, the scribble should only cover pixels belonging to the current object. Each covered pixel belonging to the background or another object degrades the segmentation. Moreover, if there are objects similar in colour but different in depth and the SIE is used, the user should be aware of the chosen spatial threshold when annotating scribbles on the different objects. Finally, when the depth change is enabled by the TDCE, the motion in the video should be considered in order to correctly identify matching scribble pairs. Since the annotated scribbles for all test videos in this evaluation stayed the same for all variants, an improvement of the results could be achieved by adapting scribbles according to the used variant.

5.3 Comparison with a Similar Algorithm

In this section, test videos shown in Table 5.2 are compared with results of the algorithm of [42]. The result of the best variant of both depth propagation approaches, i.e., the WTA and the DB, are compared with resulting depth maps of [42]. Therefore, all test videos shown in Table 5.2 were evaluated concerning the computing results for the WTA and the DB approach of the same eight variants presented in Section 5.2. For the comparison with the algorithm of [42], the best variant of each video for the WTA and the DB approach (cf. Table 5.7) was chosen.

The comparison is performed on 11 test videos (cf. Table 5.2) and compared to their respective ground truth. To enable a fair comparison, the results of both algorithms are based on the same user-input, i.e., scribbles and depth values. Specifically, for each user-annotated scribble both algorithms are assigned the average depth value from the corresponding covered pixels in the depth ground truth or reference depth map. In this context, it should be noted that [42] supports multiple depths per scribble. However, in order to have a fair comparison, [42] uses the same input as our proposed algorithm, i.e., a single depth per scribble. Thus, both algorithms start the depth propagation process with the same assigned depth values. The scribble annotations stayed the same as in [42] with annotations on the first and last frame.

	our best variant		$e_{mse} \times 100$		
Video	**WTA**	**DB**	**WTA**	**DB**	**[42]**
City	TDCE-aM + TCI	SIE	1.01	0.82	**0.47**
Parade	SIE	SIE	0.72	0.56	**0.28**
Palace	TDCE-aM + TCI	SIE	1.16	**0.98**	1.20
Staircase	SIE + TCI	SIE + TCI	0.69	0.62	**0.51**
Soccer	SIE	SIE	0.44	**0.29**	0.40
Child	SIE	TDCE-aM + TCI	0.57	**0.54**	0.58
Head	TDCE-bM + TCI	TDCE-bM + TCI	0.49	**0.37**	0.65
Interview	SIE + TCI	SIE + TCI	0.72	0.69	**0.56**
Tsukuba50-66	TDCE-bM + TCI	TDCE-bM + TCI	0.20	0.17	**0.15**
Tsukuba380-397	TDCE-aM + TCI	SIE	0.34	0.38	**0.21**
Tsukuba1-100	TDCE-bM + TCI	TDCE-bM + TCI	0.08	**0.05**	0.15

Table 5.7: Comparison with a similar segmentation-based depth propagation algorithm from [42]. The table shows the variant (cf. Section 5.2) achieving the best results regarding the WTA and the DB approach and the mean squared error $e_{mse} \times 100$ of the best result of the WTA approach and the DB approach. Those are compared with the results achieved by the algorithm of [42]. Bold printed values represent the best result and approach of each test video.

As can be seen in Table 5.7 by comparing results of the WTA and the DB approach, in the case of this comparison data set (cf. Table 5.2), DB always achieves better results. As mentioned above, due to the weighted mean computation, DB smooths mixed assignments of different depth values and thus, smooths the result which can reduce the error. Additionally, we observed as can be seen in Table 5.7 that the best resulting variants (cf. Section 5.2) applied the SIE with a threshold of $t_{spatial} = width/4$. Moreover, unlike results in Section 5.2, the best variant of the WTA and the DB approach are not always the same. Contrary to test videos in the previous Section 5.2, the impact of the optical flow in the context of the comparison data set (cf. Table 5.2) is not evaluated additionally, i.e., only the estimated optical flow is used to compute depth maps. Due to the used estimated optical flow fields, the motion segmentation is not as accurate as with ground truth data, which makes the identification of scribble pairs more difficult as mentioned above. Therefore, note that not in all test videos all scribble pairs were identified correctly. However, as mentioned above in Section 5.2, if there is no large depth change in the video, a missing match of a scribble pair has no big effect on the results.

Moreover, results in Table 5.7 show that for almost all test videos the best approach is discernable, i.e., the difference between the best and second best result is > 0.10. However, neither our DB approach nor the algorithm of [42] achieves distinctly the best results. The causes are discussed in the following. Figures 5.13 to 5.16 show results of final depth maps of our WTA and DB approaches and of the algorithm from [42].

Figure 5.13 shows results of the test video *Palace*, where our DB approach achieves the best results ($e_{mse} \times 100 = 0.98$). This test video contains two main challenges: Colour similarity of a foreground object with the background (arrow nr. 1 in Figure 5.13), and the fine structured borders of the tree (arrow nr. 2 in Figure 5.13). Analysing the colour ambiguity in case one, the advantages of the SIE can be seen (arrow nr. 1 in Figure 5.13). [42] propagates too many pixels of the background with the foreground object's depth value (cf. Figure 5.13(c) arrow nr. 1). Thus, the actual foreground object is hard to identify in front of the background. By applying the SIE with additionally enabling depth changes by the TDCE, the WTA approach achieves that the background is definitely separated from the foreground object (cf. Figure 5.13(g) arrow nr. 1). However, the DB approach increases again the depth value of the background due to the weighted mean approach. But the foreground object is still recognisable as a separate object. Moreover, DB smooths regions at the fine structured borders of the tree (arrow nr. 2 in Figure 5.13) and, thus, reduces the error from $e_{mse} \times 100 = 1.16$ (WTA) to $e_{mse} \times 100 = 0.98$ (DB), which is the best result achieved by the three methods, i.e., our approach using WTA, our approach using DB and the algorithm of [42].

