Spatio-temporal Video Analysis for Semi-automatic 2D-to-3D Conversion

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktorin der technischen Wissenschaften

by

Nicole Brosch
Registration Number 0325237

to the Faculty of Informatics
at Technische Universität Wien

Advisors: ao. Univ.-Prof. Mag. Dipl.-Ing. Dr.techn. Margrit Gelautz
Univ.-Prof. Dipl.-Ing. Dr.techn. Markus Rupp

The dissertation has been reviewed by:

(ao. Univ.-Prof. Mag. Dipl.-Ing. Dr.techn. Margrit Gelautz)

(Univ.-Prof. Dipl.-Ing. Dr.techn. Markus Rupp)

Wien, 12.09.2016

(Nicole Brosch)

Erklärung zur Verfassung der Arbeit

Nicole Brosch
Wallrißstraße 64/6, 1180 Wien

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 like to express my appreciation to all who provided me the possibility to pursue this research and supported me during it. First of all, I would like to thank my principal supervisor, Margrit Gelautz, for introducing me to the idea of pursuing a Ph.D in the first place and her advice throughout it. I also thank the Doctoral College faculty team, my colleagues, reviewers and proofreaders for the time and expertise they have invested. I am grateful for their feedback. In this context, I particularly acknowledge Markus Rupp, who is the second supervisor and reviewer of this thesis.

Given the opportunity to engage in related research projects, teaching and to visit an international summer school, enabled many interesting discussions and exchanges of ideas with collaborators, students and researchers. Thank you all for sharing your insights. I highly appreciate the support from my former students, Manuel Ivancsics, Tanja Schausberger and Matej Nezveda, who I supervised during their respective master theses. Manuel's and Tanja's works significantly contributed to the results presented in Chapter 4 and Chapter 5 of this thesis. In addition, many thanks go to all researchers who (publicly) shared their source code and test data. You certainly made my life (and the life of other researchers) easier. In this context, I want to particularly acknowledge the authors of the MPI Sintel dataset for providing test data and Raymond Phan for sharing his source code.

I gratefully acknowledge the funding sources that made it financially possible to pursue my Ph.D and all people, such as my supervisors, who were involved in allocating them. This work was supported by the Doctoral College on Computational Perception at TU Vienna and the Austrian Science Fund (FWF) under project P19797. The IEEE Austria Student Mobility and Conference Grant covered some travel expenses. Partial support for this research was provided by the Vienna Science and Technology Fund (WWTF) through project ICT08-019. Additional funding was received from the Technology Agency of the City of Vienna (ZIT) and the Austrian Research Promotion Agency (FFG) under project Hyperion3D (no. 840222).

Last, and most importantly a special thanks to my friends and family. I would like to thank everybody who listened to my never ending whining – dt. *raunzen*. This is especially true for my colleagues with whom I laughed away our joint Ph.D blues. Lastly, I thank my partner, Stefan, who has supported me in more ways than I can name, from putting up with my moods and giving encouraging pep talks to force me to go out to keep things in perspective.

Thank you!

Abstract

This thesis addresses the problem of cost-efficiently converting monoscopic (2D) videos to stereoscopic (3D) videos. Common practices to perform such a 2D-to-3D conversion are labor-intensive *manual* conversions, which are typically used for high-quality 3D cinema productions, and *fully-automatic* conversions of lower conversion quality, which may be integrated into, e.g., (auto-)stereoscopic TVs. In this thesis we focus on *semi-automatic* 2D-to-3D conversions, which can be seen as a compromise between fully-automatic and manual techniques. Such approaches are typically based on sparse user-given disparity (or depth) information, which is propagated to each pixel in a 2D video by assuming a color constancy model. This process ideally requires only minimal user input and efficiently generates disparity maps of high conversion quality, which are suitable for rendering a second 2D video that completes the 3D video. In order to avoid common artifacts related to such propagations, e.g., over-smoothed results and spatio-temporal or perceptual incoherencies, we exploit spatio-temporal segmentation information. The thesis presents two novel semi-automatic 2D-to-3D conversion algorithms that view segmentation as an integral part of the conversion process and are based on comfortable user input in the form of sparse scribbles drawn in the first (and last) frame of a 2D video.

Our first 2D-to-3D conversion algorithm tackles 2D-to-3D conversion and segmentation in a joint approach. It propagates available disparities between neighboring pixels while assigning them to the same segment. In this manner, our algorithm generates disparity maps that capture object borders in the 2D video and contain smooth disparity changes within segments and over time, which is challenging for currently available algorithms. We also provide a scalable implementation that achieves interactive runtimes of one frame per second (resolution of approximately 0.3 megapixels).

The second 2D-to-3D conversion algorithm takes a step towards the generation of perceptually coherent disparity maps. In particular, it enables temporal disparity interpolations that are performed in accordance with motion-caused occlusions between segments. This results in spatio-temporally coherent disparity maps in which disparities of moving objects harmonize with those of nearby objects. The presented segmentation algorithm, used in the conversion algorithm, relies on a spatio-temporal filtering scheme and, thus, achieves fast processing speeds (250 frames per second for a video with a resolution of approximately 0.2 megapixels per frame).

We compare our own algorithms with different semi-automatic 2D-to-3D conversion algorithms suggested in the literature and achieve results of high conversion quality. In this context, our algorithms outperform a well-established conversion algorithm. As opposed to most earlier studies, our final evaluation study is performed under consideration of different scribbling strategies and provides practical insights into the annotation process by investigating the performance of various scribble placement techniques in conjunction with different 2D image content.

Kurzfassung

Diese Arbeit widmet sich der kosteneffizienten Konvertierung von monoskopischen (2D) zu stereoskopischen (3D) Videos. Dabei stellen semi-automatische 2D-zu-3D Konvertierungsverfahren einen Kompromiss zwischen professionellen, aber aufwändigen manuellen und qualitativ schlechteren, vollautomatischen Verfahren dar. Semi-automatische Verfahren propagieren, unter der Annahme von Farbkonsistenz, von BenutzerInnen gegebene Tiefeninformation (Disparität) über das komplette 2D-Video. Die so generierten Disparitätsvideos können für das Erstellen eines zweiten 2D-Videos, welches das 3D-Video vervollständigt, verwendet werden. Ein ideales Konvertierungsverfahren vereint Faktoren wie hohe Qualität der generierten Disparitätsvideos, geringen Arbeitsaufwand für BenutzerInnen und kurze Laufzeiten miteinander. Dabei gilt es, typische Artefakte wie übermäßiges Glätten, raum-zeitliche Inkohärenz oder Konflikte zwischen generierter und im 2D-Video wahrgenommener Tiefe zu verhindern. Der wissenschaftliche Beitrag dieser Arbeit umfasst zwei semi-automatische 2D-zu-3D Konvertierungsalgorithmen, in denen raum-zeitliche Segmentierung einen integralen Bestandteil darstellt. Sie basieren auf groben Initialisierungen mit Disparitäts-Scribbles im ersten (und letzten) Frame des 2D-Videos.

Der erste Algorithmus propagiert die spärlich vorgegebenen Disparitäten im Zuge des Segmentierungsprozesses auf das gesamte Video. Disparitäten werden zwischen Nachbarpixeln propagiert, wenn diese zu einem Segment zusammengefasst werden. Diese Vorgehensweise verhindert übermäßiges Glätten von Disparitäten über Segmentgrenzen und ermöglicht raum-zeitlich kohärente Disparitätsübergänge innerhalb von Segmenten. Eine skalierbare Implementierung erlaubt effiziente Konvertierungsvorgänge (ein Frame pro Sekunde für Auflösungen von 0.3 Megapixel).

Der zweite Algorithmus beschäftigt sich mit Konflikten zwischen generiertem und im ursprünglichen 2D-Video wahrgenommenen Tiefeneindruck. Der Algorithmus bindet bewegungsbedingte Verdeckungen im 2D-Video in die zeitlichen Interpolierungen von Disparitäten ein. Dies führt zu realistischeren Tiefendarstellungen von Objekten, die sich im Laufe des Videos in der Tiefe bewegen. Der dabei verwendete Segmentierungsalgorithmus basiert auf effizienten Filteroperationen und erreicht geringe Laufzeiten (250 Frames pro Sekunde für Auflösungen von 0.2 Megapixel).

In dieser Arbeit entwickelte Algorithmen werden mit semi-automatischen 2D-zu-3D Konvertierungsalgorithmen aus der Literatur verglichen und generieren dabei Disparitätsvideos von hoher Qualität. Die Qualität ihrer Ergebnisse überrtrifft die eines etablierten Algorithmus. Eine abschließende Evaluierung berücksichtigt zusätzlich verschiedene Strategien der Scribble-Platzierung, welche die Konvertierungsergebnisse stark beeinflussen können. Eine Untersuchung dieser Strategien im Zusammenhang mit verschiedenen 2D-Bildinhalten sowie ihrer Robustheit gegenüber Ungenauigkeiten bei der Scribble-Platzierung gibt praktische Einblicke in den Scribble-basierten Initialisierungsprozess, welchem in der vorhandenen Literatur nur geringe Aufmerksamkeit geschenkt wird.

Contents

Abstract v

Kurzfassung vii

Contents ix

List of Figures xiii

List of Tables xvii

Acronyms 1

1 Introduction 1
- 1.1 Motivation 1
- 1.2 Problem Statement 2
- 1.3 Contributions 3
- 1.4 Resulting Publications 5
- 1.5 Structure of the Work 6

2 State-of-the-Art of 2D-to-3D Conversion 9
- 2.1 Principles of 3D Content Generation 9
  - 2.1.1 3D from Stereoscopic Data 10
  - 2.1.2 3D from Monoscopic Data 12
- 2.2 Prior Work on Semi-automatic 2D-to-3D Conversion 13
  - 2.2.1 Optimization-based Techniques 14
  - 2.2.2 Filter-based Techniques 17
  - 2.2.3 Segmentation-based Techniques 21
- 2.3 Summary 22

3 Fundamentals and Related Work of 2D-to-3D Conversion 23
- 3.1 User Interaction 23
- 3.2 Spatio-temporal Video Analysis 25
  - 3.2.1 Graph-based Representation 25
  - 3.2.2 Segmentation 27

3.2.3 Edge-aware Interpolation 32
3.3 Evaluation 33
3.4 Summary 36

4 Segmentation-based 2D-to-3D Conversion of Videos 37
4.1 Introduction 37
4.2 Proposed Algorithm 38
4.2.1 Disparity Scribbles 38
4.2.2 Joint Segmentation and Propagation 39
4.2.3 Disparity Interpolation and Refinement 42
4.3 Efficient Implementation Using Multicore Technology 44
4.3.1 Joint Segmentation and Propagation 45
4.3.2 Disparity Interpolation and Refinement 46
4.3.3 Clip-based 2D-to-3D Conversion 47
4.4 Experimental Results and Evaluation 49
4.5 Summary 61

5 Cost Volume Filtering for Video Segmentation and 2D-to-3D Conversion 63
5.1 Introduction 63
5.2 Proposed Algorithm 65
5.2.1 Interactive Video Object Segmentation via Cost Volume Filtering . . . 66
5.2.2 Semi-automatic 2D-to-3D Conversion via Cost Volume Filtering 70
5.3 Experimental Results and Evaluation 82
5.3.1 Video Object Segmentation Results 82
5.3.2 2D-to-3D Conversion Results 85
5.4 Summary 100

6 Evaluation of 2D-to-3D Conversion Systems in Conjunction with User Input 107
6.1 Introduction 107
6.2 Evaluation Strategy 109
6.2.1 Benchmark Setup 109
6.2.2 Scribble Generation 111
6.2.3 Investigated Semi-automatic 2D-to-3D Conversion Systems 115
6.3 Experimental Results and Evaluation 117
6.3.1 Conversion Accuracy 117
6.3.2 Robustness to Scribble Perturbations 119
6.3.3 Error Tolerance 122
6.3.4 Content-aware Scribble Placement 125
6.4 Summary 130

7 Conclusions and Future Work 133
7.1 Conclusions 133
7.2 Future Research Topics 134

A Descriptions of Used Datasets **137**

B Supplements to Cost Volume Filtering for 2D-to-3D Conversion **145**
B.1 Program Flow 145
B.2 Additional Evaluation Results 145
B.2.1 Comparison of Different Algorithm Versions with Depth Blending . . . 145
B.2.2 Quantitative Comparison of Naive and Depth Order Guided Interpolation 146

Bibliography **153**

List of Figures

2.1 Standard rectified epipolar geometry 11
2.2 Example user input and disparity map 13
2.3 Examples of optimization-based disparity propagation 16
2.4 Illustration of filter-based disparity propagation 18
2.5 Illustration of segmentation-based disparity propagation 21

3.1 User interaction examples 24
3.2 Disparity scribble examples 24
3.3 Graph-based representation 26
3.4 Illustration of automatic segmentation algorithms 28
3.5 Illustration of interactive segmentation algorithms 29
3.6 Illustration of common forms of motion information 31
3.7 Reference solutions used in evaluation 34

4.1 Input and output example of proposed 2D-to-3D conversion algorithm 38
4.2 Program flow of proposed segmentation-based 2D-to-3D conversion algorithm . . 39
4.3 Propagation rule set 40
4.4 Joint segmentation and propagation process 41
4.5 Illustration of region-graph 42
4.6 Segment-wise disparity interpolation 43
4.7 Efficient disparity interpolation 47
4.8 Illustration of clip-based disparity propagation 48
4.9 Our 2D-to-3D conversion results 50
4.10 Our 2D-to-3D conversion results with temporal disparity changes 51
4.11 Novel views generated from proposed conversion results 52
4.12 Quantitative evaluation and comparison to [56] 54
4.13 Evaluation of region merging step's influence (*Baby2*) 56
4.14 Evaluation of the region merging step's influence (*Ambush5*) 57
4.15 Computational efficiency evaluation of segment-wise interpolation 58
4.16 Computational efficiency evaluation of memory optimization 59
4.17 Evaluation of optimized 2D-to-3D conversion algorithm 62

5.1 Illustration of progressive labeling 66
5.2 Effect of local editing 67

5.3 Temporally coherent cost volume filtering 68
5.4 Spatio-temporal filter kernel 69
5.5 Overview of semi-automatic 2D-to-3D conversion algorithm 71
5.6 Cost volume with multiple scribble labels 72
5.7 Effect of spatio-temporal closeness constraint and 3D connectivity constraint . . . 74
5.8 Improved temporal coherence 75
5.9 Motion guided filtering 76
5.10 Smooth disparity changes 77
5.11 Temporal disparity change models 78
5.12 Disparity restriction example 79
5.13 Step-by-step interpolation in temporal disparity change models 82
5.14 Our video object segmentations 83
5.15 Video object segmentation based only on data cost 84
5.16 Failure case of proposed video object segmentation algorithm 84
5.17 Visual comparison of proposed video object segmentation algorithm to [125] . . . 86
5.18 Failure case of spatio-temporal alpha matting 86
5.19 Visual comparison to [4] 87
5.20 Our 2D-to-3D conversion results 88
5.21 Novel views generated from our conversion results 89
5.22 Comparison of proposed algorithm with dis- and enabled disparity change models . 91
5.23 Comparison of assignment schemes 92
5.24 Comparison of interpolation techniques (GT OF) 93
5.25 Comparison of interpolation techniques (estimated OF) 94
5.26 Comparison of results with different motion information (*Temple3*) 96
5.27 Comparison of results with different motion information (*Ambush2*) 97
5.28 Visual evaluation of sensitivity to illumination effects 98
5.29 Failure case due to illumination effects 99
5.30 Visual evaluation and comparison to [56] and [118] (*Parade*) 102
5.31 Visual evaluation and comparison to [56] and [118] (*Stairs*) 103
5.32 Visual evaluation and comparison to [118] (*Palace*) 104
5.33 Visual evaluation and comparison to [118] (*Tsukuba1*) 105

6.1 Overview of test data 110
6.2 Scribble generation and scribbling strategies 112
6.3 Frequently employed scribbling strategy 113
6.4 Illustration of automatic generation of line scribbles 113
6.5 Illustration of automatic generation of expert scribbles 114
6.6 Border scribbles in the literature 114
6.7 Visual evaluation of conversion results (*Ambush2*) 118
6.8 Example results after scribble perturbations (*Temple3*) 121
6.9 Visual evaluation of error tolerance (*Ambush2*) 124
6.10 Region classification example 126
6.11 Evaluation of content-aware scribble placement (multicolored regions) 126

6.12 Evaluation of content-aware scribble placement (homogenous regions) 127
6.13 Examples of multicolored region with strong borders 128
6.14 Evaluation of content-aware scribble placement (strong border regions) 129
6.15 Evaluation of content-aware scribble placement (weak border regions) 129

A.1 Tsukuba dataset 140
A.2 Sintel dataset 141
A.3 Long video dataset 142
A.4 Middlebury image dataset 143
A.5 Recorded stereo dataset 144

B.1 Program flow for MS 146
B.2 Program flow for STC 147
B.3 Program flow for CVF 147
B.4 Program flow for disparity assignment (WTA or DB) 148
B.5 Program flow for CON 148
B.6 Program flow for DC -tM and DC -sM 149

List of Tables

4.1 Quantitative evaluation and comparison to [56] 51
4.2 Quantitative evaluation of region merging step's influence 55
4.3 Computational efficiency evaluation of joint segmentation and propagation 58
4.4 Computational efficiency evaluation of segment-wise interpolation 59
4.5 Quantitative evaluation of optimized 2D-to-3D conversion algorithm 60
4.6 Quantitative evaluation and computational efficiency evaluation of clip-based 2D-to-3D conversion algorithm 61

5.1 Binary segmentation – comparison of per-frame [125] and spatio-temporal approach 85
5.2 Soft segmentation – comparison of per-frame [125] and spatio-temporal alpha matting 85
5.3 Comparison of different algorithm versions (WTA) 95
5.4 Evaluation of sensitivity to illumination effects (WTA, GT OF) 98
5.5 Comparison to related 2D-to-3D conversion algorithms 101

6.1 Quantitative evaluation of conversion accuracy 117
6.2 Quantitative evaluation of robustness to scribble perturbations (scribble comparison) 120
6.3 Quantitative evaluation of robustness to scribble perturbations (system comparison) 120
6.4 Quantitative evaluation of error tolerance 123

B.1 Comparison of different algorithm versions (DB) 150
B.2 Quantitative comparison of naive interpolation and proposed depth order guided interpolation (GT OF, WTA) 151
B.3 Quantitative comparison of naive interpolation and proposed depth order guided interpolation (estimated OF, WTA) 151

Acronyms

1D	One-dimensional
2D	Two-dimensional
3D	Three-dimensional
Avg.	Average
BS	Border Scribbles
CON	3D connectivity constraint
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
CVF	Cost Volume Filtering [125]
DAG	Directed Acyclic Graph
DC-tM, -sM	Disparity Change, -time Model, -segment Model
DB	Disparity Blending
DS	Dot Scribbles
EE	Endpoint Error [6]
ES	Expert Scribbles
FE	Flickering Error
FPS	Frames Per Second
GF	Guided Filtering [59]
GPU	Graphics Processing Unit
GS+P	Graph-based Segmentation and Propagation

GT	Ground Truth
GC+RW	Graph Cuts and Random Walks based 2D-to-3D conversion algorithm [118]
HD	High Definition
LAB	Lightness, color-opponent dimensions a,b
LS	Line Scribbles
LSE	2D-to-3D conversion algorithm based on Least Squares Estimation [56]
MS	Multiple Scribble labels
MSE	Mean Squared Error
MST	Minimum Spanning Tree
OF	Optical Flow
PS	Point Scribbles
RGB	Red, Green, Blue
SIFT	Scale Invariant Feature Transform [98]
STC	Spatio-Temporal Closeness constraint
TC	Improved temporal coherence (i.e., motion guided filtering)
TS	Thinning Scribbles
WTA	Winner-Takes-All

CHAPTER 1

Introduction

1.1 Motivation

Over the past years, the three-dimensional (*3D*) viewing experience has become a main attraction of the entertainment industry. 3D movies were successfully released (e.g., Avatar [176]), consumer stereo displays and 3DTV sets are commercially available, first commercial 3DTV channels (e.g., Sky 3D [140]) air 3D content and YouTube [172] allows users to upload and watch 3D videos. To maintain the success of the 3D technology, its practical usage is a crucial point. Moreover, the availability of 3D content is a bottleneck. In this context, generating new 3D content, as well as the conversion of existing two-dimensional (*2D*) content to 3D are important topics. Generally, there are several ways to generate 3D content that can be viewed on stereoscopic devices, including (1) the usage of a stereo camera during recording (e.g., Avatar [176]), (2) conversion of existing 2D content to 3D in post-production (e.g., Star Wars [178]) and (3) (re-)rendering computer animated content with 3D models (e.g., Toy Story [177]). While the latter approach is only possible for computer animated content, the first two approaches address recorded content. Contrary to the creation of 3D videos (*stereoscopic videos*) with a stereo camera, 2D-to-3D conversion techniques do not require costly and often bulky hardware. When directly recording 3D videos, a stereo camera captures two shifted views of the same scene. The shift between the two views at each point is referred to as *disparity* (and is inversely proportional to *depth*). On the other hand, conversion techniques are based on only one 2D view (*monoscopic video*) and generate disparities (and eventually a second view). The main advantage of conversion techniques is that the need for 3D content has not to be known before recording a video. Hence, existing monoscopic videos, such as older popular movies (e.g., Star Wars [178]) can be converted to 3D without re-recording them. This is especially interesting in the context of resolving the mentioned bottleneck concerning the availability of 3D content. Additionally, 2D-to-3D conversion enables the possibility to create 3D content from self-recorded 2D content.

The fundamental motivation of this thesis is to provide practical 2D-to-3D conversion algorithms that enable conversions of monoscopic image and video content. These algorithms generate new 3D content from 2D content and strive for high-quality depth impressions.

1.2 Problem Statement

This thesis addresses the problem of converting monoscopic (2D) videos to stereoscopic (3D) videos. Existing 2D-to-3D conversion techniques can be categorized according to the user involvement that is required during the conversion process [141]: *manual*, *fully-automatic* and *semi-automatic* 2D-to-3D conversion techniques. In *manual* (computer-aided) conversion techniques, 3D artists partition (*rotoscope*) each frame of a video into objects/surfaces by tracing their outlines and reconstruct the scene by assigning a depth model to each of these segments. This process is very labor-intensive (e.g., StereoD [143] converted 297000 frames in 60 weeks [137]), but can result in 3D content of high conversion quality. Thus, manual 2D-to-3D conversion is typically only used for high-quality 3DTV cinema productions.

Fully-automatic conversion techniques investigate *monocular depth cues* (i.e., that can be obtained from only one view) such as motion [105] or perspective [38] and perform the 3D conversion without any user interaction. Thus, automatic conversion techniques are based on the availability of monocular depth cues in the scenes that should be converted to 3D. In fact, automatic conversion techniques attempt to solve a highly ill-posed problem which is generally not invertible, namely recovering the scene geometry which was lost during the projection of the 3D scene onto the 2D image plane of the camera. The complexity of this task and the mentioned restrictions concerning the scenes typically cause a lower conversion quality [119,141]. Furthermore, due to the lack of user interaction, the results can be difficult to control.

To make this highly ill-posed problem more feasible, *semi-automatic* 2D-to-3D conversion techniques keep the user in the loop. Unlike labor-intensive manual techniques (e.g., [141,164]) or restrictive fully-automatic techniques (e.g., [38,105]), the goal of semi-automatic techniques is to generate 3D videos from arbitrary 2D videos with only little user interaction. The conversion quality of semi-atuomatic techniques ideally matches the conversion quality of manual ones. Thus, this group of conversion techniques can be seen as a compromise between fully-automatic and manual ones. Typically, semi-automatic 2D-to-3D conversion algorithms propagate sparse user-given disparity information to the remaining image or video, by using global optimization (e.g., [56, 163]), filtering techniques (e.g., [156]) or segmentation (e.g., [91, 165]). Various semi-automatic 2D-to-3D conversion approaches will be discussed in Chapter 2.

In general, 3D video content generation algorithms, including those for 2D-to-3D conversion, attempt to provide a *spatio-temporally coherent* 3D impression. While abrupt temporal disparity changes (flickering) can be distracting and should be avoided, smooth temporal disparity changes over time (e.g., object approaches camera) should be supported. Ideally, such a temporal disparity change is *perceptually coherent*. Specifically, the disparities of the object that moves in depth, should harmonize with those of nearby objects and occlusions should be taken into account. Two further problems in this context are *edge-sharpness mismatches* and the *cardboard effect*, which are caused by *over-smoothing* of disparity edges and not enough disparity variation within objects, respectively [15,158]. In particular, spatial disparity edges should be aligned with color edges and indicate an abrupt disparity transition at object borders. Ideally, disparity transitions inside objects are smooth. This provides objects with volume when watched in 3D and avoids that they appear unnaturally flat. Another crucial point in 3D content generation, especially when processing video material, is the *scalability* of a conversion algorithm. For practical usage, fast processing speeds and little required user-interactions are desirable.

In summary, semi-automatic 2D-to-3D-conversion techniques pose a promising alternative to high-quality, but labor-intensive manual techniques and restrictive, fully-automatic techniques. This thesis focuses on semi-automatic techniques, i.e., on generating 3D content based on sparse user input and with high conversion quality. In this context, common problems, such as spatio-temporally coherent disparity changes, over-smoothing and scalability, are addressed.

1.3 Contributions

This thesis suggests novel 2D-to-3D conversion algorithms that produce 3D content from monoscopic 2D content and sparse user-given disparity information. The provided 2D-to-3D conversion algorithms strive for high conversion quality, which includes spatio-temporal and perceptual coherence and to avoid common artifacts such as over-smoothing of disparities at object borders. The contributions of these algorithms rely on spatio-temporal video analysis techniques. In particular, we exploit the similarity of the problem of semi-automatic 2D-to-3D conversion and the problem of video (object-)segmentation. Semi-automatic conversion algorithms propagate sparse user-given disparities to the remaining pixels of the 2D image or 2D video. During this propagation similar disparities are assigned to pixels that are similar (e.g., in terms of color). Likewise, *segmentation* refers to the partitioning of scenes (e.g., a 2D video) into regions (*segments*) which are homogenous in a certain feature space (e.g., color). In this thesis, we investigate the joint segmentation and 2D-to-3D conversion of 2D videos. Contrary to existing 2D-to-3D conversion algorithms that are based on segmentation algorithms (e.g., [91,165]), in the conversion algorithms that are presented in this thesis, the role of segmentation goes beyond segment-wise disparity assignments. Instead, segmentation is viewed as an integral part of the conversion process. More precisely, the 2D-to-3D conversion's disparity propagation takes place during the segmentation process. This joint approach is, to the best of our knowledge, new for 2D-to-3D conversion.

The fusion of segmentation and 2D-to-3D conversion has several advantages. First, a joint conversion and segmentation process provides the opportunity to transfer the knowledge of the well-studied field of video segmentation to the less explored field of semi-automatic 2D-to-3D conversion. This also includes the usage of its efficient algorithms.

Second, spatio-temporal grouping information enables potentially more stable disparity propagation and provides additional information that can be exploited during the 2D-to-3D conversion process. This can, for example, be achieved by using more global similarity measures based on intermediate segmentation results instead of per-pixel measures. Additionally, spatio-temporal grouping information facilitates the distinction between areas where smooth disparity variations are desired (i.e., inside objects, to avoid the cardboard effect) and areas where smooth disparity variations should be prevented (i.e., at object borders to avoid edge-sharpness mismatches). This also extends to the temporal domain, where temporal grouping information can be used to perform smooth disparity interpolations over time or can be further analyzed to consider interactions between different video objects in the 2D-to-3D conversion process.

In our first algorithm, the 2D-to-3D conversion is performed jointly with a (automatic) graph-based video segmentation. In Chapter 4, our proposed joint segmentation and disparity propagation process assigns spatio-temporally neighboring pixels that are similar in color to the same disparity and to the same segment. In this way sparse user-provided disparity information is propagated over the entire 2D video. The advantage of our approach over previous work comes from the direct use of the segmentation technique during the propagation process. It preserves disparity edges at object borders, which is challenging for previous works. A subsequent selective filtering step enables disparity changes within spatio-temporal segments and over time and yields the final disparity map. The joint propagation/segmentation and filtering steps are further optimized in a scalable implementation on recent graphics architecture (*GPU*) that leverages modern multicore technology. With this implementation our 2D-to-3D conversion algorithm achieves runtimes of one frame per second (*fps*) for a video with a resolution of 640 × 480 pixels per frame. Thus, it scales to longer videos, i.e., it is still suitable when processing longer videos. Quantitative evaluations show that the proposed algorithm generates results of high conversion quality and outperforms a well-established semi-automatic 2D-to-3D conversion approach.

In our second contribution (Chapter 5), we introduce a fast (e.g., 250 fps for a video with a resolution of 620 × 360 pixels) approach for interactive video object segmentation and alpha matting. It is then extended to perform 2D-to-3D conversions. Our interactive segmentation and alpha matting approach is based on recent work that solves label-based optimization problems within a specific filtering scheme. We show that using this filtering scheme not only spatially, but also temporally enables our segmentation algorithm to achieve temporally coherent and flicker-free video object segmentations. The provided video object segmentations further include alpha mattes for the extraction of semi-transparent pixels at object borders that are likewise temporally coherent. The subsequently developed 2D-to-3D conversion algorithm considers the special requirements of 2D-to-3D conversion, including the extended pool of labels (multiple disparities instead of only fore- and background in the interactive segmentation) and smooth disparity variations within objects that are obtained with a simple disparity blending approach. In this context, a main contribution of our proposed 2D-to-3D conversion algorithm is the introduction of models for the temporal interpolation of disparities that exploit spatio-temporal grouping information. These temporal disparity models allow us not only to capture the disparity change of objects (i.e., segments) that move towards or farther away from the camera, but also to ensure that the disparities of these moving objects harmonize with those of nearby objects (i.e., are perceptually coherent). To achieve the latter, we analyze the motion in a video and perform a temporal disparity interpolation that is restricted by disparities of neighboring (occluding) objects. In this manner, we generate perceptually coherent disparity maps that contain smooth temporal disparity changes. Quantitative evaluations show that the proposed algorithms in Chapter 5, i.e., the interactive video object segmentation algorithm and the 2D-to-3D conversion algorithm, generate temporally coherent object segmentations and 2D-to-3D conversions. Comparisons of both algorithms with their respective related works indicate that our algorithms are competitive. In case of the 2D-to-3D conversion algorithm, the added functionality of capturing perceptually coherent disparity changes over time can improve our conversion results.

The final contribution of this thesis is a systematic evaluation of different semi-automatic 2D-to-3D conversion techniques including our newly proposed methods. Contrary to previous work, the evaluations in Chapter 6 are performed under consideration of the given user input. The given user inputs are sets of scribble-based annotations that follow different strategies regarding their placement within an object. They include minimalistic and effortless scribbling strategies as might be drawn from a system-unaware user as well as more advanced and labor-intensive scribbling strategies that take the content of the 2D images more specifically into account. We generate these annotations with an algorithm that simulates a user who takes on these different strategies and automatically places sparse scribbles in 2D images. The disparities of these sparse scribbles are taken from ground truth data which is provided with the 2D images that are used in our evaluation. We perform three experiments that focus on (i) the conversion accuracy, (ii) the robustness to small variations in the scribbles' spatial positions and (iii) the error tolerance of 2D-to-3D conversion techniques to inaccurate scribble inputs. In an additional fourth series of experiments, (iv) the ideal scribbling strategy under consideration of the given 2D content is investigated. This provides practical insights for the scribble-based annotation process. While none of the tested 2D-to-3D conversion techniques excels in combination with all tested scribbling strategies in all of our experiments, the graph-based algorithm from Chapter 4 together with a scribbling strategy that places scribbles close to object borders, throughout our experiments generated conversion results that are close to their reference solutions. Our evaluations further demonstrate that segmentation algorithms can improve the spatial coherence and reduce over-smoothing when being used in the 2D-to-3D conversion process.

1.4 Resulting Publications

The work presented in this thesis resulted in the following publications:

1. N. Brosch, C. Rhemann and M. Gelautz. “Segmentation-Based Depth Propagation in Videos“, Proceedings of the Austrian Association for Pattern Recognition (OAGM) Workshop, 2011. [24] Best student paper award.
2. N. Brosch, A. Hosni, C. Rhemann and M. Gelautz. “Spatio-temporally Coherent Interactive Video Object Segmentation via Efficient Filtering“, Proceedings of the Joint Pattern Recognition Symposium of the German Association for Pattern Recognition and the Austrian Association for Pattern Recognition (DAGM/OAGM), 2012. [22]
3. M. Ivancsics, N. Brosch and M. Gelautz. “Efficient Depth Propagation in Videos with GPU-Acceleration“, Proceedings of the International Conference on Visual Communications and Image Processing (VCIP), 2014. [67]
4. N. Brosch, T. Schausberger and M. Gelautz. “Towards Perceptually Coherent Depth Maps in 2D-to-3D Conversion“, Proceedings of the Electronic Imaging Conference (EI), 2016. [25]

Although their content is not explicitly included in this thesis, the work presented in this thesis additionally contributed to the following publications:

1. N. Brosch, A. Hosni, L. He, G. Ramachandran and M. Gelautz. “Content Generation For 3D Video/TV“, Journal of Elektrotechnik und Informationstechnik (e & i), 2011. [21]
2. S. Ghuffar, N. Brosch, N. Pfeifer and M. Gelautz. “Motion Segmentation in Videos from Time of Flight Cameras“, Proceedings of the International Conference on Systems, Signals and Image Processing (IWSSIP), 2012. [47]

3. S. Ghuffar, N. Brosch, N. Pfeifer and M. Gelautz. “Motion Estimation and Segmentation in Depth and Intensity Videos“, Journal of Integrated Computer-Aided Engineering (ICAE), 2014. [48]
4. N. Brosch, M. Nezveda, M. Gelautz and F. Seitner. “Efficient Quality Enhancement of Disparity Maps Based on Alpha Matting“, Proceedings of the Electronic Imaging Conference (EI), 2014. [23]
5. M. Nezveda, N. Brosch, F. Seitner and M. Gelautz. “Depth Map Post-Processing for Depth-Image-Based Rendering: A User Study“, Proceedings of the Electronic Imaging Conference (EI), 2014. [106]

1.5 Structure of the Work

This thesis is organized in seven chapters. The current chapter, Chapter 1, provided a general overview of the fundamental motivation, the problem statement and the contributions of this work. In the following, the remaining chapters of the thesis are briefly discussed and related to the publications that were listed before.

- Chapter 2 gives an overview of existing 3D content generation approaches. In this overview, special emphasis is put on semi-automatic 2D-to-3D conversion techniques, which are the focus of this thesis. Some parts of the text are take from our published paper [21].
- Chapter 3 discusses fundamental concepts of spatio-temporal video analysis that are relevant to 2D-to-3D conversion. In particular, we discuss automatic and interactive video segmentation techniques, interactive (disparity) annotation approaches and edge-aware interpolation techniques for image and video data. These concepts are exploited in the following chapters of the thesis. Chapter 3 further provides a brief overview of different evaluation benchmarks and evaluation strategies for 2D-to-3D conversion results.
- In Chapter 4, we introduce a novel approach for segmentation-based 2D-to-3D conversion for videos and discuss its efficient implementation using multicore technology. It exploits similarities of 2D-to-3D conversion and segmentation in a joint approach, which reduces over-smoothing at object borders. The main text of this chapter is a compilation of our published papers [24] and [67] with additional results and experiments. The efficient implementation of the segmentation-based 2D-to-3D conversion approach was performed in the master’s thesis [68] that was supervised by the author of this dissertation.
- In Chapter 5, we present a fast (250 fps for frames with a resolution of 620×360 per frame) interactive video object segmentation approach and extend it to perform 2D-to-3D conversions. The focus of the latter is to model perceptually coherent disparity changes over time. We aim to capture the changing disparity of objects that approach the camera or diverge from the camera and ensure that their disparities harmonize with disparities of nearby moving objects. In this chapter, parts of the text are taken from our published papers [22], [47, 48] and [25] with additional results and experiments. The 2D-to-3D conversion approach presented in this chapter was implemented in the master’s thesis [136] under supervision of the author of this dissertation.