Figure 5.14 shows results of the test video *Parade*, where the algorithm of [42] achieves the best results ($e_{mse} \times 100 = 0.28$). However, as can be seen in Figure 5.14, all three approaches struggle especially with the foreground object (cf. arrow nr.1 in Figure 5.14). Since the foreground object contains many different colours and the background, i.e., the people in the back, is multi-coloured as well, it is hard for both algorithms, i.e., the algorithm of [42] and our proposed algorithm, to segment only pixels belonging to the foreground object. Moreover, annotated scribbles on the foreground object cover pixels not belonging to the actual corresponding object but to the background. Thus, additionally pixels from the background are assigned to the foreground object's depth value. An equal case can be observed regarding the three persons walking in the background (cf. arrow nr.2 in Figure 5.14). The scribble in the form of a cross (cf. in Figure 5.14(a) lila scribble) annotating these three persons covers many pixels belonging to the street and the background. The result is the same as described above, i.e., the foreground object's depth value is propagated to the background as well. While the results of our WTA approach (cf. Figure 5.14(g)) with a mean squared error of $e_{mse} \times 100 = 0.72$ show a mix of different depth values at the object borders, the DB approach smooths the result by computing a weighted mean of the depth values belonging to the foreground and background object. Thus, the DB approach (cf. Figure 5.14(d)) reduces the error to $e_{mse} \times 100 = 0.56$. However, the algorithm of [42] can handle the above described issues the best, since contrary to our approach, the segmentation algorithm used in [42] is independent from the user-assigned scribbles.

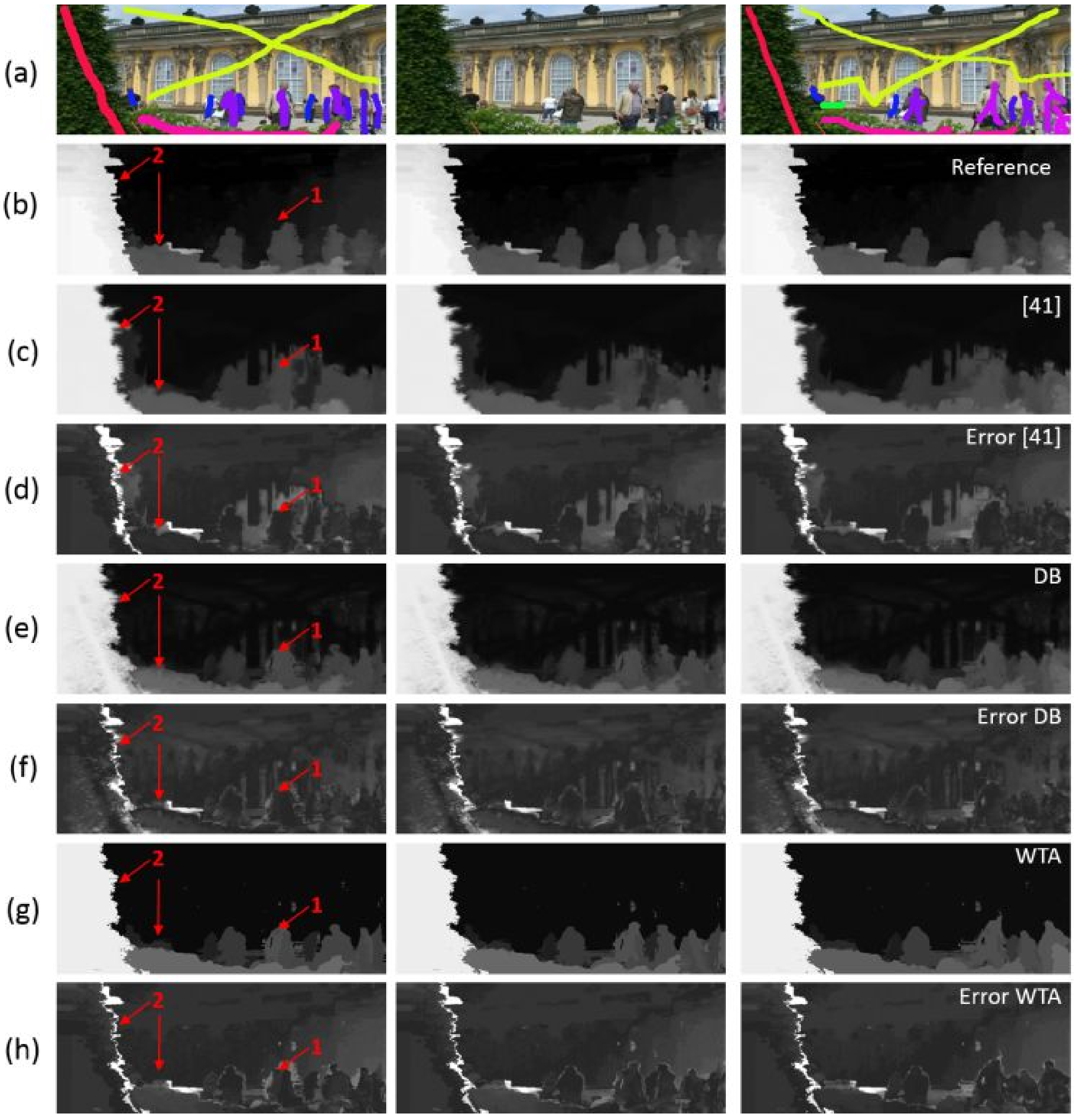


Figure 5.13: Results of test video *Palace*. (a) shows the input frames 1, 5 and 10 with user-assigned scribbles on the first and last frame. (b) shows the corresponding reference solution depth maps. (c) shows results of [42] and (d) the corresponding error with $e_{mse} \times 100 = 1.20$. (e) shows our results of the DB approach and (f) the corresponding error with $e_{mse} \times 100 = 1.16$. (g) shows our results of the WTA approach and (h) the corresponding error with $e_{mse} \times 100 = 0.98$. This video contains persons moving differently, colour similarity between a foreground object and the background (arrow nr. 1) and fine border structures (arrow nr. 2).