- Chapter 6 provides a systematic evaluation of different semi-automatic 2D-to-3D conversion techniques, including our algorithms that are presented in this thesis. This evaluation is carried out on a dataset of 2D images with ground truth data and considers the given user input, i.e., different scribbling strategies. In this context, we investigate (i) the conversion accuracy, (ii) the robustness to small position shifts of scribbles, (iii) the error tolerance to inaccurate scribbles and (iv) the ideal scribbling strategy for specific 2D image regions. In this manner, we reveal strengths and weaknesses of the tested 2D-to-3D conversion techniques in conjunction with different scribbling strategies and provide practical insights concerning the scribble-based annotation process.
- Chapter 7 concludes this thesis and discusses possible future research questions in the field of semi-automatic 2D-to-3D conversion.

CHAPTER 2

State-of-the-Art of 2D-to-3D Conversion

This chapter gives a general overview of 3D content generation approaches that consider stereoscopic and monoscopic content as input. This overview and the subsequent discussion of prior work put emphasis on semi-automatic 2D-to-3D conversion which uses monoscopic content.

2.1 Principles of 3D Content Generation

The key idea behind the generation of 3D content is to provide viewers with an illusion of depth as seen in the real world. In general, the stereoscopic depth experience emerges when watching two slightly shifted views of the same scene, each with one eye. In 3D cinemas, the separation of the two views is accomplished by using special glasses that direct each presented view to the corresponding eye. The human visual system processes these images yielding a depth perception by exploitation of the geometric differences (denoted as *disparities*) between the two images.

In general, there are several approaches for generating 3D content that can be viewed on stereoscopic displays. An obvious approach would be to use a stereo camera when recording a video. In principle, the acquired *stereo video* (consisting of two synchronized video streams) could be displayed directly on a suitable 3D display. In many cases, however, further processing steps are required to adjust the stereo content to different types of displays and viewing distances. For example, a stereo video that had been recorded for display on a 3D cinema screen would – in its original form – yield an uncomfortable viewing experience on a small-size mobile 3D display. The viewing freedom, encompassing the viewing distance as well as the position of the viewers, plays an important role in determining the number of views required. Hence, a key requirement for 3D content adaptation is the generation of *novel views* that simulate virtual cameras that were not available during the original video acquisition. A particular need for novel views comes up in the context of (multi-user) *autostereoscopic displays*, which rely on multiple views to enable glass-free 3D viewing. The related adjusting procedures typically require the computation of a depth or *disparity map* as intermediate product. Section 2.1.1 briefly discusses the fundamentals of the stereoscopic computation of disparity maps from stereoscopic videos.

In many applications, the disparity map is computed from two views of the same scene using stereo vision approaches. However, if only one view is available, such as for existing monocular videos, 2D-to-3D conversion approaches can be considered as an alternative solution. The conversion can be performed manually by assigning disparities[1] to each pixel of a video, semi-automatically by propagating sparse user-given disparities over the entire video, or fully-automatically by investigating monocular depth cues [119, 141]. Like in the stereo case, disparity maps that were computed by these approaches can be adjusted to different types of displays and utilized for novel view generation. Section 2.1.2 provides a general overview of different types of 2D-to-3D conversion approaches and their principles. In Section 2.2, special emphasis is put on the state-of-the-art of semi-automatic 2D-to-3D conversion, which is the focus of this thesis.

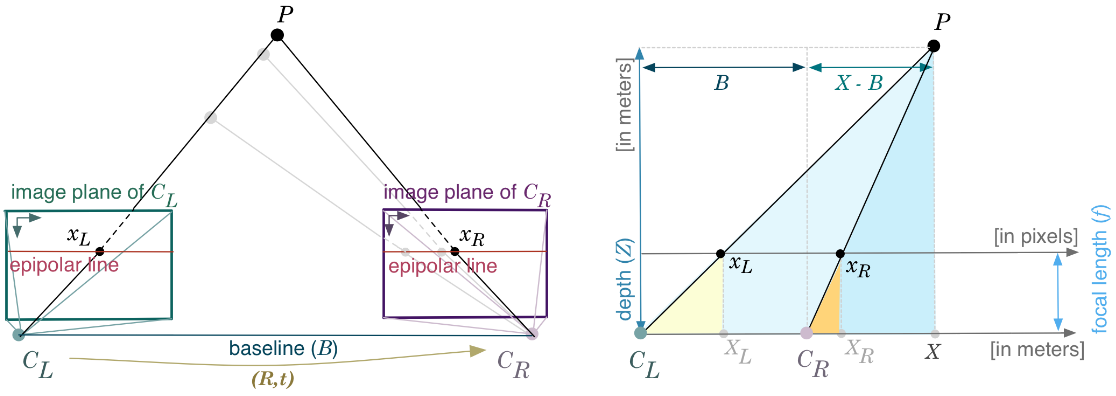
Additional approaches for generating content for 3D viewing may be based on the availability of a 3D model, from which disparities and multiple views can be rendered, e.g., with 3D computer graphic software such as Blender [10]. Furthermore, depth can be captured directly with special depth sensors and scanners such as Microsoft Kinect [103].

2.1.1 3D from Stereoscopic Data

Given multiple, e.g., two, images that were taken from slightly shifted viewpoints of the same scene, a 3D model of the scene can be estimated by determining pixel correspondences between these images (*stereo correspondence problem* or *stereo matching problem* [146]). The shift in position of these corresponding pixels, the *disparity*, directly relates to the depth of a scene. This relationship can be derived from the *standard rectified stereo geometry* [146]. In particular, Figure 2.1 *a)* illustrates the two images within the standard rectified camera setup captured by two cameras. The cameras C_L and C_R are connected by a horizontal line, which is called the *baseline*. C_L and C_R are *calibrated*, i.e., the transformation (R, t) of the camera coordinate system of one camera to the other camera is known, and *rectified*, i.e., the image planes of C_L and C_R lie in a common plane that is parallel to the baseline. (For more details concerning camera calibration and rectification interested readers are referred to [146].) When capturing a 3D scene using the camera setup in Figure 2.1 *a)*, the 3D point P is projected into the points x_L and x_R on the image planes of C_L and C_R. During this process, C_L, C_R, P, x_L and x_R span a plane, the *epipolar plane* [146]. Due to the rectified camera setup, matching points in one image plane (e.g., x_L in the left view and x_R in the right view) must lie on a particular horizontal line that intersects the epipolar plane with the image plane, i.e., the corresponding *epipolar line*, in the other view. This restriction concerning the location of corresponding points provides an advantageous *epipolar*

---

[1] The term disparity was introduced to describe position differences in stereoscopic conditions and refers to the field of stereo vision [101, 146]. However, semi-automatic 2D-to-3D conversion algorithms (e.g., [56, 117, 163]) use equivalent values for their depth information. As stereo disparity, it encodes the closeness of pixels to the camera (i.e., is large in the fore- and low in the background) and can be used to generate novel views by shifting pixel positions accordingly. The depth information used in 2D-to-3D conversion algorithms is typically either, exactly as disparity, given in terms of position shifts that can be used directly to generate novel views (e.g., [56, 163]) or as normalized values $\in [0, 1]$ that have to be scaled prior to that (e.g., [117]). It is not given in meters as scene depths. In the 2D-to-3D conversion literature the terms disparity and depth are used both. As in [56, 163], in this thesis we use the term disparity when referring to depth information in the context of 2D-to-3D conversion.



a) b)

Figure 2.1: Standard rectified epipolar geometry. *a)* The horizontally neighboring cameras C_L and C_R are *rectified*, i.e., the image planes lie in a common plane that is parallel to the *baseline*. Matching points in one view lie on a horizontal line, i.e., the *epipolar line*, in the other view. C_L and C_R are *calibrated*, i.e., the transformation (R,t) between their camera coordinate systems is known. *b)* In this setup the *disparity* $d_x = x_R - x_L$ and depth Z of a 3D point P with coordinates (X,Y,Z) and its projections in the image planes with x_L and x_R are related via similar triangles (i.e., (x_L, X_L, C_L), (P, X, C_L), (x_R, X_R, C_R) and (P, X, C_R)). [11, 146]

constraint, which reduces the search space for corresponding pixels in the left and the right view to their horizontal scan-lines.

Having identified two corresponding pixels, e.g., x_L and x_R, which are located in the left and the right view, the *disparity* d_x can be determined by their horizontal position shift, i.e., $d_x = x_L - x_R$. As shown in Figure 2.1 *b)*, the disparity d_x is inversely proportional to the depth Z of a scene. They are related via the similar triangles (x_L, X_L, C_L), (P, X, C_L), (x_R, X_R, C_R) and (P, X, C_R), which leads to the following equation:

$$Z = f\frac{B}{d_x}. \tag{2.1}$$

Here, f is the focal length (in pixels) and B is the baseline between C_L and C_R. Thus, the task of estimating depth from a stereo image pair is reduced to the task of estimating the disparity of each pixel (*disparity map)*. In the context of the standard rectified stereo geometry, the process of stereo matching can be solved by finding corresponding (*matching*) pixels in horizontal scan-lines of the left and the right view. A stereo matching algorithm's foundation to find these correspondences is the definition of a measure that expresses the quality (or *matching costs*) of a potential match between a pixel of the left and a pixel of the right view. This is typically done by measuring the similarity, e.g., the color difference, of these pixels [134, 146]. While high similarities indicate good matches, large matching costs point to a low matching quality. As a second step, these costs can be aggregated. The final pixel correspondences (and thus the resulting disparity map) are determined in terms of an optimization that is defined over the previously computed costs. This optimization can be performed *locally* (e.g., [125, 126]), by selecting the disparities with the lowest costs according to a local pixel neighborhood or *globally* (e.g., [12–14]), by minimizing a global cost function that seeks a disparity map that is optimal according to all pixels. Finding the correct match for a pixel is, e.g., hindered by untextured regions or repetitive patterns (multiple pixels in one view potentially can be matched to one pixel in the other view), illumination differences (low pixel similarities at corresponding pixels), and occlusions (no correspondence available). Obviously, stereoscopic 3D content generation requires stereoscopic source material, i.e., two views. Thus, contrary to 3D content generation by 2D-to-3D conversion techniques, the need for 3D content (and for a second view) has to be known before recording a scene.

2.1.2 3D from Monoscopic Data

Given a single view, e.g., a monoscopic video captured by a single camera, the goal of 2D-to-3D conversion is to generate a disparity map and eventually a second view of the same scene. Thus, compared to the problem of stereoscopic 3D content generation (Section 2.1.1), less geometric information is given in the case of monoscopic 3D content generation. To compensate for this missing information, *manual* and *semi-automatic* 2D-to-3D conversion techniques incorporate user input. The *(computer-aided) manual* workflow of manual techniques consists of (1) rotoscoping, i.e., partitioning of each frame into its objects/surfaces by tracing their outlines [1], and (2) reconstruction of the scene by assigning a 3D model to each object/surface [141, 164, 175]. Concerning (1), segmentation techniques can assist 3D artists in converting 2D content. Segmentation techniques group pixels into regions that are homogenous in a certain feature space (e.g., color) and enable the extraction of objects from image and video content (e.g., [5,88,97,122,127]). For example, in [164] an interactive image segmentation technique [127] extends a rough user-provided selection of an object in a frame of a 2D video. This segmentation of the marked object can be further refined by the user and, as an initialization of a spatio-temporal object segmentation, transferred to the next frame. Concerning (2), depth templates (e.g., planes or spheres [164]) can be used to build and assign a 3D model to a segment (contrary to assigning each pixel separately) or automatic 2D-to-3D conversion techniques can be employed to generate an initial depth reconstruction that is further corrected by the 3D artist [164, 175]. Although using techniques that assist 3D artists in the manual conversion process, it is still labor-intensive (e.g., StereoD [143] converted 297000 frames in 60 weeks [137]). Thus, manual 2D-to-3D conversion is typically only used for high-quality cinema productions.

Additional efforts in reducing and omitting the manual labor of 3D artists directly address the disparity assignments to pixels. *Fully-automatic* 2D-to-3D conversion algorithms often investigate monocular depth cues (i.e., depth cues that can be obtained from only one view) to infer 3D models for a single 2D image or 2D video. In the following, we briefly discuss the depth extraction from a few of such depth cues. For a more detailed review interested readers are referred to corresponding literature surveys in [174] and [119]. In several fully-automatic 2D-to-3D conversion algorithms, the principle of *structure from motion* has been exploited (e.g., [65,92,105,111]). In this context, the key idea is that in the special case of static scenes with a moving camera, foreground objects typically move faster than objects that are in the background. Thus, from one frame to the next the shift in position of foreground objects should be larger than for background objects. Consequently, frame-to-frame pixel correspondences that are determined by motion estimation techniques are used to convert a video into 3D (e.g., [65, 105, 111]). In fact, this key idea is similar to stereoscopic 3D content generation (Section 2.1.1) in which

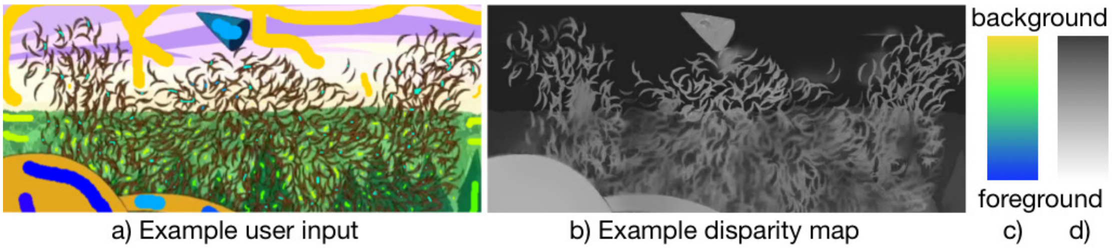


Figure 2.2: Example user input and disparity map. *a)* 2D image with user input and *b)* a corresponding disparity map. The *hue* of the *disparity scribbles* in *a)* encodes disparity *c)*. In the disparity map *b)* the disparity of each pixel, is encoded by *gray* intensities *d)*, whereas bright pixels are located in the foreground and dark pixels are in the background.

horizontal displacements between two 2D images specify a disparity map. However, in the context of structure from motion the displacements are motion vectors that describe horizontal as well as vertical movement in consecutive frames. For example, the automatic approach in [65] converts a monoscopic video in real-time by interpreting the motions' magnitudes as disparities (and, thus, essentially treats consecutive frames as a stereo pair). However, while this simple approximation may hold for static scenes with camera movement (epipolar conditions), it is not necessarily sufficient for arbitrary camera movement, arbitrary object movement or a lack of movement [174]. Additionally, fully-automatic 2D-to-3D conversion algorithms that are based on learning (e.g., [58, 71]) or that investigate other monocular depth cues and principles have been proposed (e.g., [2, 38, 54, 57, 70, 74, 147]). For example, when exploiting the principle of *shape from focus/defocus* (e.g., [2, 54]), the amount of blur in an image is analyzed to assign sharp (focused) image regions to larger disparities than blurred (defocused) image regions. To distinguish fore- and background objects in images, foreground objects have to be focused [174]. Another example, are fully-automatic 2D-to-3D conversion algorithms that are based on the principle of *shape from geometric cues*. Such algorithms (e.g., [38, 57]) extract geometric scene properties, such as *vanishing points* (i.e., distant points in which parallel lines converge due to linear perspective), to identify fore- and background regions accordingly. In general, fully-automatic 2D-to-3D conversion systems depend on the availability of suitable monocular depth cues in the scenes and their results are difficult to control. Despite the large amount of research [119, 174] in this field, it was pointed out [73, 174] that fully automatic generation of 3D content from 2D content is still an open challenge.

Contrary to labor-intensive manual conversion techniques or restrictive fully-automatic conversion techniques, *semi-automatic* 2D-to-3D conversion techniques propagate sparse user-given disparities to the remaining pixels of an image or video. In Section 2.2 prior works on semi-automatic 2D-to-3D conversion, which is the focus of this thesis, are discussed.

2.2 Prior Work on Semi-automatic 2D-to-3D Conversion

Semi-automatic 2D-to-3D conversion techniques are based on sparse user input. Typically, users mark pixels in 2D images or in keyframes of 2D videos by drawing into them. The disparity of the marked pixels is encoded by the color that was chosen by the user to mark the pixels (Figure 2.2). Although such a disparity assignment is not necessarily accurate in terms of absolute disparity values, an assignment that follows a perceptually consistent depth order is compliant with the human visual system and sufficient for 2D-to-3D conversion [56, 73, 75][2].

The sparse user-given disparities are used to estimate the disparities of all remaining pixels of a monoscopic image or video. To this end, semi-automatic 2D-to-3D conversion approaches assume that the searched disparity map is piecewise smooth with disparity edges at object borders (being indicated by color edges). The key idea of this assumption is that neighboring pixels with similar colors might belong to the same object and, hence, are likely to have similar disparities (e.g., in Figure 2.2, violet sky pixels should be assigned to similar disparities). Contrary, pixels that are separated by color edges are more likely to belong to different objects and are less constrained to have similar disparities, i.e., can also be assigned to different disparities (e.g., in Figure 2.2, green trees should be assigned to different disparities than orange cartoon character). Another common assumption of 2D-to-3D conversion approaches regards the spatial closeness between two pixels, i.e., spatial close pixels are more likely to be assigned to similar disparities than pixels that are spatially more distant. Thus, based on these assumptions initially unknown disparities can be derived from close-by pixels with given disparities and similar color.

The main difference between different semi-automatic 2D-to-3D conversion approaches proposed in the literature is the implementation of the propagation step, while the basic assumptions essentially remain the same. For example, the propagation can be performed by means of global optimization (e.g., [56]), by local filtering techniques (e.g., [156]) or by assignments of disparities to segments (e.g., [165]). In the following, we give an overview of these different propagation procedures and discuss state-of-the-art semi-automatic 2D-to-3D conversion approaches.

2.2.1 Optimization-based Techniques

Optimization-based conversion techniques express the propagation in terms of a global energy (*cost*) function which is solved by minimization techniques. This function typically consists of a data term and a smoothness term. The data term relates the user-given disparities to the user-marked pixels. The smoothness term – the core component of the propagation process – is responsible for the given disparities' dense propagation to the remaining pixels. It implements the key assumption of semi-automatic 2D-to-3D conversion by assigning lower costs to a disparity propagation between neighboring pixels with similar colors than to a propagation between neighboring pixels with large color differences. In this manner, the smoothness term regulates the contribution of the user-given disparities to the final disparity assignment of each pixel. The final disparity map, in which neighboring pixels with similar colors also have similar disparities, is determined according to the global minimum of the cost function.

As representative examples of optimization-based techniques, the semi-automatic 2D-to-3D conversion approaches proposed in [56, 159] and [163] exploit a global edge-aware interpolation technique that was previously used in the context of video colorization [84]. In particular, they propagate sparsely given disparities to each pixel i through approximating the solution of a linear system of equations by minimizing the sum of its squared errors (i.e., using *least squares*

---

[2] In Chapter 3 the user input is discussed further.

estimation [84, 146]). In the well-established 2D-to-3D conversion approach proposed in [56], the given disparities comprise user-given disparity D_i from scribbles that were drawn in the first and the last frame of a video shot (to support disparity changes over time) and disparity assignments predicted at *anchor points* A_i. These anchor points are located in between the first and the last frame to support conversion of videos with significant motion. They are essentially predicted by training a *support vector machine* [120] based on the user-given disparities and applying it on the entire video shot. The mentioned system of equations contains per-pixel sets of equations that each belong to one of four types, i.e., (1) spatial smoothness constraints (e.g., Equation 2.2), (2) temporal smoothness constraints (e.g., Equation 2.3), (3) anchor point constraints (e.g., Equation 2.4) and (4) scribble constraints (e.g., Equation 2.5), in the form of:

$$c_1 W_{Ei}(d_i - d_j) = 0, \tag{2.2}$$

$$c_2 W_{Mi}(d_i - d_k) = 0, \tag{2.3}$$

$$c_3 d_i = c_3 A_i \tag{2.4}$$

$$c_4 d_i = c_4 D_i. \tag{2.5}$$

Here, c_1, c_2, c_3 and c_4 are constant parameters that control the contribution of the respective sets of equations. i and j are neighbor pixels with (unknown) disparities d_i and d_j. Sets of equations of type (1) and (2) belong to the smoothness term, while (3) and (4) belong to the data term of the corresponding global energy function. In particular, sets of equations of the type (1) take on the form of Equation 2.2 and enforce the key assumption of semi-automatic 2D-to-3D conversion. They constrain the propagation of disparity to neighboring pixels by the image gradient W_{Ei} (i.e., image edges in the grayscale version of the image). Thus, the propagation models smooth disparity changes in regions of similar colors (or intensities) and assumes disparity edges at edges in an image (or frame of a video). According to sets of equations of type (2) temporal disparity propagations among temporal neighboring pixels are constrained by the type of observed motion (i.e., using the weight W_{Mi} in Equation 2.3). In particular, the disparities of two temporally neighboring pixels with predominantly vertical motion are less constrained to be similar (lower W_{Mi}) than the disparities of two temporal neighboring pixels with predominantly horizontal motion (larger W_{Mi}). This relates to the observation that a change in depth (and, thus, disparity) is more often observed in conjunction with vertical motion than in conjunction with horizontal motion [56]. Sets of equations of type (3) relate predicted disparities A_i to the locations of their anchor points. Sets of equations of type (4) relate the user-given disparities D_i in the first and the last frame to the user-marked pixels. When solving the described system of equations using least squares estimation, the result is a smooth and plausible disparity map for each frame of a video [56] (e.g., Figure 2.3 *b)*). However, this approach struggles with two main problems, over-smoothing in its disparity maps and the scalability of the solution. Concerning the latter, the complexity of the solution increases quadratically with the number of pixels in a video (i.e., $N \times N$ sized system of equations, N denoting the number of pixels in the video). To accelerate the runtime and enable the conversion of large videos, the authors suggest to convert a low resolution version of a video (reduced by a factor of four in each dimension [56]) and to up-sample it with a *joint bilateral filter technique* [78]. The second main problem, over-smoothed disparity maps (e.g., Figure 2.3 *b)*, *red arrows*), is caused by the quadratic smoothness constraints. These quadratic smoothness constraints apply relatively low costs to a disparity propagation

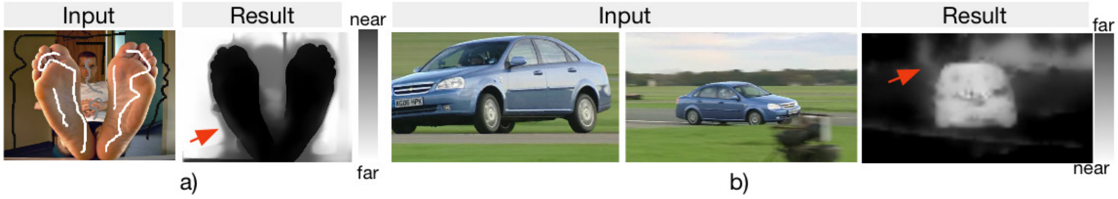


Figure 2.3: Examples of optimization-based disparity propagation. The examples are taken from *a)* [163] and *b)* [56] and show input images (*left*) and disparity maps (*right*). In *a)* the input is a 2D image with disparity scribbles. In *b)* the input is a 2D video, where the first and last frame are shown. The disparity map for an intermediate frame is given. (The scribbles and the 2D frame that correspond to this disparity map are not given by the authors.) In *a)* and *b)* disparity is encoded by gray values (see *legends*). *Red arrows* indicate over-smoothed areas.

between dissimilar pixels (in terms of color and spatial closeness) compared to the costs of a propagation between similar pixels. Thus, they tend to over-smooth disparity maps at color edges in the scenes. This is especially true in combination with improper image edge definitions in [56], that were pointed out by [117], and in the presence of fuzzy object borders (e.g., unsharp color edges or motion blur). Furthermore, in the context of optimization-based approaches, such as [163] and [56], insufficient propagations to pixels that are far away from the user input and over-smoothing for very sparse user input were reported [30, 66]. Note that the above mentioned acceleration strategy based on a reduction in video resolution can as well add over-smoothing to the disparities maps [23, 45, 69]. Moreover, erroneous disparity predictions at anchor points can introduce errors in the resulting disparity maps [117].

In the same line of work as [56, 84], Wang et al. [163] solve a similar system of equations with the same interpolation technique as in [56], but applied iteratively from a coarse-to-fine resolution of a given image or video. Contrary to [56], the disparity maps are estimated indirectly by expressing the energy function in terms of *image warps*, i.e., mapping operations according to pixel disparities that generate a novel view. This means, the cost function is solved for a new left and a new right view, which implicitly generates a disparity map as additional output. In this manner, multiple steps in the 2D-to-3D conversion process, i.e., disparity map generation, its adjustment to a comfortable disparity range and novel view generation, are addressed simultaneously. The data term of their cost function enforces image warps to be consistent with disparity maps and relates all pixels to the user-given disparities. The data term constrains all pixels without given disparities by spatial (and temporal) distance to the pixels with user-given disparities in the original image or video. Similar to [56], the (spatio-temporal) smoothness constraint in [163] is based on color image edges, but additionally weighted by *visual saliency* [49] to further reduce disparity over-smoothing in salient image regions. Visual salience is a measure for the human attention towards specific image regions (e.g., distinctive foreground objects on uniformly colored background) and, thus, (automatically) specifies regions of special interest. The main motivation for its incorporation in the 2D-to-3D conversion process is that especially in these regions, artifacts in novel views are noticeable and should be avoided. To accelerate the conversion and increase the scalability of the approach, their linear system of equations is solved with a multi-scale GPU implementation (e.g., [163]'s GPU implementation requires approximately one minute per four frames with HD resolution). However, as in [56] the quadratic smoothness constraints may still cause some over-smoothing of disparity edges at object borders.

Rzeszutek et al. [130] employ a simpler approach to semi-automatically convert 2D videos to 3D. It is based on their 2D-to-3D image conversion approach in [118]. The authors track user-given disparity scribbles from the first frame to the next frames. Subsequently they solve a system of equations that constrains the disparity propagation only by color differences of spatio-temporally neighboring pixels. Compared to the previously discussed conversion approaches, the solution is obtained by an equivalent global optimization procedure [118], i.e., *random walks* [51]. In the same year (2011) as our first publication in the field of semi-automatic 2D-to-3D conversion (i.e., Chapter 4), Rzeszutek et al.'s approach was improved by exploiting segmentation algorithms. Although Rzeszutek et al.'s algorithm is quite different compared to ours, it takes on a similar view. Specifically, to reduce the mentioned over-smoothing effect at object borders, which is also present in the equivalent solution, the authors incorporated an interactive object segmentation algorithm in the optimization process in [41, 116–118, 169]. In these approaches, as a first step each user-given disparity scribble is used to interactively segment an object from an image and assign it with corresponding scribble disparities as an initialization. These initial disparities are, subsequently, used to additionally constrain the disparity propagation between neighboring pixels. In the system of equations to be solved, the disparity propagation from the user input to the remaining pixels is not only constrained by color similarities of neighboring pixels in the 2D image, but also by the disparity similarities of neighboring pixels in the initial disparity map. In [41] the disparity difference is additionally weighted by color edge information to further reduce the over-smoothing effect.

Following the same idea as [118], in [66] a different segmentation process, i.e., [4], and a different optimization technique, are used to construct 3D models for 2D images from scribbles that indicate a depth order. This approach additionally computes transparencies close to borders of foreground objects, which can be used when generating 3D models or novel views.

In [116] the original 2D-to-3D image conversion approach [118] is extended to convert videos by employing an improved tracking method compared to [130]. The additional usage of bounding boxes in the tracking process allows the assigned disparities to automatically change with the size of those bounding boxes. The underlying idea is that objects become larger when they move closer to the camera and that the bounding boxes of the scribbles behave similarly. To reduce the runtimes and the memory footprint of the conversion, long videos are split into overlapping sub-videos that are processed independently.

Another 2D-to-3D conversion approach that should be mentioned is [92]. In [92], user-given disparity information is enriched with additional information, including disparity estimates from motion analysis. Similarly to the approaches discussed above, the authors solve a quadratic optimization problem to obtain a disparity map. Contrary to these approaches, the approach in [92] supports inequality constraints (e.g., one object is in front of another object) that are provided by the user. Thus, it addresses temporal disparity changes and perceptual coherence.

2.2.2 Filter-based Techniques

Filter-based conversion techniques (e.g., [28, 90, 94, 156]) typically assume that the disparities of an entire keyframe are given. The key idea of these methods is to locally propagate the

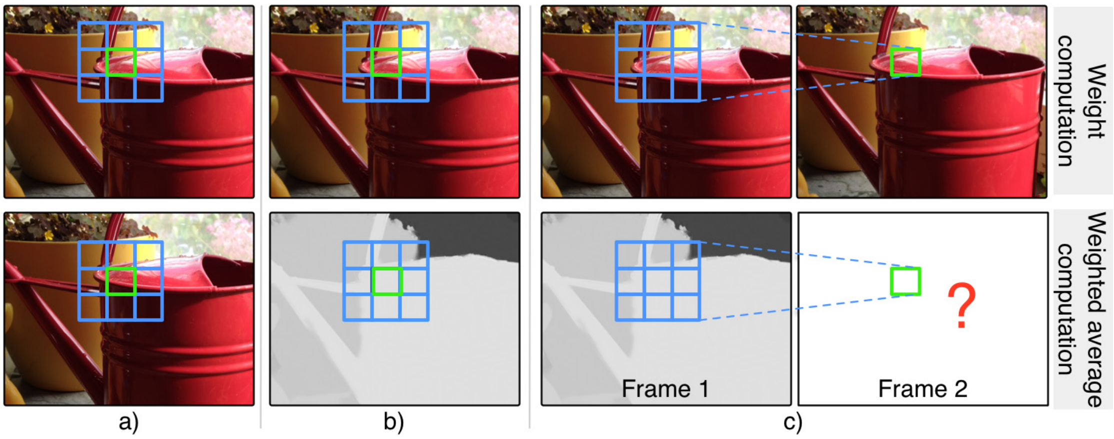


Figure 2.4: Illustration of filter-based disparity propagation. *a)* Several (linear) filtering techniques update the center pixel (*green*) by weighted average that is computed within filter window (*blue*). The weight computation is performed in the same image as the computation of the average. *b) Joint* (or *cross*) *filtering techniques*: The filter weights and the weighted average are computed in different images (e.g., color image and disparity map, respectively). *c) Filter-based propagation*: The center pixel is located in one frame (e.g., *frame 2*) while the remaining pixels of the filter window are located in another frame (e.g., *frame 1*). The center pixel in *frame 2* is updated with the weighted average computed from the remaining window pixels in *frame 1*.

given disparities from one frame to the next frame with unknown disparities by using *joint* (or *cross*) *filtering techniques*, which are briefly explained in the following. Several image filtering techniques (e.g., the *bilateral filter* [153]) place a window of fixed size at each pixel in an image and update this pixel by a weighted average according to its neighboring pixels within the window (Figure 2.4 *a)*). *Joint filtering techniques* (e.g., the *joint bilateral filter* [115] or [59]) decouple the weight computation from the average computation. To this end, the filter window is placed at the same pixel position in two corresponding images, e.g., a disparity map and a color image (Figure 2.4 *b)*). The averaging is then performed in one image, e.g., the disparity map, while the weights for the averaging operation are determined in another corresponding image, e.g., the color image. The weights regulate the influence of pixels in the window on the computed average. They can be computed based on color similarities of each pixel in the window to the center pixel. Thus, when using a joint filtering technique, predominantly disparities of pixels with similar colors are averaged, while disparities of pixels with differing colors are largely suppressed. Due to this behavior, joint averaging operations locally implement the key assumption of semi-automatic 2D-to-3D conversion and can be used to average known disparities within a window according to color similarities to center pixels with unknown disparities. As mentioned above, in the context of filter-based 2D-to-3D conversion, disparities are typically given for one frame and are propagated to a following frame. Thus, to perform a filter-based propagation the center pixel of a filter window is located in a frame without user given disparities, while the remaining pixels of the filter window are located in a frame with given disparities (Figure 2.4 *c)*). The disparity of the center pixel is assigned according the computed average.

The main advantage of filter-based conversion techniques is the low complexity of their approach. Since they propagate disparities locally, they can potentially exhibit short runtimes and low memory footprints. This is especially true in comparison with conversion techniques that are based on global optimization techniques (Section 2.2.1), which face large (global) and complex optimization problems. However, the efficiency of these filter-based conversion techniques comes at some cost. More precisely, these conversion techniques struggle with three main problems [156]: (1) Color ambiguities, (2) new colors, and (3) temporal error accumulation. First, local color similarities at different disparities within a window cause the disparities of similar objects to be averaged together (e.g., at unsharp color edges). Second, if the color of the center pixel is not present in its remaining window, the new averaged disparity will be dominated by the pixels which are most similar to this new color and might belong to a different object. This problem, for example, arises if a new object enters the scene or if the window size is too small to capture the motion of a scene. Thus, in this context, windows with large sizes are beneficial. Note, however, that with increasing the size of a window, the probability of local color ambiguities, i.e., (1), increases. Depending on the used filtering technique, larger windows as well increase the runtime of the propagation process [59]. Third, filter-based conversion techniques typically perform the conversion in a frame-to-frame manner, using the propagation result from a frame as given disparities to convert the subsequent frame. Thus, erroneous disparities, e.g., due (1) or (2), are further accumulated to subsequent frames. This leads to decreases in quality of the disparity map with the distance from the user-marked keyframe [156]. These problems usually result in over-smoothed object borders in the generated disparity maps.

The authors of the popular filter-based 2D-to-3D conversion algorithm [156] adopt a joint bilateral filter [115] to perform the propagation that was described above and obtain an initial disparity map. They attempt to correct errors in this initial disparity map that are caused by (1) and (2) by block-based motion compensation. To this end, estimated, initial disparities are overwritten by given disparities from the previous frame that are connected by motion vectors. The used motion vectors are estimated between the given disparity map and the initial disparity map. Likewise, in [31] motion vectors are estimated between a smoothed version of the given disparity map and the initial disparity map. However, estimating motion vectors from disparity maps is not beneficial, since they do not capture image texture, which is crucial for motion estimation methods [7]. Although the performed motion compensation suppresses the mentioned over-smoothing effect, inaccurate motion estimation can introduce visible artifacts (e.g., *blocking artifacts* [28]) due to erroneously overwritten disparities. Furthermore, since color ambiguities (i.e., untextured regions) and new colors also affect motion estimation [7], the mentioned problems largely remain. The additionally required motion estimation adds computational complexity to the initial low complex filter-based 2D-to-3D conversion approach. Several authors (e.g., [28,64,93,94]) as well attempt to improve the propagation result by refining initial results or modifying the basic algorithm of [156]. For example, Cao et al. [28] present an improved filtering strategy, in which the filter window is shifted according to motion vectors that are estimated from color frames. To reduce the accumulation of erroneous disparities, i.e., (3), and enable disparity changes over time, a bi-directional propagation strategy is applied to determine the initial disparity map. In particular, frames without user-given disparities are directly based on the user-given disparities of the two nearest frames, instead on the estimated disparities of their previous frames as in [156]. This means, the propagation is performed separately with respect to a former and with respect to a later frame with user-given disparities. The final disparity map is a weighted (distance to frames with user input) average of both disparity maps. Cao et al. [28] report runtimes of 18 seconds per frame with a resolution of 720×576 pixels for a CPU implementation. Lie at al. [93]'s improvement over the basic approach is to post-filter the motion compensated disparity maps with a joint filtering technique that is based on both color and disparity differences. This strategy can smooth out errors that were caused by inaccurate motion compensation, but results in over-smoothed object borders that are further accumulated over time and eventually lead to over-smoothed disparity maps [31]. In some similarity, in [64] the refinement step of an initial disparity map is performed in the context of a global optimization scheme. In [94], a fixed threshold for color differences along motion vectors is additionally used to identify potentially erroneous motion vectors that caused erroneously propagated disparities in [93]. The identified disparities are corrected according to disparities of spatially neighboring pixels, disparities that were propagated from only one annotated frame or by repeating the propagation process with different parameters.