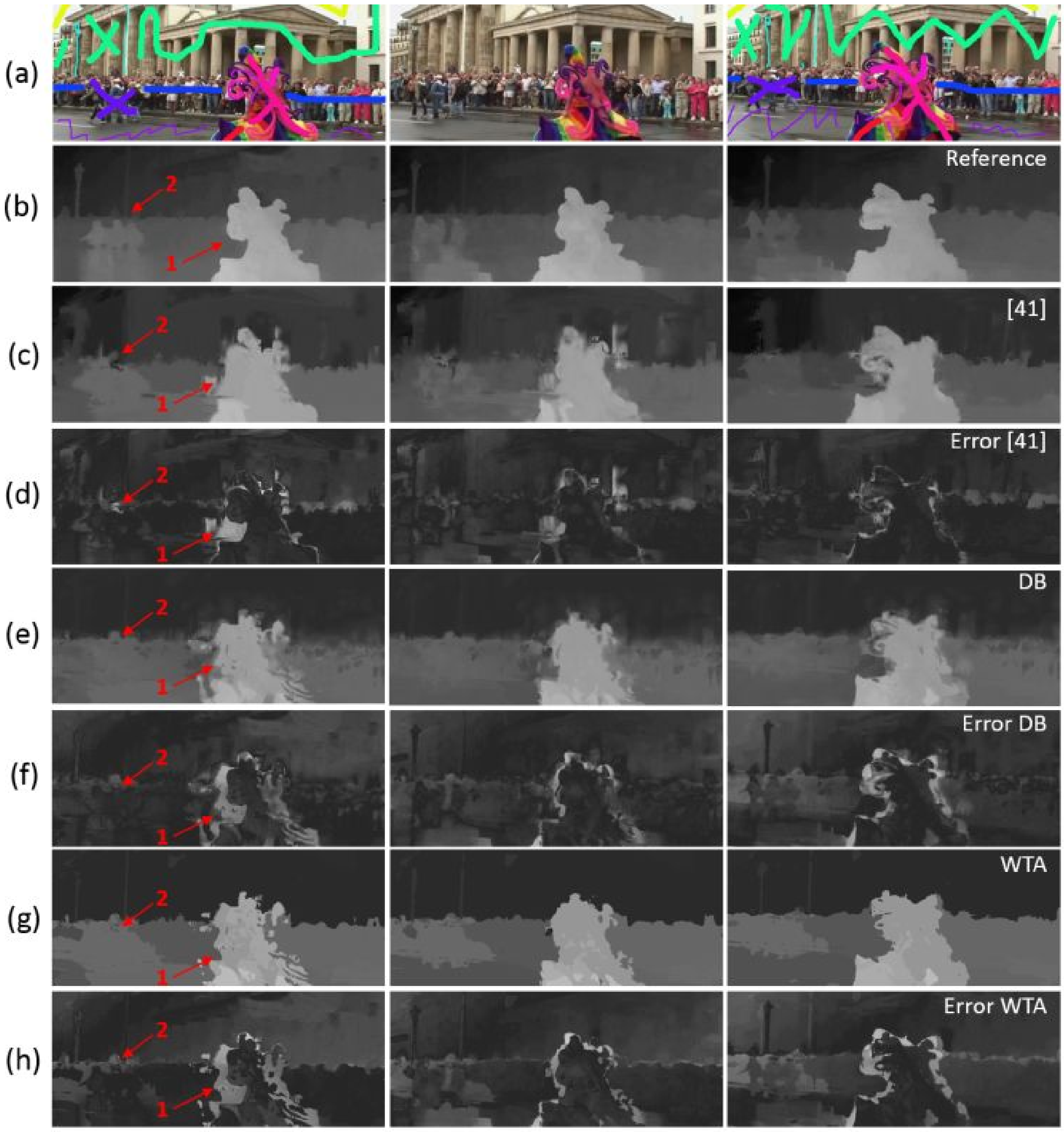


Figure 5.14: Results of test video *Parade*. (a) shows the input frames 1, 5 and 11 with user-assigned scribbles on the first and last frame. (b) shows the corresponding reference solution depth maps. (c) shows results of [42] and (d) the corresponding error with $e_{mse} \times 100 = 0.28$. (e) shows our results of the DB approach and (f) the corresponding error with $e_{mse} \times 100 = 0.56$. (g) shows our results of the WTA approach and (h) the corresponding error with $e_{mse} = 0.72$. This video contains many different colours (e.g., arrow nr. 1) and weak scribble annotations (e.g, arrow nr. 2).

Hence, compared to our DB approach, [42] achieves a result with an error of $e_{mse} \times 100 = 0.28$.

Figure 5.15 shows results of the test video *Head*, where our DB approach achieves the best results ($e_{mse} \times 100 = 0.37$). As can be seen, the test video contains only one foreground object, i.e., the rotating person, occluding the background. As can be observed while the person is rotating throughout the video, a hole between the hand and the body appears through which the background can be seen (cf. Figure 5.15 red arrow). As resulting depth maps of [42] show (cf. Figure 5.15(c)), the algorithm can not recognise this hole. Due to the graph-based segmentation approach which requires regions with the same depth value to be connected, the algorithm of [42] wrongly propagates the depth value corresponding to the scribble annotated on the person. This wrong propagation results in an high error ($e_{mse} \times 100 = 0.65$) compared to our DB approach result ($e_{mse} \times 100 = 0.37$). Contrary to the algorithm of [42], our proposed algorithm performs a global segmentation by using CVF. Thus, the appearing hole is correctly segmented and propagated with the backgrounds depth value. The final WTA depth maps (cf. Figure 5.15(g)) only contain two different depth values, i.e., one for the background and one for the foreground object, which results in the cardboard effect, i.e., a flat 3D viewing experience caused by missing depth variation within the object. Contrary, the DB approach achieves a smoother result. Especially smooth borders in the region of the hair improve the results.