Contrary to the previously discussed approaches, in [90, 173] user-given disparities from annotated frames are directly copied to frames in which the disparities are unknown by following motion vectors between them. The filtering technique from [156] is only applied in regions where no disparities could be copied to (e.g., due to occlusion or disocclusion). Likewise, motion compensation is only applied in regions without disparities, to suppress artifacts such as over-smoothed disparity edges. The bi-directional propagation strategy from [28] determines the final disparities and captures linear disparity changes over time. Similarly to [90], in [173] the copied disparities in each frame are refined, but unlike [90] within a global optimization procedure that aims to assign similar disparities to similar (color) neighbor pixels. While the former approach, i.e., [90], and some of the approaches discussed above, support disparity changes over time, the perceptual coherence of disparity changes is not addressed.

In [82, 129] an alternative approach is employed to convert 2D content to 3D, namely the approximation of global optimization with local filtering operations. Here, sparsely provided disparities in videos are propagated by iteratively applying an edge-aware filtering operation [46] spatially and temporally. The used filter exhibits interactive runtimes (i.e., a one megapixel image is filtered within milliseconds using a GPU implementation [46]). To improve the temporal coherence of the results, temporal filtering is performed by following predetermined motion vectors from frame to frame. While the original work in [82] only mentions that their optimization scheme can be applied for 2D-to-3D conversion of images and videos, the authors of [129] adopt the idea for this application and focus on images. They propagate sparse, automatically estimated disparities given for a 2D image to obtain dense disparity maps. In this context, the authors of [129] point out that filter-based conversion techniques are inferior to optimization-based conversion techniques. Typically, they are sensitive to their parameters, which have to be carefully adjusted dependent on the 2D content and density of the given disparities. It is possible that disparities of some image areas remain unknown due to numerical issues. In particular, larger filter windows typically tend to over-smooth the results, whereas small filter windows increase the number of pixels with unknown disparities. To alleviate this behavior for smaller filter windows, an iterative refinement procedure fills in such areas after each filter iteration in [129].

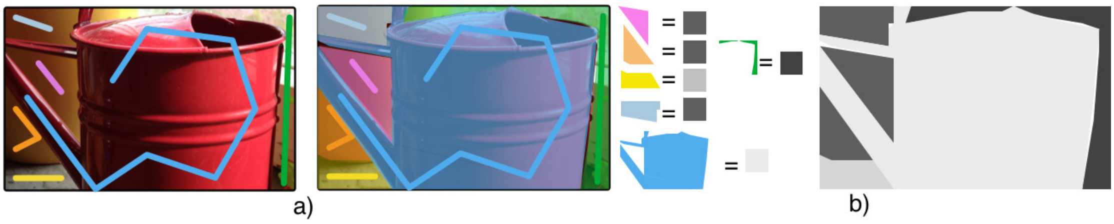


Figure 2.5: Illustration of segmentation-based disparity propagation. This type of propagation typically comprises two steps: *a)* Interactive object segmentation according to user input (e.g., colored scribbles, *left*), segments (*right*) are *color-coded* with the same color as the scribbles (but *semi-transparent*). *b)* Assignment of disparities (*bright*: front, *dark*: back) to these segments.

2.2.3 Segmentation-based Techniques

Closely related to manual 2D-to-3D conversion approaches, segmentation-based techniques (e.g., [29, 44, 91, 165]) typically comprise two main steps: (1) segmentation of the 2D content into regions that have similar colors (Figure 2.5 *a)*), and (2) assignment of disparities to these segments (Figure 2.5 *b)*). For example, the algorithms in [91, 165] follow these two basic steps. First, users interactively extract foreground objects from keyframes by using a scribble-based image cutout tool [89] and in each keyframe assign a single disparity to each extracted foreground object. Additionally, each pixel of the remaining stationary background is assigned to a disparity. Subsequently, the extracted foreground objects are tracked, which results in spatio-temoral segments that extend across frames. Second, the disparity of each segment is given by its disparity assignment in keyframes. To enable temporal disparity changes within segments, a segment's disparity can be determined by interpolating its disparity assignments performed in the two nearest keyframes [165]. The main advantage of these approaches is that the performed segmentation can provide clear borders between segments that can be assigned to different disparities without smoothing disparities across these borders. However, the above discussed approaches do not support disparity variations within segments. This leads to unnaturally flat 3D scenes (i.e., cardboard effect). Concerning the required user interaction, the authors of [165] note that users have to approximately annotate every tenth frame to track foreground objects through the video. Thus, this and similar approaches (e.g., [29]) can be laborious. This is especially true for complex scenes (many objects), much object motion and motion-caused occlusions.

In [44] an alternative approach is employed to propagate user-given disparities that are provided for an entire keyframe to the remaining pixels of a video. Each frame is automatically segmented into small segments (*over-segmentation*). Next, each segment in frames without given disparities is matched to a segment in the keyframe by searching the segment with the smallest color difference and spatial distance. Finally, the known disparities are assigned to segments without disparities. Depending on the segment sizes, in this approach objects are not restricted to a single disparity. While smaller segment sizes allow more disparity variation within objects, the resulting increased number of segments also increases the complexity of the matching step. It is worth noting that small, similar segments might introduce color ambiguities which lead to wrong matches and erroneous propagations.

Finally, interactive segmentation was also used to further guide optimization-based 2D-to-3D conversion techniques (e.g., [117, 169]). As discussed in Section 2.2.1, this usage of segmentation reduces over-smoothing of object borders in disparity maps.

2.3 Summary

In this chapter, we have discussed prior work on 3D content generation with a focus on 2D-to-3D conversion techniques. First, we have explained the basic idea behind stereoscopic 3D content, i.e., the need of two shifted views and their geometric differences (disparities) to trigger the depth perception. Then, we have outlined different approaches to obtain these views and their disparities, including approaches that are based on stereoscopic data and approaches that are based on monoscopic data. The major part of this chapter has focused on 2D-to-3D conversion approaches that require different degrees of user interaction, i.e., manual, fully-automatic and semi-automatic conversion approaches. Since the algorithms proposed in this thesis belong to the latter category, our survey focused on the state-of-the-art in the latter category. We have discussed semi-automatic 2D-to-3D conversion approaches that view the propagation process as an optimization problem that aims for a globally optimal interpolation of the given disparities across the entire 2D video. These approaches typically face complex global (quadratic) optimization problems that lead to long runtimes, scalability issues and often over-smooth object borders in disparity maps. Then, simpler conversion approaches that are based on local filtering techniques were discussed. While exhibiting a lower complexity than global optimization-based techniques, filter-based techniques struggle with problems such as over-smoothing of disparity edges, error accumulation and enabling disparity changes over time for objects that move in depth. Due to their restricted (i.e., local) support from neighboring pixels, problems such as color ambiguities in the given 2D content are more pronounced than in optimization-based techniques. Our review continued with segmentation-based conversion techniques that interpret segment borders as hard object borders where smooth disparity variations should be avoided. This facilitates the distinction between different objects on the scene and leads to hard, not over-smoothed disparity edges near segment borders. However, these techniques often restrict the segments or objects in a scene to a single disparity, which results in unnaturally flat 3D scenes.

As the discussed optimization- and segmentation-based conversion techniques, the 2D-to-3D conversion algorithms presented in the following chapters are based on sparse user-input in form of scribbles, as opposed to denser input given for an entire keyframe which is common for filter-based techniques. Since our proposed algorithms exploit segmentation information in the conversion's disparity propagation, they can be regarded as segmentation-based 2D-to-3D conversion techniques. Related to optimization-based techniques, they further strive for smooth disparity variations within segments. In some similarity to filter-based techniques, our algorithms use efficient filtering techniques for disparity interpolation/aggregation and refinement.

CHAPTER 3

Fundamentals and Related Work of 2D-to-3D Conversion

3.1 User Interaction

As discussed in the previous chapters, 2D-to-3D conversion approaches attempt to recover scene geometry, i.e., disparity, that was lost during capturing a scene with a single camera. To make this highly ill-posed problem more feasible, semi-automatic 2D-to-3D conversion approaches incorporate user input to constrain the given infinite solution space. The most common form of user interaction used in previous work on semi-automatic 2D-to-3D conversion (e.g., [56, 82, 92, 117, 163, 165]) and an often used form of user interaction in other user-centric image and video editing tasks such as interactive object segmentation (e.g., [4, 5, 19, 87, 89, 97, 107, 125, 168]), alpha matting (e.g., [85, 87]), colorization (e.g., [82, 84, 87]) or tone mapping (e.g., [87, 95]) are *scribbles*. With scribble-based annotations, users directly draw strokes (*scribbles*) in images or keyframes to initialize a few pixels (Figure 3.1 *b)*). The color of these scribbles, which can also be chosen by the user, typically indicates different labels. Depending on the task, a single scribble is often restricted to a single color and label (e.g., in the case of object segmentation, one color for the object of interest). As discussed in Section 2.2.3, various segmentation-based 2D-to-3D conversion techniques (e.g., [91,165]) perform segment-wise disparity assignments. In these cases, user-scribbles are used to (successively) segment objects from the scene and are, thus, restricted to a single label in each iteration (e.g., object of interest). Contrary, in various optimization-based 2D-to-3D conversion techniques (Section 2.2.1), the color of these scribbles directly encodes the relative closeness of the marked pixels to the camera (Figure 3.2) and might also contain color falloffs, i.e., multiple labels (disparities), to indicate slanted surfaces (Figure 3.2 *c)*). Thus, the drawn scribbles intuitively initialize user-marked pixels and their appearance (i.e., pixel color as opposed to scribble color) with chosen disparities. Independent of the performed task, previous work either employs progressive scribble-based annotations, i.e., provide intermediate (or locally updated) results as instant feedback while drawing a scribble (e.g., [97, 163]), or process the annotated image or video after all scribbles have been placed (e.g., [56, 84]). Note that in both

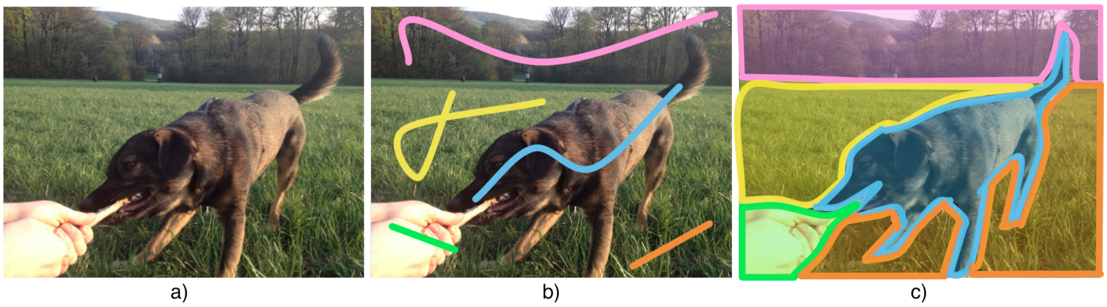


Figure 3.1: User interaction examples. *a)* Input image without user annotations. *b)* Input image annotated with scribbles. *c)* Input image with boundary-based image annotations. Scribble colors encode labels, e.g., closeness to camera, segment labels or colors to recolor the image.

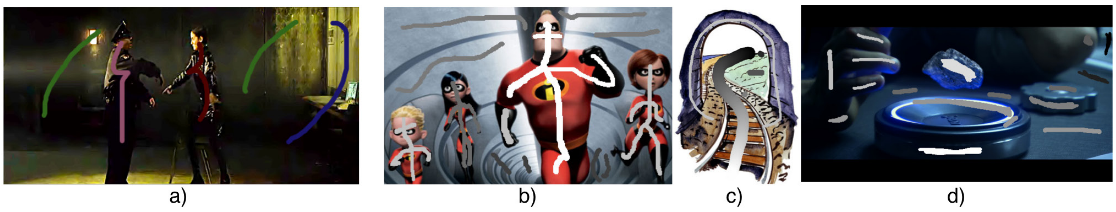


Figure 3.2: Disparity scribble examples. Input image and scribble-based annotations: *a)* color-coded disparity scribbles (from fore- to background: *rose, red, blue, green*). Example taken from [56]. *b)-d)* disparity scribbles encoded by gray values (*bright* fore-, *dark* background). Note that *c)* also contains a scribble with an intensity falloff which indicates a smooth change in disparity. Examples in *b)* and *c)* are taken from [163]. Example in *d)* is taken from [118].

cases, users have the option to change their annotations. While in the first case the result is obtained in a step-by-step manner, e.g., by updating the previous result (potentially considering only a subproblem), in the second case the processing step is repeated from scratch (and for the entire image or video). Generally, the main reason to employ a scribble-based user interface is its simplicity and its small amount of required user interaction.

An alternative form of user interaction, that is mainly used in the context of (computer-aided) manual 2D-to-3D conversion (e.g., rotoscoping [1, 72, 104]), are *boundary-based image* annotations. With boundary-based image annotations, users essentially redraw the boundaries of objects in images or video frames (Figure 3.1 *c)*), which subsequently are optimized to (ideally) cling to the object boundaries. The main disadvantage arises at objects with complicated boundaries (e.g., fur) and at fine details which can be hard to trace.

Concerning the choice of specific disparities for a scribble, the question of feasibility and correctness of the disparities may arise. In this context, cognitive studies on depth perception from 2D images suggest that establishing a depth order, i.e., assessing one point as closer or farther from the camera than another point, can be considered a mostly simple task [75, 155]. In these studies user-performed 3D reconstructions were very similar to each other, when factoring out scaling of the depth range. This indicates that in our task of semi-automatic 2D-to-3D conversion, the correct depth order of objects and their relative closeness to the camera are perceptually more important than the accuracy of the chosen disparities. A more recent user study with focus on 3D content generation [73] further indicates that an adjusted (i.e., scaled or smoothly remapped [81]) disparity range can result in equally plausible depth impressions. Thus, a disparity assignment that follows a perceptually consistent depth order is compliant with the human visual system and sufficient for semi-automatic 2D-to-3D conversion. On a related note, generated disparity maps are in many cases further adjusted to different types of displays or viewing distances, which can also involve scaling or compressing their disparity range prior to viewing [141]. We suspect that the usually performed adjustment of the disparity maps is also a reason why some semi-automatic 2D-to-3D conversion algorithms (e.g. [130]) work with normalized disparities. In these normalized disparity maps (often referred to as depth maps), zero corresponds to the point farthest away from and one to the point closest to the camera. In any case, the depth order given by (normalized) disparities can be scaled to a suitable disparity range after 2D-to-3D conversion. It is worth mentioning that the density of the annotations affects the quality and level of detail of 2D-to-3D conversion results. Dense annotations that cover disparity details within objects result in more detailed conversion results than sparser annotations that only capture the main objects. In this context user studies [73] indicate that the latter is sufficient with exception for large or salient objects.

3.2 Spatio-temporal Video Analysis

In the previous chapters, it has been indicated that spatio-temporal video analysis plays a vital role in the field of 2D-to-3D conversion. To propagate sparsely given disparities, a relation between pixels with given disparities and pixels without given disparities has to be established. For example, grouping techniques, i.e., segmentation algorithms, have been used to establish spatial (and temporal) connections between neighboring pixels (e.g., Section 2.2.1). Additionally, spatio-temporal disparity propagation in videos often relies on motion estimation (e.g., some approaches in Section 2.2.2) or tracking (e.g., some approaches in Section 2.2.3) to establish a temporal connection between pixels or pre-segmented objects across frames. The topics mentioned above within the field of spatio-temporal video analysis are discussed in the following. Section 3.2.1 focuses on graph-based representations, which connect neighboring image or video pixels. Section 3.2.2 and Section 3.2.3 review two different approaches to establish spatial and temporal relations between neighboring pixels, i.e., segmentation and edge-aware interpolation algorithms. Additional topics, such as automatic shot boundary detection (e.g., [16, 33]) to pre-segment videos into sub-videos that are more suitable for conversion or spatio-temporal video filtering for filter-based 2D-to-3D conversion techniques (e.g., Section 2.2.2) or for disparity map post-processing (e.g., [69, 78]) are as well related to the field of 2D-to-3D conversion.

3.2.1 Graph-based Representation

In context of 2D-to-3D conversion and segmentation, graph-based representations of image or video content have shown to be useful (e.g., [43, 51, 53, 117, 123, 139, 146]). In a graph-based representation, a given image or video is considered as a *weighted adjacency graph* $G = (V, E)$, where V is a set of *vertices* (or *nodes*) and E is a set of undirected *edges*. In this graph, pixels i

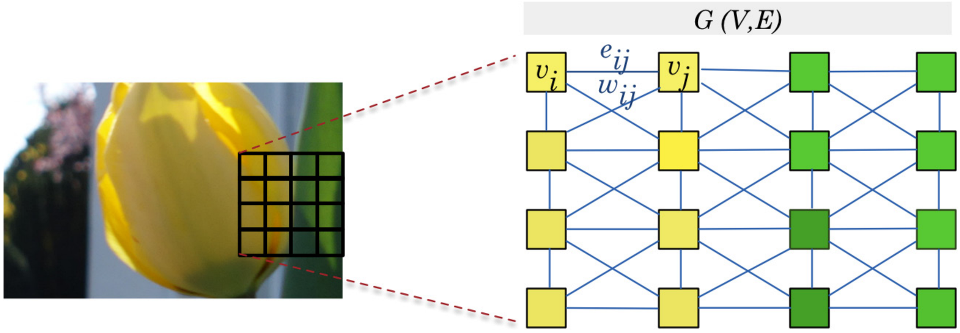


Figure 3.3: Graph-based representation. The image (*left*) is represented as an 8-connected adjacency graph $G = (V, E)$ (*right*). Graph for the marked pixels (illustrated as *squares*) in the image: Pixels i are nodes $v_i \in V$. Neighboring nodes, e.g., v_i and v_j, are connected by edges $e_{ij} \in E$. Edge weights, e.g., w_{ij}, express the similarity of the nodes connected by them.

correspond to nodes $v_i \in V$ and two neighboring pixels i and j are connected by edges $e_{ij} \in E$. Edges are typically assigned weights (*affinities*) w_{ij} that express the similarity (or costs, e.g., color difference) of the two pixels that are connected by an edge. Figure 3.3 shows an example of such a graph. In this example, each pixel of the image is connected to its eight direct neighbor pixels (8-connected neighborhood). Alternatively, in a 4-connected neighborhood, nodes are only connected to their horizontal and vertical neighbors (without the diagonal edges), which has a smaller memory footprint. In case of a video, pixels can be additionally connected across frames, e.g., with their direct temporal neighbors (same pixel position, but different frame) or with their temporal neighbors according to previously estimated motion vectors. In terms of connectivity in a neighborhood, for example, a spatially 4-connected neighborhood plus the temporal connections would result in a 6-connected neighborhood for a video.