Figure 5.16 shows results of test video *Tsukuba1-100*, where our DB approach achieves the best results ($e_{mse} \times 100 = 0.08$). As can be seen in Figure 5.16(c) and (d) in the red highlighted region, the algorithm of [42] struggles with recognising the hole in the foreground object. This issue was already discussed above in the context of test video *Head*. Figure 5.16(g) and (h) show an enlarged comparison between the result of [42] and our DB approach result. As can be seen, the occurring misassignment of the background region is prevented in the context of our algorithm due to a global segmentation approach. Moreover, Figure 5.16(i) and (j) show another enlarged region green highlighted. In Figure 5.16(i) and (j) it can be seen that the depth change, which appears throughout the video, can be better modelled by our proposed basic depth change model. The divergence of the depth values propagated by [42] compared with the ground truth is larger than with our DB approach.

Conclusion

As seen in the results and discussions presented above, the quality of our depth propagation results depends on the scribble placement. Thus, as discussed above, annotated scribbles that addtionally cover pixels of background objects complicate a valid segmentation and effect the final depth maps, e.g., in the case of test video *Parade*. Such cases can be better handled by the algorithm of [42] since the underlying segmentation process is independent from the annotated scribbles. Therefore, in above mentioned cases, [42] achieves better results. However, due to the global segmentation based on colour, our approach is able to identify regions not connected with each other but belonging to the same object, e.g., a background partly occluded by foreground objects. This, on the one hand, reduces the user effort, since not every disconnected region has to be annotated with a scribble. On the other hand, global segmentation enables the identification of disconnected regions that occur during the video as, e.g., in test videos *Head* and *Tsukuba1-*

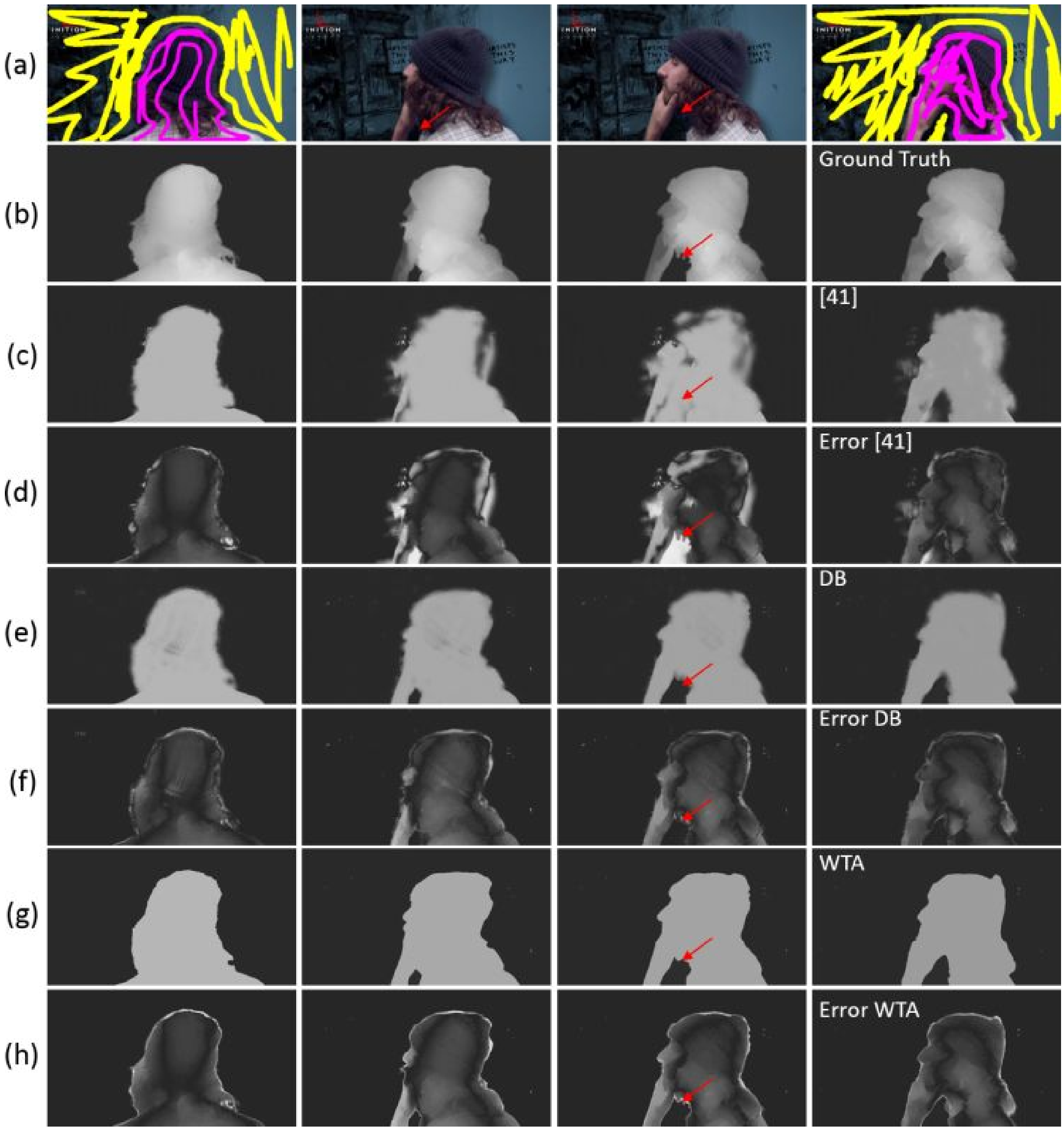


Figure 5.15: Results of test video *Head*. (a) shows the input frames 1, 41, 61 and 81 with user-assigned scribbles on the first and last frame. (b) shows the corresponding ground truth depth maps. (c) shows results of [42] and (d) the corresponding error with $e_{mse} \times 100 = 0.65$. (e) shows our results of the DB approach and (f) the corresponding error with $e_{mse} \times 100 = 0.37$. (g) shows our results of the WTA approach and (h) the corresponding error with $e_{mse} \times 100 = 0.49$. This video contains occlusions of the background caused by the rotating foreground object. The highlighted area shows a disoccluded hole which appears during the video and represents a difficulty for [42].

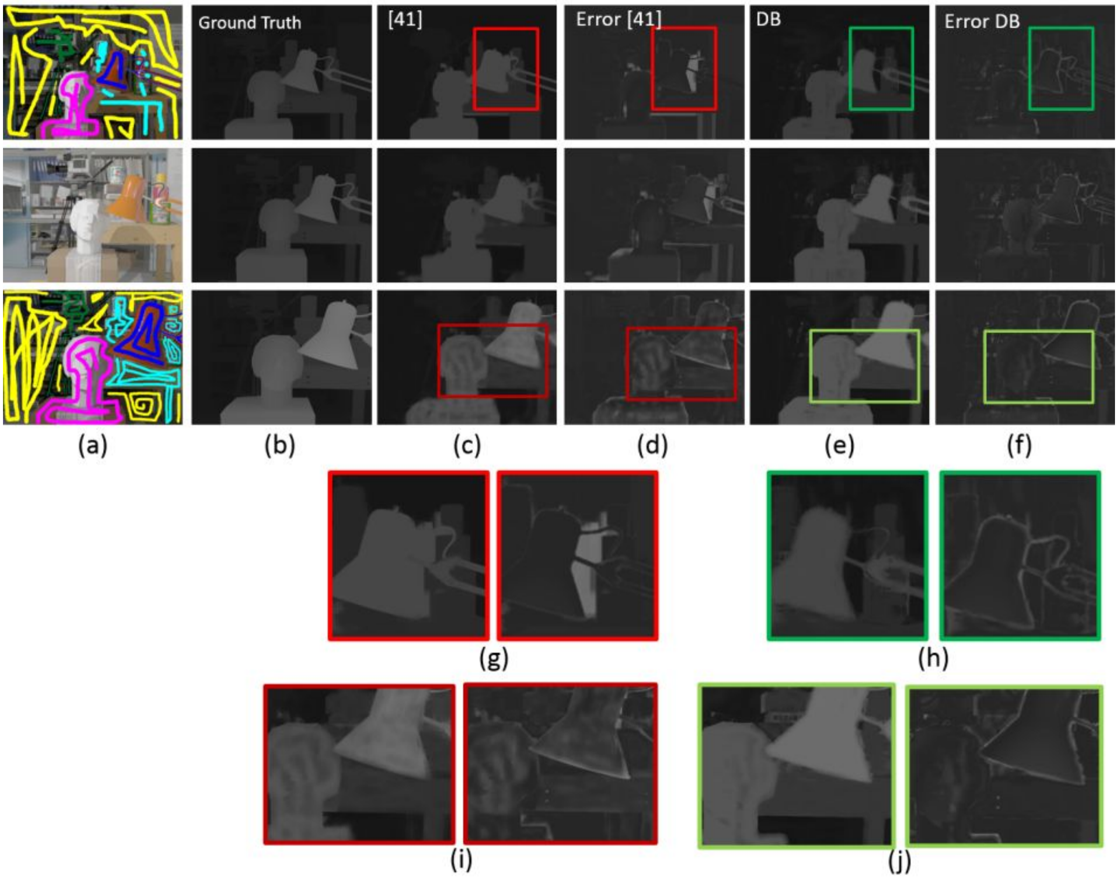


Figure 5.16: Result of test video *Tsukuba1-100*. (a) shows the input frames 1, 50 and 100 with user-assigned scribbles on the first and last frame. (b) shows the corresponding ground truth depth maps. (c) shows results of [42] and (d) the corresponding error with $e_{mse} \times 100 = 0.15$. (e) shows our results of the DB approach and (f) the corresponding error with $e_{mse} \times 100 = 0.08$. (g) and (h) shows an enlargement of the region highlighted in (c)-(f) in frame 1 (first row). (i) and (j) show an enlargement of the region highlighted in (c)-(f) in frame 100 (third row). This video contains a smooth camera movement closer to the scene.

100. Moreover, our algorithm is able to reduce the influence of wrong pixels due to colour ambiguity by applying the spatial influence extension, e.g., in test video *Palace*. Additionally, the DB approach can improve results by smoothing depth values at finely structured borders and enabling depth variants within objects.

In summary, when the user keeps colour similarity, depth positions and movement in the video in mind while annotating scribbles on the first and last frame, the quality is improved by our algorithm. Moreover, the amount of scribble annotations can be kept low by such a robust scribble annotation.