Related to the weighted adjacency graph described above, a *weighted adjacency matrix* (or *affinity matrix*) A, which corresponds to such a graph, captures the pairwise edge weights w_{ij}. In particular, the symmetrical $N \times N$ matrix $A = (a_{ij})$, with N being the number of pixels (nodes), stores each edge weight w_{ij} at a_{ij} (and a_{ij}).

In the application of 2D-to-3D conversion or interactive object segmentation some nodes might contain additional information that was given by the user, e.g., a disparity if the pixel was marked by a disparity scribble or a segment label (i.e., fore- or background), respectively. Given a graph-based representation of an image or a video, 2D-to-3D conversion's disparity propagations, edge-aware data interpolation in general and segmentation techniques take the similarities between neighboring pixels, i.e., the edge weights, into account. Disparity propagation algorithms (smoothly) propagate the given data, i.e., disparities, preferably along edges that indicate large similarities (as characterized by small edge weights). Segmentation algorithms separate groups of pixels with large similarities (i.e., segments) from each other. In a segmentation result different groups of pixels are connected by edges with relatively small similarities (i.e., large edge weights).

3.2.2 Segmentation

In the previous chapters, it has been indicated that semi-automatic 2D-to-3D conversion poses a similar problem as segmentation. While segmentation strives for a partitioning of an image's or video's pixels into homogenous groups (that are, e.g., similar in color), 2D-to-3D conversion's disparity propagation aims for an assignment of similar disparities to similar pixels. In case of interactive techniques, another similarity arises, namely the user-centric component of both applications. Thus, this section gives a brief overview of popular segmentation approaches and discusses their similarities and differences to semi-automatic 2D-to-3D conversion. In this context, it is important to note that although segmentation and semi-automatic 2D-to-3D conversion are two closely related problems, there are differences between them. While a "hard segmentation" distinctly separates segments by their unique segment labels (i.e., all pixels within a segment obtain the same label, different segments have different labels), this is not necessarily the case in the application of 2D-to-3D video conversion. Semi-automatic 2D-to-3D conversion often (e.g., in [56, 117, 163]) allows interpolations between user-given labels (i.e., disparities) in order to obtain more realistic 2D-to-3D conversion results.

3.2.2.1 Spatial Connectivity

In computer vision, segmentation is intensively studied and a fundamental problem [5, 43, 53, 88, 89, 97, 122, 127, 146]. Segmentation is used with various applications such as tooning (e.g, [162]), object extraction (e.g., [5]), alpha matting (e.g., [4]), stereo matching (e.g., [14]), automatic 2D-to-3D conversion (e.g., [54]) and semi-automatic 2D-to-3D conversion (e.g., Section 2.2.3). As a backbone for higher-level vision tasks, there is an even wider spectrum of applications for segmentation, including recognition of human motion (e.g., [121]). One way of classifying segmentation algorithms is to consider the required user interaction. On the one hand, *automatic segmentation* approaches (e.g., [26, 32, 43, 53, 86, 114, 139]) automatically partition images into multiple groups of similar pixels. While the grouping criterion (e.g., color) might be defined, the number or properties (e.g., the specific colors) of labels are not known a priori. On the other hand, *interactive* (or semi-automatic) *segmentation* approaches (e.g., [5, 72, 79, 97, 104, 122, 125]) assign each pixel to a label from a pre-defined set of labels (e.g., object of interest and rest). The number of labels and their properties (e.g., the object of interest is green and the rest is yellow) are given by the user (i.e., in form of a few user-initialized, pre-labeled pixels).

Among the *automatic segmentation approaches* Felzenszwalb and Huttenlocher's region merging approach [43], *normalized cuts* [139] and the *mean shift* algorithm [32] can be considered popular approaches [146]. As a starting point, let us consider a graph-based representation $G = (V, E)$ (Section 3.2.1, Figure 3.3) of a given image. A simple segmentation approach can in principle apply a fixed rule, e.g., a constant threshold that is applied on the graph's edge weights w_{ij}, to merge nodes v_i into connected groups of nodes. These groups of nodes (and edges between them) are referred to as segments (or *regions*). While this approach is fast and simple, it is usually not sufficient, since it fails to model (color) variations within segments (e.g., due to lighting). While a chosen threshold might be optimal for a low contrast edge between two objects, the same threshold might in other image regions split an object into many small segments (*over-segmentation*) due to its smooth change in color. Felzenszwalb and Huttenlocher's

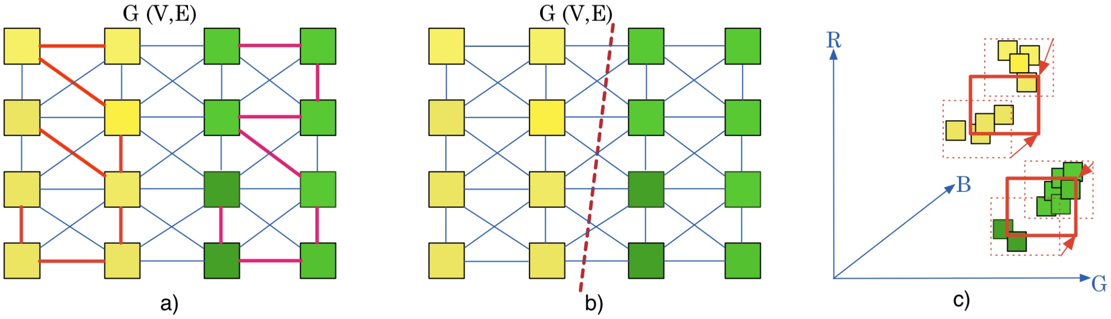


Figure 3.4: Illustration of automatic segmentation algorithms. *a)* Region merging [43]: Similar nodes (pixels) are merged (*red* edges) to larger segments. *b)* Normalized cut: [139]: The graph is iteratively split (*red* cut) in two groups of dissimilar nodes. *c)* Mean shift [32] identifies peaks in an image's color distribution (here: *R*ed, *G*reen, *B*lue) by iteratively averaging it within a local neighborhood (*red* rectangles) and groups pixels accordingly.

region merging approach [43] addresses this problem by introducing adaptive thresholds (denoted as *internal variations* or *differences* [43]) that are set relative to the (color) variation within a segment. In particular, in the beginning each node $v_i \in V$ is considered as a segment of its own. Subsequently, edges $e_{ij} \in E$ are traversed in decreasing order of their weights w_{ij}. To merge two connected nodes v_i and v_j and their associated segments, the weight w_{ij} of e_{ij} has to be smaller than the largest edge weight within their respective segments. Obviously, in the beginning a segment does not contain an edge. Hence, in this case, no maximum edge weight exists. Thus, in [43] the adaptive thresholds further depend on segment sizes. An important property of this algorithm is its runtime of $O(N \log N)$ with N being the number of pixels in an image, which makes it a fast automatic segmentation algorithm [53, 114]. This algorithm was adopted, e.g., in a hierarchical application [53] or with different features than color [86].

Contrary to forming segments through merging nodes in G (Figure 3.4 *a)*), normalized cuts [139] directly separate groups of nodes by cutting connections, i.e., removing edges, with high costs (e.g., large color differences) (Figure 3.4 *b)*). Essentially, G is recursively divided into two groups, in such a way that a normalized sum of the edge weights w_{ij} that connect nodes of the two groups is minimized. This minimization problem is defined over G's affinity matrix A (Section 3.2.1) and can be approximately solved by turning it into a *generalized eigenvalue problem* [50, 139]. This basic algorithm was extended and accelerated by several authors (e.g., [34, 138, 142, 151]). For example, Sharon et al. [138] introduce a multi-scale version of the algorithm to reduce the exponential computational costs of the original algorithm to linear computational costs (in terms of the number of pixels in an image).

Besides segmentation approaches based on merging or splitting, mode finding algorithms, such as the mean shift algorithm [32], have been proposed. These algorithms identify peaks (*modes*) in an image's color distribution (Figure 3.4 *c)*) by iteratively averaging it within local neighborhoods (*multiple restart gradient descent* [146]). Although they are computationally expensive, they have been modified and applied to several computer vision problem [114, 146, 161].

In the group of *interactive segmentation approaches*, *graph cut* algorithms (e.g., [18, 19, 127])

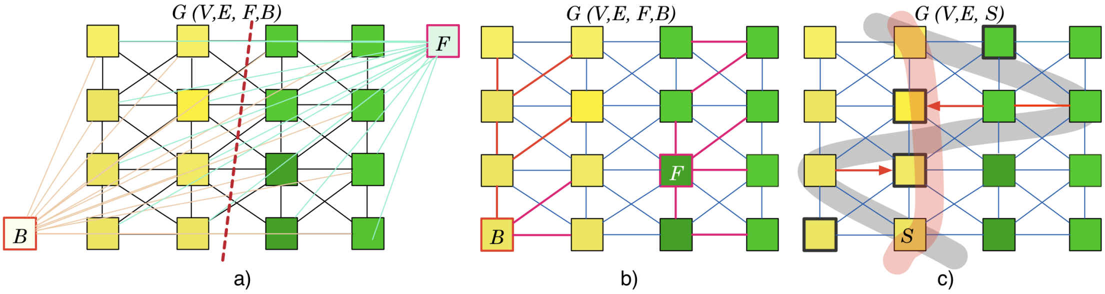


Figure 3.5: Illustration of interactive segmentation algorithms. *a)* Graph cuts [19]: Find the optimal cut (*red line*) that separates F and B. *b)* Geodesic segmentation [4]: For each node find the lowest cost path (*red*) to one of the marked pixels (F or B). *c)* Intelligent scissors [104]: Correct a user-drawn curve (*gray*) with a new curve (*red*) that follows the lowest cost path. This path starts from the user-given starting point S of the curve and ends at its current end point. (Contrary to *b)*, in *c)* edges have low weights if they are likely to lie on an object boundary.)

and *active contours* (e.g., [72,104]) are popular approaches to perform a binary segmentation [146]. Concerning the former approaches, Boykov and Jolly [19] extend the graph-based presentation discussed in Section 3.2.1 by two additional *seed* nodes that correspond to the user-marked foreground pixels F and the user-marked background pixels B, respectively (Figure 3.5 *a)*). The two additional nodes also come with additional edges to each of the other nodes. The weights of these additional edges express the similarity of the given nodes to the user-marked nodes, i.e., F and B. In [19], Boykov and Jolly show that the binary segmentation problem, i.e., here, finding the globally optimal cut to separate F and B, can be efficiently and exactly solved by formulating it as a *maximum flow/minimum cut* graph optimization problem [52]. The basic graph cut algorithm [19] was extended and applied by many authors to different problems in computer vision (e.g., [5, 18, 89, 97, 118, 122, 127, 157]). For example, *GrabCut* [127] is an iterative version of the basic algorithm with an improved user interface for binary segmentation (i.e., bounding box to roughly mark the object of interest) and an additional step to estimate transparencies at object borders (*alpha matting*). Alternative to graph cut algorithms other techniques can be used to separate F and B, e.g., [4, 18, 125]. For example, in [4] the binary segmentation is efficiently ($O(N)$, N number of pixels in an image) computed based on the geodesic distance of each node to fore- and background scribbles (F and B) (Figure 3.5 *b)*). Contrary, in [79, 125] the label-based optimization is approximated using a filtering technique. Due to embedding the local pixel similarities in a local filter-based context instead of a global formulation, a binary segmentation can be performed at interactive rates [125].

Finally, the concept of active contours (e.g., [72, 104]) should also be briefly discussed. Given a rough initialization by the user (e.g., boundary-based annotation of an object), a given curve is optimized to cling to strong gradients (edges) in an image. For example, *intelligent scissors* [104] update the curve in real-time while a user is drawing it. This is essentially done by continuously searching the lowest cost path in G between the starting point S of the user-drawn curve and the current end position of the user-drawn curve (Figure 3.5 *c)*). Note that in this case, edge weights have lower costs if they are likely to coincide with an object boundary. To prevent the entire curve from flickering, intermediate curves are regularly saved and new starting points (i.e., the end point of the saved curve) are assigned. As mentioned above, active contours are, among other applications, also used for rotoscoping (e.g., [1]) and, thus, for manual 2D-to-3D conversion approaches. They are also used in the context of semi-automatic segmentation-based 2D-to-3D conversion approaches (e.g., [165]).

3.2.2.2 Temporal Connectivity

When segmenting videos or propagating sparsely given data within videos, not only a spatial but also a temporal connection between pixels has to be established. Applying an image segmentation algorithm individually to each frame of a video results in segments which do not extend over time. Furthermore, establishing a temporal correspondence of spatial segments between adjacent frames poses the problem of temporal coherence, with unstable segmentation results causing flickering of region boundaries. To this end, video segmentation approaches often take one of the following two fundamental strategies: They either (1) consider the whole video volume or (2) process the video in a frame-to-frame manner. In both cases, motion information can be incorporated to address the problem of temporal coherence. As briefly mentioned in Section 3.2.1, temporally neighboring pixels can be connected across frames to, for example, their direct temporal neighbors (same pixel position, different frame) or according to previously estimated motion information. The most common form of dense motion information used in the context of video segmentation are *optical flow vectors* (OF) (e.g., [6, 7, 61, 99, 110, 125]). Assuming that between a pair of adjacent frames pixel colors are approximately constant and only their position changes from frame to frame, the optical flow vector for each pixel is estimated according to the change in patterns from one frame to the next (Figure 3.6 *a)*). While not discussed further in this thesis, detailed reviews on optical flow estimation techniques and their challenges can be found in [6], [7] and [146]. Another common approach to estimate motion, that shall be briefly mentioned in this thesis, is *tracking* (e.g., [98, 152, 171]). Contrary to optical flow, motion information provided by tracking approaches is usually sparse, i.e., only determined for a few points or segments (Figure 3.6 *b)*). These sparse points can be locally tracked by iteratively computing their new position in the next frame (e.g., [152]) or matched by individually comparing each point (i.e., its local image properties) in one frame with a set of sparse points in the following frame and choosing the best fit (e.g., [98]). The latter is especially useful when large motions are expected. For an in-depth review on object- and point-tracking, interested readers are referred to [171] or [146].

Returning to our initial problem, namely establishing temporal connectivity in video segmentations, processing the entire video volume at once (e.g., [4, 32, 36, 53, 88, 160, 161]) can be considered as one basic strategy. For example, when extending the previously discussed mean shift segmentation [32, 36, 161] to videos, peaks are identified with respect to each pixel's color and location. DeMenthon et al. [36]'s mean shift version additionally adds OF to the pixels' feature space. Considering an example in the field of interactive segmentation, in [160] a binary graph cut optimization is defined across the entire video volume to obtain a binary segmentation of a video. In [53] and [86], the previously discussed region merging approach from Felzenszwalb and Huttenlocher [43] is extended to process videos by adding temporal edges to the adjacency graph. These additional edges connect temporally neighboring pixels according to previously computed optical flow vectors. Some recent segmentation approaches take on a more global view

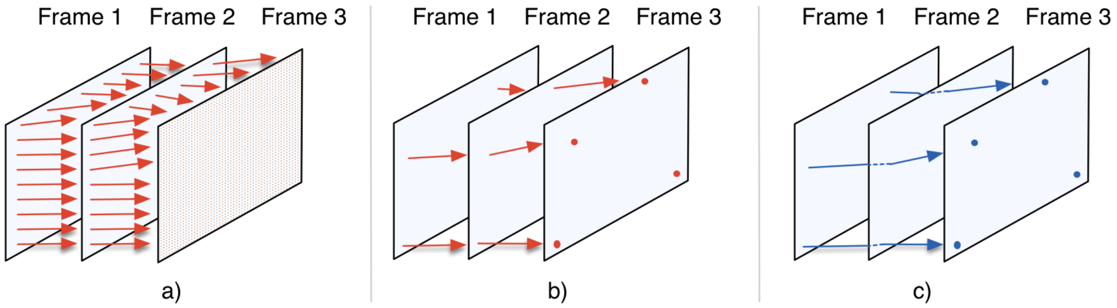


Figure 3.6: Illustration of common forms of motion information. *a)* Dense motion between pairs of frames, e.g., optical flow: Each pixel has a motion vector (*red*) that points to a corresponding pixel in the next frame. *b)* Sparse motion between pairs of frames, e.g., tracking: Motion information (*red*) between pairs of frames is only determined for a few points. *c)* Long-term motion: The entire motion path (*blue*) from the first to the last frame builds a long-term motion cue (compared to the shorter motion paths between pairs of frames in *b)*).

when incorporating motion information in the segmentation process (e.g., [26, 86, 109]). These approaches investigate *long-term motion information*, which can provide richer information than local motion information (such as optical flow). For example, objects that initially exhibit similar movement, but move apart in later frames, can be more easily separated from each other. To this end, these approaches use tracking to derive point trajectories (e.g., [131, 145]) that can expand across multiple frames (Figure 3.6 *c)*). Trajectories are then grouped according to global motion similarity (e.g., [26, 86, 109]). However, as these trajectories are sparse, so too is the segmentation result. In order to assign the remaining points to segments, additional information (e.g., color [86, 109]) can be incorporated.

The second basic strategy is to establish temporal connectivity in video segmentations in a frame-to-frame manner (e.g., [113, 122, 149, 160]). For example, in [149] active contours that were computed in one frame are locally tracked [152] to the next frame. Likewise, in [122], various segmentation cues that were extracted from the previous frame are transferred to the next frame, using local motion vectors and tracked points. Subsequently, the cues are used to segment the frame to which they were transferred to. In [160], segments that were obtained for each individual frame are merged by applying a spatio-temporal mean shift segmentation procedure to them. Relatedly, Paris [113] suggested a mean shift segmentation that only takes past frames into account (as opposed to the entire video volume). Since the mentioned approaches process a reduced amount of data compared to approaches that process the entire video at once, they are more scalable. This is especially true for segmentation algorithms that rely on global optimization and, thus, have to access all pixels in a video at the same time. A similar strategy to [113] can also be used to reduce the runtime and the memory footprint of a segmentation approach. To this end, a video can be split into overlapping sub-videos which are processed sequentially, while incorporating methods for handling borders between the sub-videos to maintain the spatio-temporal coherence of the segmentation (e.g., [53]).

3.2.3 Edge-aware Interpolation

Besides discrete label-based solutions such as segmentation, semi-automatic 2D-to-3D conversion is also closely related to sparse data interpolation problems (e.g., in Section 2.2.1). Contrary to segmentation, interpolation does not restrict the solution to a prior given set of integer labels, but allows pixels to be assigned to a mixture of labels. Such an interpolation ideally respects the edge weights in G (i.e., adjacency graph in Section 3.2.1), e.g., is smooth in regions with homogenous colors and allows abrupt changes at color edges. For the purpose of scribble-based image and video manipulation a number of edge-aware interpolation approaches have been proposed (e.g., [30,40,55,84,87,100,167,170]) in the field of edit propagation. In this context, an image manipulation operation (e.g., colorization [84]) is specified with scribbles and interpolated to the rest of the image or video. This section briefly discusses a few of these examples.

A fundamental example of solving such an interpolation is global *regularization*, which was originally developed by statisticians to fit models to data (e.g., finding a smooth surface through a set of sparsely given data points) [146, 150] and was applied to various computer vision problems (e.g., [56, 61, 84, 85, 87, 95]). In regularization, sparse data interpolation problems are globally solved in the context of continuous optimization problems that assume a solution that smoothly varies between closely related nodes and has a minimal distance to the sparsely given data. When considering the graph-based representation G of an image or video, the smoothness of the solution in a particular image area is controlled by G's edge weights (or its corresponding weighted adjacency matrix A, in Section 3.2.1). For simplicity such optimization problems (e.g., those discussed in Section 2.2.1) are often expressed as quadratic energy functions, that can be minimized by solving an equivalent linear system of equations using standard techniques (e.g., in the simplest case *least squares estimation* [84]). As alternative to quadratic energy functions that tend to over-smooth the solutions for sparse user inputs and often cause *halo artifacts* (i.e., inverse gradient) around objects [30, 66], *robust regularization* [146] uses non-quadratic (*robust*) energy functions to further reduce the smoothness near color edges. However, in these cases different iterative optimization techniques have to be used [146]. In any case, as pointed out in many publications (e.g., [30, 55, 63]), solving the corresponding global optimization problem in a global manner, inevitably results in low processing speeds and large memory footprints, especially when working with videos or high-resolution images. Thus, previous works in the field of edit propagation with focus on the improvement of the quality of their results (e.g., [40, 55, 56, 87]) and with focus on the scalability of the problem (e.g., [30, 100]) have been proposed. Concerning the former, in [87] and [56] this basic problem was extended by adding "given data" to increase the density of the user input and the quality of the results. This additional "given data" is previously determined by a classification according to the appearance (i.e., color) of the marked pixels. Other proposed quality improvements include definitions of more suitable similarity measures (e.g., [40]) or sparse models [167]) which relate each pixel to only a part of the given user inputs to avoid over-smoothing (due to influence of several input data) in areas that are far away from or in between user inputs. Related works in the field of edit propagation that focuses on scalability includes the definition of the propagation in terms of segments as opposed to pixels (e.g., [100]) or the prior propagation within sparser representations of images (e.g., [30]).

In some similarity to geodesic segmentation (which was briefly mentioned in Section 3.2.2.1), efficient lowest-cost-path algorithms can also be used for edge-aware sparse data interpolation.

For example, in [170] nodes in G without given data are assigned a weighted average of the given data points. The weights are determined by geodesic distances (i.e., computed from the lowest cost path in G) between each node without given data and each node with given data.

Although not specifically designed for sparse data interpolation, edge-preserving smoothing techniques also achieve edge-aware interpolations, however, based on dense input data. More precisely, given an image or video, these techniques aim to compute a new image that is similar to the given one, but smoother in image regions that are similar in color, while still containing its strong edges. In [39] this is achieved with a regularization that considers dense input data. Additionally, joint filtering techniques that filter one image according to local weighted averages in another image have been proposed (e.g., [46, 59, 69, 153]). To preserve strong image edges, these techniques choose the weights according to local color similarities. In Section 2.2.2, we have already discussed the usage of such filtering techniques in the context of 2D-to-3D conversion, including recent advances on approximating the global optimization process in edge-aware data interpolation problems by iterative edge-aware smoothing (e.g., [46, 82, 129]).

3.3 Evaluation

Missing evaluation benchmarks that are explicitly tailored to the problem of semi-automatic 2D-to-3D conversion complicate the quantitative evaluation of the proposed conversion algorithms. Ideally, such a benchmark not only provides suitable *ground truth* (GT), but also considers the interactive component of such conversions. When comparing two semi-automatic conversion algorithms, it makes sense to use exactly the same user input for both of them. However, this information is usually not available from the published papers and the user interfaces of different conversion algorithms vary (e.g., discussed in Section 3.1). In the context of user input, it was pointed out in [76] that user-centric applications, which include semi-automatic 2D-to-3D conversion approaches, are not only highly dependable on the given user input, but might also prefer a certain type of user input over another. This observation further complicates the quality assessment in the field. As in previous works, most of the evaluations in this thesis are performed using the same input scribbles for all compared 2D-to-3D conversion approaches. However, in a final contribution, in Chapter 6, we also consider different types of user input for our evaluations.

In this thesis and in the 2D-to-3D conversion literature, different recorded or computer-generated 2D data and corresponding reference solutions are used to evaluate 2D-to-3D conversion results. Below, we give examples for such data and discuss its usage for such evaluations. Appendix A provides a more detailed description of the data used in this thesis.

The recorded test data used in this thesis comprise self-recorded stereoscopic videos (Figure 3.7 *a)*) and recorded stereoscopic images that are provided by the well-known Middlebury stereo matching benchmark [132–135] (Figure 3.7 *b)*). Furthermore, monocular color videos that were recorded with a special camera that captures a color video and additionally measures depth (e.g., [42]) have been used to evaluate 2D-to-3D conversion results (e.g., in [28, 90]). In case of the images from the Middlebury dataset, the provided corresponding GT disparity maps are either a combination of hand-labeled and estimated disparity maps [134] or were generated using structured lighting [132, 133, 135]. These images were originally intended for evaluating stereo matching algorithms. For our self-recoded videos, the corresponding reference solutions

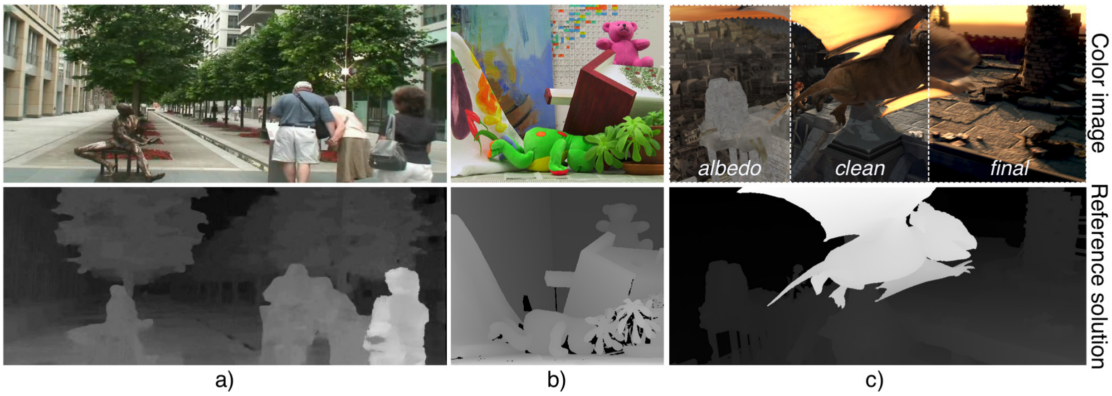


Figure 3.7: Reference solutions used in evaluation. Color images (*top*) and corresponding reference solutions (*bottom*). Reference solutions are encoded by gray values: Foreground: *bright*, background: *dark*. *a)* Disparity map generated by a stereo matcher [12]. *b)* Disparity GT from the Middlebury stereo matching benchmark [134]. (Here, the disparity for *black* pixels is not provided.) *c)* Depth GT from the MPI Sintel dataset [27], which provides the color videos in three levels of difficulties (i.e., *albedo*, *clean* and *final*). The shown color frame is split into three parts and each displays a different level of difficulty. The GT corresponds to the entire frame.

are disparity maps that were estimated using a stereo matching algorithm (i.e., [12]). In our quantitative evaluations, we compare 2D-to-3D conversion results, i.e., the conversion-generated disparity maps, with the respective reference solutions. The generated 2D-to-3D conversion results are based on disparity scribbles that are drawn in the image or in the first and the last frame of one (monocular) view from the stereoscopic image or video. Note that we completely disregard the other view. The scribbles define which disparity should be propagated. To enable a comparison with the reference solution, the disparities for the marked pixels are defined as the disparities of the reference data at the scribble positions. For 2D-to-3D conversion algorithms that support multiple disparities per scribble, the disparities from the reference solution can be used directly. For algorithms that only support one disparity per scribble the definition has to be adapted, e.g., to use the mean disparity of all marked pixels' disparities for each individual scribble. The subsequent comparison of the conversion-generated disparity map and the reference solution gives a hint on the quality of the conversion process, relative to the quality of the reference solution. In this context, disparity maps obtained by stereo matching are typically inferior to disparity maps obtained by structured light techniques [132, 133, 135]. However, the latter require a special hardware setup to capture the disparity of each pixel. Reference solutions determined by cameras that measure depth might suffer from high noise levels and systematic distortions [124]. While computer-generated test data might be most accurate, performing the conversion on real (self-)recorded videos as opposed to computer-generated videos is a closer fit concerning the practical operating conditions of a 2D-to-3D conversion algorithm.

The computer-generated test data used in this thesis are stereoscopic videos (e.g., Figure 3.7 *c)*) that were rendered with disparity GT (e.g., [102]) and monocular videos that were rendered with depth GT (in meters) (e.g., [27]). The rendered GT directly results from the depth of the 3D models in a scene. The videos from [102] and [27] were originally intended for evaluating other computer vision problems than semi-automatic 2D-to-3D conversion. The Tsukuba dataset was proposed by [102] to evaluate stereo matching algorithms. The MPI Sintel dataset [27] (Figure 3.7 *c)*) initially only provided OF GT to evaluate algorithms that estimate OF. However, recently the authors of [27] extended their dataset by the MPI Sintel Depth and MPI Sintel Stereo data, which both can be used for the evaluation of 2D-to-3D conversion algorithms[1]. For the purpose of evaluating our 2D-to-3D conversion results, the reference solution is used as discussed above (i.e., propagation of reference solution at scribble positions and subsequent comparison with the conversion result). Although being computer-generated and not (self-)recorded, this dataset poses challenges for 2D-to-3D conversion algorithms, including fast moving objects, objects that move in depth and color ambiguities between objects. The color videos are provided in different levels of difficulty, e.g., one containing shadows, reflections and motion blur, while another one is rendered without these effects. In Figure 3.7 *c)* the difference of these renderings can, for instance, be observed at the dragon's wing that has a solid color in the *albedo* rendering and is transparent in the *clean* rendering. While the stone floor in the albedo and clean rendering is not affected by the red twilight, the stone floor in *final* rendering takes on a reddish color tone. Given these renderings, evaluations that focus on the sensitivity to their additional illumination effects and, thus, caused color ambiguities can be performed.

Finally, it should be mentioned that in the 2D-to-3D conversion literature, evaluations focus not only on quantitative comparisons of conversion results with reference solutions (e.g., [28, 31, 56, 58, 94, 156, 159, 175]), but also on user studies in which participants subjectively assess the quality of the conversion results (e.g., [28, 56, 92, 94, 117, 163, 175]). In these cases, the subjective quality of a generated disparity map, or more precisely, of stereoscopic data that was generated from it, is evaluated. This evaluation strategy has two potential advantages. It keeps the user in the loop and considers additional steps in the 3D content generation pipeline (i.e., novel view generation). The accuracy of disparity maps does not necessarily reflect the quality of novel views that were generated from it and the quality of the 3D impression that eventually emerges from the generated data. For example, while an erroneous disparity assignment in homogenous image areas is often barely noticed in stereoscopic viewing conditions, errors in textured or salient image areas might lead to visible distortions. In fact, our published user study in [106] shows that the results of quantitative evaluations of stereoscopically generated disparity maps only weakly correlate with those of a corresponding subjective user study. Under consideration of our field of 2D-to-3D conversion, this was recently also confirmed by Kellnhofer et al. [73]. They observe a wide tolerance to distortions that are common in this field (e.g., spatial and temporal smoothing artifacts) when 3D content is viewed in stereoscopic conditions compared to more sensitive responses to these distortions when performing a quantitative evaluation with respect to a reference solution. Hence, in the context of 3D content generation subjective quality assessments can be more meaningful than quantitative comparisons with GT disparities or with other reference solutions. A major drawback of this evaluation strategy is, that it is labor-intensive and time-consuming for both the person who conducts the study and for the participants (e.g., approximately 40 minutes per participant to perform 64 trials [73]). Moreover, if the user

---

[1] Neither depth nor disparity GT were publicly released at the time our evaluations were performed. In our evaluations we use an early version of the depth GT (in meters) that was provided by the authors of [27].

study uses paired comparisons (e.g., as in [106]) the rapid growth in the number of required trails (e.g., six images and seven approaches results in 126 trails) further limits the amount of investigated content. This is especially the case, if the study considers videos. Therefore, in the literature (e.g., [28, 31, 56, 58, 94, 156, 159, 175]) as well as in this thesis, video conversion results are often evaluated by comparing them with reference solutions.

3.4 Summary

In this chapter, we have described fundamental techniques that are used in semi-automatic 2D-to-3D conversion algorithms and concepts closely related to them. First, we have discussed two common forms of user interaction, i.e., less labor-intensive, sparse scribble-based annotation and more comprehensive boundary-based annotation. This discussion also included the feasibility of annotating 2D content with disparity in general.

Then, we have given an overview of spatio-temporal video analysis techniques that are relevant for the field of 2D-to-3D conversion. In this context, we have briefly discussed commonly used graph-based representations of image and video content as a starting point. On this basis, we have reviewed two fundamentally different approaches to establish spatial and temporal relations between neighboring pixels, i.e., discrete spatio-temporal segmentation techniques and continuos edge-aware sparse data interpolation techniques. While the former provide discrete, hard boundaries between dissimilar pixels, the latter produce smooth interpolations of the given data in the image or video. In the context of segmentation techniques, we have briefly discussed popular interactive and automatic segmentation approaches and common strategies towards spatio-temporal segmentation, which include the incorporation of motion information. In the context of edge-aware sparse data interpolation, we have given an overview of the general problem and have briefly discussed proposed improvements in terms of accuracy and scalability. Although not specifically designed for sparse data interpolation, we have touched on edge-aware filtering techniques which perform interpolations on dense input data, i.e., they smooth images or videos. In the context of 2D-to-3D conversion, both approaches (i.e., segmentation and interpolation techniques) have their individual advantages and disadvantages. In particular, segmentation and hard boundaries facilitate the distinction between areas in which smooth disparity changes are desired (i.e., inside objects) and areas in which smooth disparity variations should be prevented (i.e., at object borders). Edge-aware interpolation techniques are able to produce such smooth disparity variations, which are necessary to model disparity changes over time and for rounded objects. However, when employing edge-aware interpolation techniques over-smoothing and scalability issues have been reported. Hence, the algorithms that will be presented in the subsequent chapters leverage both techniques aiming to combine their advantages.

Finally, we have briefly discussed issues considering the evaluation of 2D-to-3D conversion algorithms, including evaluation benchmarks, the role of user input and the choice between objective and subjective evaluations. In this context, we have given a brief description of evaluation strategies that are used in the literature and this thesis.

CHAPTER 4

Segmentation-based 2D-to-3D Conversion of Videos

4.1 Introduction

This chapter describes the first semi-automatic 2D-to-3D conversion algorithm that is presented in this thesis. The proposed algorithm cost-efficiently converts 2D videos to 3D based on scribble-based annotations in the first and last frame of the 2D video. As mentioned in the previous chapters, related existing 2D-to-3D conversion algorithms mainly focus on segment-wise disparity assignments or edge-aware interpolations, which can result in either not smooth enough conversions (i.e., cardboard effect) or too smooth conversions (i.e., over-smoothing of object borders). We propose a semi-automatic 2D-to-3D conversion algorithm that performs the disparity propagation jointly with an automatic video segmentation. This joint approach is, to the best of our knowledge, new for semi-automatic 2D-to-3D conversion. A subsequent spatio-temporal filtering step enables smooth disparity changes within segments and over time and yields the final disparity video for each frame of a 2D video. The main advantage of our approach over previous works comes from the direct use of segmentation during the propagation process. Specifically, jointly performing a segmentation preserves disparity edges at object borders, which is challenging for previous works. Contrary to previous segmentation-based 2D-to-3D conversion algorithms (e.g., [29, 44, 91, 165]), the proposed algorithm models smooth disparity variations within segments.

In this chapter, we further describe a scalable, optimized GPU implementation of our 2D-to-3D conversion algorithm. Our proposed acceleration leads to runtimes, e.g., one fps for a video with a resolution of 640 × 480 pixels per frame.

We test our 2D-to-3D conversion algorithm on various videos and compare it to (GT) reference solutions. These quantitative evaluations show that our proposed algorithm and its acceleration generate disparity videos of high conversion quality and outperform a well-established semi-automatic 2D-to-3D conversion approach [56] when using the same user input.