CHAPTER 6

Summary and Outlook

In this thesis, we have proposed a semi-automatic 2D-to-3D depth propagation algorithm which propagates depth on all frames of a video based on scribble-based user input. In particular, we focused on good quality while keeping the user input low. To this end, we extended the video object segmentation algorithm from [11], which performs a colour-based segmentation, to a depth propagation algorithm that achieves temporally coherent depth maps by using cost volume filtering. Only sparse user input is necessary in order to achieve robust depth maps. Thus, the user annotates colour coded scribbles on two frames, i.e., the first and the last frame, which encodes the favoured depth. Moreover, in this thesis we focused on problems such as edge-sharpness, temporal consistency, erroneous propagation caused by segmentation issues and temporal depth changes over time. Therefore, we proposed several extensions addressing these problems. The temporal consistency improvement (TCI) addresses the problem of edge-sharpness and temporal consistency by improving depth propagation at depth boundaries in videos with fast moving objects by filtering along an object's movement throughout the video (cf. Section 4.2). The spatial influence extension (SIE) addresses wrong propagation caused by segmentation issues. The SIE reduces wrong depth assignments by adding a spatial influence term to the cost computation. Thus, pixels which are similar in colour but different in depth have less influence on the final depth propagation (cf. Section 4.3). Finally, the temporal depth change extension (TDCE) addresses temporal depth changes and enables perceptually consistent depth changes of objects throughout the video by generating depth change models that achieve a significant improvement in videos with movement in depth. As our results of the performed evaluation show (cf. Chapter 5), these proposed extensions can achieve a significant improvement of the quality of the computed depth maps depending on the content of the video.

In order to achieve high-quality depth maps, a robust user input is necessary. Therefore, the user has to consider colour similarities between objects, and the depth position and movement of each object throughout the whole video. Moreover, the depth propagation result is dependent on a robust optical flow estimation. Our proposed 2D-to-3D conversion algorithm requires only little user interaction and generates spatio-temporally coherent depth maps with perceptually consistent depth changes for objects that change their depth in time by incorporating the depth order of the video.

Future work may include a more efficient implementation of the described algorithms since we have focused on quality rather than efficiency. This improvement could be achieved by implementing computationally parts of the algorithm on the GPU (e.g., as in [42]). Moreover, a subjective evaluation could be performed in order to elaborate the visual quality of novel views generated from the computed depth maps using our proposed algorithm. Additionally, an evaluation regarding the scribble placement could be a next step in order to examine the impact of scribble placement on the algorithm.

Bibliography

[1] Blender. Available: `http://www.blender.org`. Accessed: 8 December 2014.

[2] A. Agarwala, A. Hertzmann, D. H. Salesin, and S. M. Seitz. Keyframe-based tracking for rotoscoping and animation. In *ACM Transactions on Graphics*, volume 23, pages 584–591. ACM, 2004.

[3] R. Bellman. On a routing problem. Technical report, DTIC Document, 1956.

[4] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, pages 417–424. ACM Press/Addison-Wesley Publishing Co., 2000.

[5] A. Björck. *Numerical methods for least squares problems*. Siam, 1996.

[6] M. Bleyer and M. Gelautz. Simple but effective tree structures for dynamic programming-based stereo matching. In *International Conference on Computer Vision Theory and Applications*, pages 415–422, 2008.

[7] G. Borgefors. Distance transformations in digital images. In *Computer Vision, Graphics, and Image Processing*, volume 34, pages 344–371. Elsevier, 1986.

[8] Y. Boykov and G. Funka-Lea. Graph cuts and efficient nd image segmentation. In *International Journal of Computer Vision*, volume 70, pages 109–131. Springer, 2006.

[9] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 23, pages 1222–1239. IEEE, 2001.

[10] Broadband Network and Digital Media Lab. Available: `http://media.au.tsinghua.edu.cn/2Dto3D/testsequence.html`. Accessed: 16 November 2014.

[11] N. Brosch, A. Hosni, C. Rhemann, and M. Gelautz. Spatio-temporally coherent interactive video object segmentation via efficient filtering. In *Proceedings of the Joint 34th DAGM and 36th OAGM Symposium*, pages 418–427. Springer, 2012.

[12] N. Brosch, C. Rhemann, and M. Gelautz. Segmentation-based depth propagation in videos. In *Proceedings of the ÖAGM/AAPR Workshop 2011*, pages 1–8, 2011.

[13] T. Brox and J. Malik. Object segmentation by long term analysis of point trajectories. In *European Conference on Computer Vision*, pages 282–295. Springer, 2010.

[14] A. Bruhn, J. Weickert, and C. Schnörr. Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. In *International Journal of Computer Vision*, volume 61, pages 211–231. Springer, 2005.

[15] W. Burger and M. J. Burge. *Digitale Bildverarbeitung: Eine Einführung mit Java und ImageJ*. Springer, 2005.

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

[17] J. Canny. A computational approach to edge detection. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, number 6, pages 679–698. IEEE, 1986.

[18] X. Cao, Z. Li, and Q. Dai. Semi-automatic 2D-to-3D conversion using disparity propagation. In *IEEE Transactions on Broadcasting*, volume 57, pages 491–499. IEEE, 2011.

[19] T. L. Carron et al. Color edge detector using jointly hue, saturation and intensity. In *Proceedings of IEEE International Conference on Image Processing*, volume 3, pages 977–981. IEEE, 1994.

[20] R. C. Carter and E. C. Carter. Cie l* u* v* color-difference equations for self-luminous displays. In *Color Research & Application*, volume 8, pages 252–253. Wiley Online Library, 1983.

[21] J. Cheolkon, Z. Xiaohua, W. Lei, S. Tian, H. Mingchen, H. Biao, and J. Licheng. 2D to 3D Conversion in 3DTV Using Depth Map Generation and Virtual View Synthesis. In *3rd International Conference on Multimedia Technology*. Atlantis Press, 2013.

[22] B. Child. Peter jackon's hobbit triology production costs revealed at 560m. Available: `http://www.theguardian.com/film/2013/oct/07/peter-jackson-hobbit-production-costs`. Accessed: 16 June 2014.

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

[24] L. S. Davis. A survey of edge detection techniques. In *Computer Graphics and Image Processing*, volume 4, pages 248–270. Elsevier, 1975.

[25] C. Fehn and R. Pastoor. Interactive 3-dtv-concepts and key technologies. In *Proceedings of the IEEE*, volume 94, pages 524–538. IEEE, 2006.

[26] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. In *International Journal of Computer Vision*, volume 59, pages 167–181. Springer, 2004.

[27] J. D. Foley, A. Van Dam, S. K. Feiner, J. F. Hughes, and R. L. Phillips. *Introduction to computer graphics*, volume 55. Addison-Wesley Reading, 1994.

[28] L. R. Ford. Network flow theory. 1956.

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

[30] S. Ghuffar, N. Brosch, N. Pfeifer, and M. Gelautz. Motion estimation and segmentation in depth and intensity videos. In *Integrated Computer-Aided Engineering*, volume 21, pages 203–218. IOS Press, 2014.

[31] L. Grady. Random walks for image segmentation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 28, pages 1768–1783. IEEE, 2006.

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

[33] M. Guttmann, L. Wolf, and D. Cohen-Or. Semi-automatic stereo extraction from video footage. In *IEEE 12th International Conference on Computer Vision*, pages 136–142. IEEE, 2009.

[34] R. M. Haralick, K. Shanmugam, and I. H. Dinstein. Textural features for image classification. In *IEEE Transactions on Systems, Man and Cybernetics*, number 6, pages 610–621. IEEE, 1973.

[35] R. M. Haralick and L. G. Shapiro. Image segmentation techniques. In *Technical Symposium East*, pages 2–9. International Society for Optics and Photonics, 1985.

[36] P. V. Harman, J. Flack, S. Fox, and M. Dowley. Rapid 2D-to-3D conversion. In *Electronic Imaging*, pages 78–86. International Society for Optics and Photonics, 2002.

[37] K. He, J. Sun, and X. Tang. Guided image filtering. In *European Conference on Computer Vision*, pages 1–14. Springer, 2010.

[38] B. K. Horn and B. G. Schunck. Determining optical flow. In *Technical Symposium East*, pages 319–331. International Society for Optics and Photonics, 1981.

[39] A. Hosni, C. Rhemann, M. Bleyer, and M. Gelautz. Temporally consistent disparity and optical flow via efficient spatio-temporal filtering. In *Advances in Image and Video Technology*, pages 165–177. Springer, 2011.

[40] A. Hosni, C. Rhemann, M. Bleyer, C. Rother, and M. Gelautz. Fast cost-volume filtering for visual correspondence and beyond. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 35, pages 504–511. IEEE, 2013.

[41] B. Huhle, S. Fleck, and A. Schilling. Integrating 3D time-of-flight camera data and high resolution images for 3dtv applications. In *3DTV Conference*, pages 1–4. IEEE, 2007.

[42] M. Ivancsics. Effiziente Tiefenpropagierung in Videos mit GPU-Unterstützung. Master's thesis, Vienna University of Technology, 2013.

[43] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. In *International Journal of Computer Vision*, volume 1, pages 321–331. Springer, 1988.

[44] J. J. Koenderink, A. J. van Doorn, A. M. Kappers, and J. T. Todd. Ambiguity and themental eye in pictorial relief. In *Perception-London*, volume 30, pages 431–448. Pion Ltd, 2001.

[45] J. Kopf, M. F. Cohen, D. Lischinski, and M. Uyttendaele. Joint bilateral upsampling. In *ACM Transactions on Graphics*, volume 26, page 96. ACM, 2007.

[46] M. Kung. George Lucas says converting ‚Star Wars' to 3D has cost more than original movie. Available: `http://starwars.wikia.com/wiki/Star_Wars_Episode_I:_The_Phantom_Menace`. Accessed: 17 June 2014.

[47] M. Lang, O. Wang, T. Aydin, A. Smolic, and M. H. Gross. Practical temporal consistency for image-based graphics applications. In *ACM Transactions on Graphics*, volume 31, page 34, 2012.

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

[49] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum. Lazy snapping. In *ACM Transactions on Graphics*, volume 23, pages 303–308. ACM, 2004.

[50] G.-S. Lin, J.-F. Huang, and W.-N. Lie. Semi-automatic 2D-to-3D video conversion based on depth propagation from key-frames. In *20th IEEE International Conference on Image Processing*, pages 2202–2206, 2013.

[51] C. Liu. *Beyond pixels: exploring new representations and applications for motion analysis*. PhD thesis, Electrical Engineering and Computer Science at the Massachusetts Institute of Technology, 2009.

[52] L. Liu, C. Ge, N. Zheng, Q. Li, and H. Yao. Spatio-temporal adaptive 2D to 3D video conversion for 3DTV. In *IEEE International Conference on Consumer Electronics*, pages 465–466. IEEE, 2012.

[53] D. G. Lowe. Distinctive image features from scale-invariant keypoints. In *International Journal of Computer Vision*, volume 60, pages 91–110. Springer, 2004.

[54] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of 7th International Joint Conference on Artifical Intelligence*, volume 81, pages 674–679, 1981.

[55] L. Luccheseyz and S. Mitray. Color image segmentation: A state-of-the-art survey. In *Proceedings of the Indian National Science Academy*, volume 67, pages 207–221, 2001.

[56] S. Martull, M. Peris, and K. Fukui. Realistic cg stereo image dataset with ground truth disparity maps. In *International Conference on Pattern Recognition Workshop TrakMark2012*, volume 111, pages 117–118, 2012.

[57] K. McLaren. Xiii—the development of the cie 1976 (l* a* b*) uniform colour space and colour-difference formula. In *Journal of the Society of Dyers and Colourists*, volume 92, pages 338–341. Wiley Online Library, 1976.

[58] H.-J. Mittag. *Statistik: eine interaktive Einführung*. Springer-Verlag, 2012.

[59] E. N. Mortensen and W. A. Barrett. Intelligent scissors for image composition. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, pages 191–198. ACM, 1995.

[60] P. Ochs, J. Malik, and T. Brox. Segmentation of moving objects by long term video analysis. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 1. IEEE, 2013.

[61] A. S. Ogale and Y. Aloimonos. A roadmap to the integration of early visual modules. In *International Journal of Computer Vision*, volume 72, pages 9–25. Springer, 2007.

[62] OpenCV. Opencv optical flow. Available: `http://docs.opencv.org/trunk/doc/py_tutorials/py_video/py_lucas_kanade/py_lucas_kanade.html#dense-optical-flow-in-opencv`. Accessed: 22 August 2014.

[63] G. Palou and P. Salembier. Depth order estimation for video frames using motion occlusions. In *IET Computer Vision*, volume 8, pages 152–160. IET, 2013.

[64] G. Paschos. Perceptually uniform color spaces for color texture analysis: an empirical evaluation. In *IEEE Transactions on Image Processing*, volume 10, pages 932–937. IEEE, 2001.

[65] M. Peris, A. Maki, S. Martull, Y. Ohkawa, and K. Fukui. Towards a simulation driven stereo vision system. In *21st International Conference on Pattern Recognition*, pages 1038–1042. IEEE, 2012.

[66] R. Phan and D. Androutsos. Robust Semi-Automatic Depth Map Generation in Unconstrained Images and Video Sequences for 2D to Stereoscopic 3D Conversion. In *IEEE Transactions on Multimedia*, volume 16, pages 122–136. IEEE, 2014.

[67] S. Porter, M. Mirmehdi, and B. Thomas. Detection and classification of shot transitions. In *Proceedings of the 12th British Machine Vision Conference*, pages 73–82. Citeseer, 2001.

[68] T. P. Roosendaal. Sintel. blender foundation, durian open movie project (2010). Available: `http://www.sintel.org`. Accessed: 8 December 2014.

[69] R. Rzeszutek and D. Androutsos. Efficient automatic depth estimation for video. In *18th International Conference on Digital Signal Processing*, pages 1–6. IEEE, 2013.

[70] D. Scharstein. Middlebury optical flow benchmark. Available: `http://vision.middlebury.edu/flow/`. Accessed: 12 October 2014.

[71] M. Seymour. Art of Stereo Conversion: 2D to 3D. Available: `http://www.fxguide.com/featured/art-of-stereo-conversion-2d-to-3d-2012/`. Accessed: 17 June 2014.

[72] J. Shi and J. Malik. Normalized cuts and image segmentation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 22, pages 888–905. IEEE, 2000.

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

[74] J. Smisek, M. Jancosek, and T. Pajdla. 3D with Kinect. In *Consumer Depth Cameras for Computer Vision*, pages 3–25. Springer, 2013.

[75] A. Smolic, P. Kauff, S. Knorr, A. Hornung, M. Kunter, M. Muller, and M. Lang. Three-dimensional video postproduction and processing. In *Proceedings of the IEEE*, volume 99, pages 607–625. IEEE, 2011.

[76] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis, and Machine Vision*. 1998.

[77] Star Wars Episode I: The Phantom Menace. Available: `http://blogs.wsj.com/speakeasy/2011/03/30/george-lucas-says-converting-star-wars-to-3-d-has-cost-more-than-original-movie/`. Accessed: 17 June 2014.

[78] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *European Conference on Computer Vision*, pages 438–451. Springer, 2010.

[79] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2010.

[80] R. Tarjan. Depth-first search and linear graph algorithms. In *SIAM Journal on Computing*, volume 1, pages 146–160. Society for Industrial and Applied Mathematics (SIAM), 1972.

[81] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Sixth International Conference on Computer Vision*, pages 839–846. IEEE, 1998.

[82] E. Turetken and A. A. Alatan. Temporally consistent layer depth ordering via pixel voting for pseudo 3D representation. In *3DTV Conference: The True Vision-Capture, Transmission and Display of 3D Video*, pages 1–4. IEEE, 2009.

[83] C. Varekamp and B. Barenbrug. Improved depth propagation for 2D to 3D video conversion using key-frames. IET, 2007.

[84] G. Wahba. *Spline models for observational data*, volume 59. Society for Industrial and Applied Mathematics (SIAM), 1990.

[85] H. Wang, Y. Yang, L. Zhang, Y. Yang, and B. Liu. 2D-to-3D conversion based on depth-from-motion. In *International Conference on Mechatronic Science, Electric Engineering and Computer*, pages 1892–1895. IEEE, 2011.

[86] O. Wang, M. Lang, M. Frei, A. Hornung, A. Smolic, and M. Gross. Stereobrush: interactive 2D to 3D conversion using discontinuous warps. In *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling*, pages 47–54. ACM, 2011.

[87] C. Wu, G. Er, X. Xie, T. Li, X. Cao, and Q. Dai. A novel method for semi-automatic 2D to 3D video conversion. In *3DTV Conference: The True Vision-Capture, Transmission and Display of 3D Video*, pages 65–68. IEEE, 2008.

[88] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 15, pages 1101–1113. IEEE, 1993.

[89] L. Yatziv and G. Sapiro. Fast image and video colorization using chrominance blending. In *IEEE Transactions on Image Processing*, volume 15, pages 1120–1129. IEEE, 2006.

[90] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. In *ACM Computing Surveys*, volume 38, page 13. ACM, 2006.

[91] L. Zhang, C. Vázquez, and S. Knorr. 3D-TV content creation: automatic 2D-to-3D video conversion. In *IEEE Transactions on Broadcasting*, volume 57, pages 372–383. IEEE, 2011.