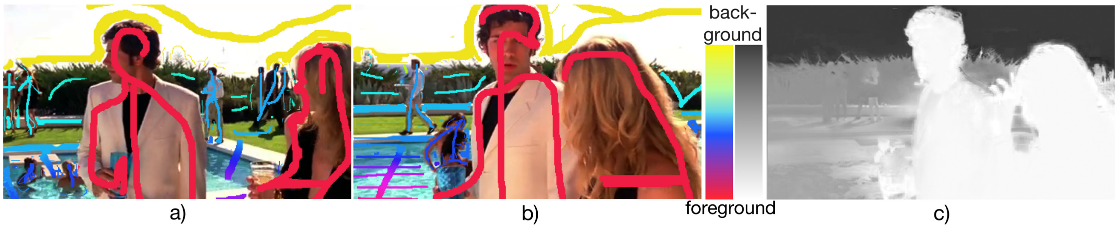


Figure 4.1: Input and output example of proposed 2D-to-3D conversion algorithm. The first frame *a)* and the last frame *b)* of the input video are annotated with scribbles, whose *hues* encode disparity. The disparity map *c)* for an intermediate frame, generated by our approach, encodes disparity by *gray values*. Copyright of original video: Warner Bros.

4.2 Proposed Algorithm

The proposed 2D-to-3D conversion algorithm consists of three steps. First, the user annotates a 2D video with disparity scribbles (Section 4.2.1). Then a joint segmentation and propagation algorithm identifies spatio-temporal segments in the 2D video and assigns disparities to each pixel (Section 4.2.2). A final step interpolates disparities over time and refines the disparity video by applying a smoothness filter (Section 4.2.3). Figure 4.2 illustrates the corresponding program flow.

4.2.1 Disparity Scribbles

As discussed in Section 3.1, scribbles are a common and effortless form of user interaction. Thus, we have chosen a scribble-based user interface which enables the user to provide initial disparities in the 2D video (Figure 4.1 *a)* and *b)*) as an input to our algorithm (Figure 4.2). For each 2D video shot, the user draws colored scribbles in the first and in the last frame. The scribble colors indicate whether the user believes the marked pixels are more in the foreground or more in the background. Similar to [56], in our user interface the hues of the scribbles encode the (assumed) normalized disparity for the marked pixels.[1] In principle, it is enough to annotate, e.g., only the first frame of a video. However, to indicate disparity changes over time, the last frame has to be annotated as well. Contrary to various previous segmentation-based 2D-to-3D conversion approaches (e.g., [82, 91, 165]), in our approach the scribbles are only used to initialize disparities and not to guide a segmentation algorithm. Although at the first glance this decision might seem surprising, it, in fact, has a practical motivation. Since objects (or segments) can be rounded or slanted, they might contain multiple disparities and, hence, should be able to be marked with several scribbles that encode different disparities. We also allow scribbles to contain disparity falloffs to indicate slanted or rounded objects (e.g., Figure 4.1 *a)* and *b)*, pool). Furthermore, different objects might have the same disparity, and, hence, for simplicity should be able to be marked with a single scribble without being forced to belong to one segment (e.g., Figure 4.1 *a)*, person and glas). By using scribbles only for disparity initialization, when performing the annotations users do not have to consider borders between objects with the same disparity.

---

[1] After the 2D-to-3D conversion the resulting normalized disparity maps $\in [0, 1]$ can be further adjusted for different types of displays and viewing distances, e.g., by scaling or mapping them to a desired disparity range.

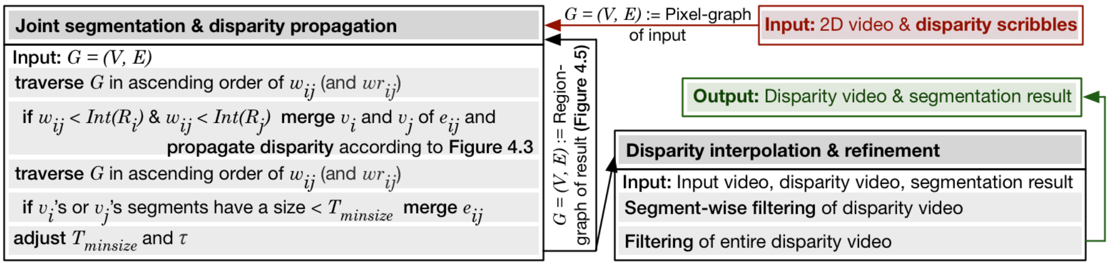

Joint segmentation & disparity propagation
Input: $G = (V, E)$
traverse G in ascending order of w_{ij} (and wr_{ij})
if $w_{ij} < Int(R_i)$ & $w_{ij} < Int(R_j)$ merge v_i and v_j of e_{ij} and propagate disparity according to Figure 4.3
traverse G in ascending order of w_{ij} (and wr_{ij})
if v_i's or v_j's segments have a size < $T_{minsize}$ merge e_{ij}
adjust $T_{minsize}$ and τ
$G = (V, E)$:= Pixel-graph of input
Input: 2D video & disparity scribbles
$G = (V, E)$:= Region-graph of result (Figure 4.5)
Output: Disparity video & segmentation result
Disparity interpolation & refinement
Input: Input video, disparity video, segmentation result
Segment-wise filtering of disparity video
Filtering of entire disparity video

Figure 4.2: Program flow of proposed segmentation-based 2D-to-3D conversion algorithm. The *joint segmentation and propagation* (Section 4.2.2) iteratively assigns user-given disparities (Section 4.2.1) to each pixel in a 2D video, which results in a dense disparity video. The subsequent *disparity interpolation and refinement* steps (Section 4.2.3) spatio-temporally filter disparities per-segment and independent of the segmentation, respectively. See text for details.

4.2.2 Joint Segmentation and Propagation

The basic idea of our algorithm is to propagate the user-given disparities to the remaining pixels of the video while segmenting it. To this end, we adopt the segmentation algorithm of Grundmann et al. [53] for our task. As we will describe in this section, the merging approach of the segmentation algorithm agrees with our task of disparity propagation. The adopted video segmentation algorithm [53] efficiently (one fps [53], corresponding frame resolution not provided in [53]) produces temporal-coherent segmentations and can be applied to videos shots containing motion, partial occlusions and illumination changes [53]. The segmentation algorithm comprises two steps. First, a generalized version of an image segmentation algorithm [43] is applied, which is based on region merging according to local pixel similarities. Then, neighboring segments (or regions) are merged according to their segment similarity [53]. These steps use the same merging process [43], but apply it on different graph-based representations of the video (Figure 4.2). Below, we give a review of this algorithm and adopt it for disparity propagation.

We begin by discussing the generalization of Felzenszwalb and Huttenlocher's image segmentation approach [43] (Section 3.2.2) for videos and its adoption for disparity propagation. Let us consider the graph-based representation $G = (V, E)$ of the given 2D video and the user-provided disparity scribbles (Section 3.2.1). In this *pixel-graph*, each pixel i in the video is considered as a vertex $v_i \in V$, which additionally stores a disparity if it was marked by a disparity scribble. Vertices are connected to their spatial and temporal neighbors by edges $e_{ij} \in E$, where i and j are pixels (vertices). Temporal edges either connect direct neighbors of a pixel in an adjacent frame or when using optical flow (OF) [110], the neighbors along the corresponding OF vector. To express the similarity of two connected pixels, each edge is associated with a weight w_{ij}, i.e., their normalized color difference [43]. In the following, these vertices are grouped into segments. Initially, each vertex v_i is considered as a segment R_i of its own. We traverse edges e_{ij} in decreasing order of their weights w_{ij}. Following this fixed merging order, vertices connected by an edge e_{ij} and their associated segments are merged if the *internal variations* [43] of both segments are larger than the weight w_{ij}. The internal variation $Int(R_i)$ of pixel i's associated

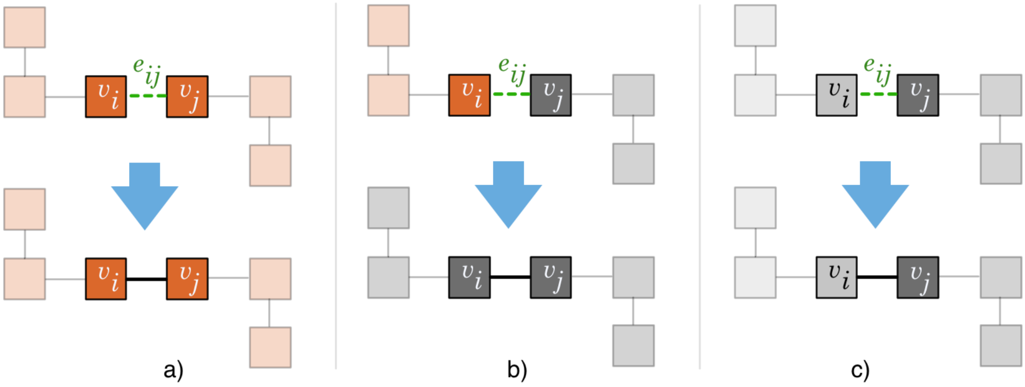


Figure 4.3: Propagation rule set. *a)-c)* illustrate possible merging scenarios for an edge e_{ij} (*green*) that connects vertices v_i and v_j (and their associated segments that are connected by solid lines) with disparities (*gray*) and without disparities (*orange*). *a)* Merging two segments without disparity results in a segment without disparities. *b)* Merging a segment with and a segment without disparities propagates the given disparity (i.e., from v_j) to the pixels of the segment without disparities. *c)* Merging two segments with disparities preserves them.

segment R_i can be considered as an adaptive threshold and is defined as [43]:

$$Int(R_i) = \max_{e_{ij} \in MST} w_{ij} + \frac{\tau}{|R_i|}. \tag{4.1}$$

In the first term of Equation (4.1), the maximal edge weight of the *minimum spanning tree* (MST), which spans a segment R_i, is used to express a segments's internal variation. Thus, color variations inside a segment are tolerated. The second term of Equation (4.1) makes this expression dependent on the segment size $|R_i|$. Since for small segments $max_{e_{ij} \in MST} w_{ij}$ is determined only from a small number of edges, it might not be a good estimate [43]. Thus, the dependency on $|R_i|$ requires a stronger evidence for a segment boundary (larger edge weight) for smaller segments than for lager ones. Furthermore, $|R_i|$ is necessary for handling segments that consist only of one vertex and, thus, no edges. In Equation (4.1), τ is a parameter which influences the precision of the segmentation result (larger τ produces larger segments, but less accurate results [53]). Finally, the number of segments is reduced by merging low-cost edges of segments containing less than a fixed amount of pixels $T_{minsize}$ in ascending order of their weights.

In the proposed semi-automatic 2D-to-3D conversion approach, we use the segmentation algorithm described above to simultaneously propagate disparities when merging two segments (Figure 4.2). Specifically, when forming segments by merging edges e_{ij} according to Equation (4.1), there are three possible merging scenarios with respect to the combinations of given and not given disparities (Figure 4.3). e_{ij}'s vertices (and their associated segments) can have a (user-given or propagated) disparity or the disparity is still unknown. As stated above, before applying our joint segmentation and propagation process each vertex is considered as a segments of its own, which only has a disparity if it was covered by a user-scribble. If a segment with unknown disparity is merged with a segment with known disparity, the known disparity is propagated to the other segment, i.e., assigned to its pixels (Figure 4.3 *b)*). Secondly, merging two segments

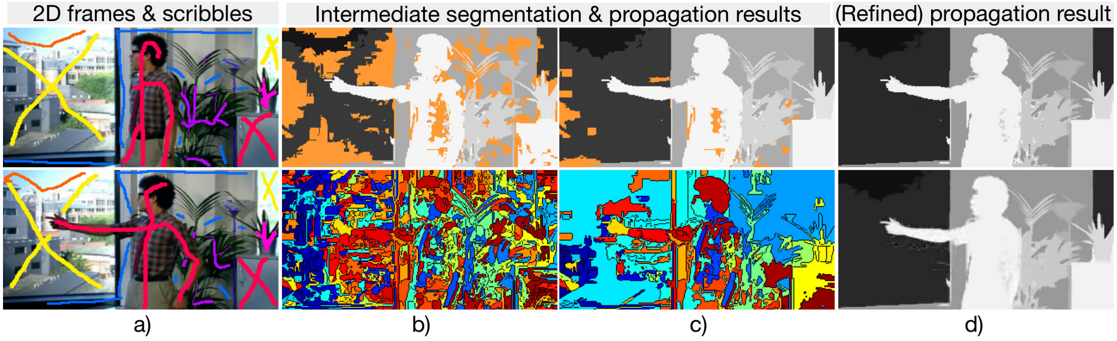

2D frames & scribbles
Intermediate segmentation & propagation results
(Refined) propagation result
a)
b)
c)
d)

Figure 4.4: Joint segmentation and propagation process. *a)* Disparity scribbles in the first (*top*) and last frame (*bottom*). *Hues* encode disparity. *b)* Pixel-wise over-segmentation of a middle frame (*bottom*) and its corresponding propagation result (*top*). *Orange* pixels (*top*) have unknown disparities. Different segments have different randomly chosen colors. *c)* Region merging, i.e., segmentation of the region-graph (*bottom*) and corresponding propagation result (*top*). *d)* Assignment of disparities to the missing segments and inter-segment smoothing (*top*). Result after refinement step (*bottom*). Copyright of original video: FremantleMedia Limited.

without disparity also yields a segment without disparity (Figure 4.3 *a)*). Thirdly, if segments with conflicting disparities are merged, their respective disparities are preserved (Figure 4.3 *c)*). As a result, spatio-temporal segments may contain vertices with different disparities (e.g., *blue* segment in Figure 4.4 *c)*, *second row* contains different disparities for the wall and lamp in the disparity map shown in Figure 4.4 *c)*, *first row*). Note that this property is of particular importance for segments that represent slanted surfaces or change their disparity over time.

The result of the pixel-wise segmentation and joint propagation process that was described above is a spatio-temporal over-segmentation of the 2D video, in which some segments are assigned to disparities (e.g., Figure 4.4). Note that for some segments the disparities remained unknown (Figure 4.4 *c)*, *top, orange pixels*) and, thus, have to be further merged applying our propagation rule set. To this end, we represent the intermediate result as a graph (Figure 4.5). In particular, we define a *region-graph* $G = (V, E)$ (Figure 4.5 *a)*), in which each border pixel i of a segment is a vertex $v_i \in V$. Neighboring border pixels (vertices) that belong to different segments, e.g., v_i and v_j, are connected by edges $e_{ij} \in E$. Edges have two weights (Figure 4.5 *b)*), the color similarity of the connected border pixels w_{ij} and a segment edge weight wr_{ij}. Segment edge weights are derived from a segment's normalized color histogram and, if OF is used, their *per-frame OF histograms* [53]. As suggested in [53], the color histograms are generated in the *LAB* color space [146] and are typically divided into 20 bins per color channel *L*, *A* and *B*. Per-frame OF histograms capture the motion for a specific segment for each frame. The mentioned division into frames is necessary, since the motion of an object can change over time. Per-frame OF histograms are further binned with respect to the OF vectors' orientations (as in [53], we typically use 14 bins for the orientation). The main reason for using these, more global descriptors is that they provide a richer description of the intermediate segmentation result than pixel-wise measures [53]. Contrary to the pixel-based differences that are used in the first segmentation step, they also take the distribution of the colors and the motion within a segment into account. In case

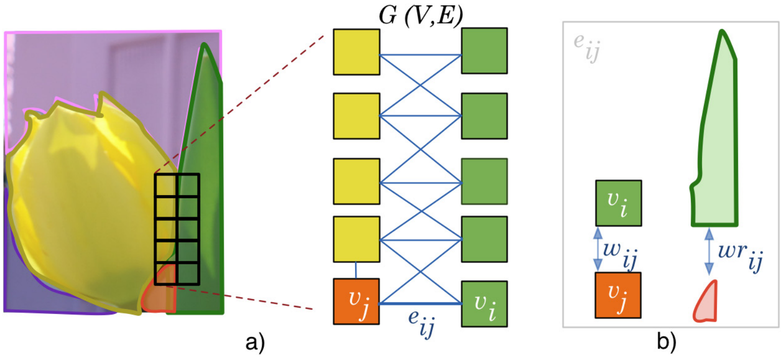


Figure 4.5: Illustration of region-graph. *a)* The over-segmentation of an image (*left*) is represented as a region-graph $G = (V, E)$. The (semi-transparently) colored regions in the image are segments. Region-graph (*right*) for the marked pixels (*squares*). The vertex colors correspond to segment colors. The border pixels i and j of each segment are vertices $v_i, v_j \in V$ and are connected by $e_{ij} \in E$ to neighboring vertices that belong to other segments. *b)* Illustration of edge weights w_{ij} and segment edge weights wr_{ij} for an edge e_{ij}, that represent the similarity of the two connected vertices (i.e., v_i, v_j) and the similarity of their associated segments, respectively.

motion information is not used during the segmentation and propagation process, the segment weights wr_{ij} are defined as the χ^2 distance of the normalized *LAB* histograms of two connected segments. If color and motion similarity of segments are used, the segment weights wr_{ij} are a combination of the χ^2 distances of both histograms ($dc_{ij} \in [0, 1]$ distance of normalized *LAB* histograms, $df_{ij} \in [0, 1]$ distance of normalized per-frame OF histograms) [53]:

$$wr_{ij} = (1 - (1 - dc_{ij})(1 - df_{ij}))^2. \tag{4.2}$$

Hence, the resulting weights are close to zero for segments with similar motion and color properties and otherwise close to one. As in the pixel-wise algorithm, the region-wise algorithm proceeds with traversing edges in ascending order of their weights and merges connected segments, which may contain various disparities (e.g., Figure 4.4 *c)*, Figure 4.2). To ensure that the disparity of the most similar border pixel is propagated, edges are firstly sorted by wr_{ij} and secondly by w_{ij}. We iteratively merge segments and jointly propagate disparities by applying the previously described merging process (Figure 4.2). As suggested in [53], in each iteration the parameters $T_{minsize}$ and τ are scaled by the factor 1.1. To accelerate the algorithm, the iterative process can be stopped before all segments contain pixels that are assigned a disparity. The remaining segments can be assigned to disparities from neighboring segments with disparities that are connected by low-cost edges (in ascending order of their weights).

4.2.3 Disparity Interpolation and Refinement

Having applied our joint segmentation and propagation step (Figure 4.2), every video pixel is associated with a disparity (e.g., Figure 4.4 *d)*, Figure 4.6 *b)*). However, the current disparity video does not capture fine details (e.g., hair) and contains abrupt temporal disparity changes. For instance, a segment with a low disparity in the first and a high disparity in the last frame consists

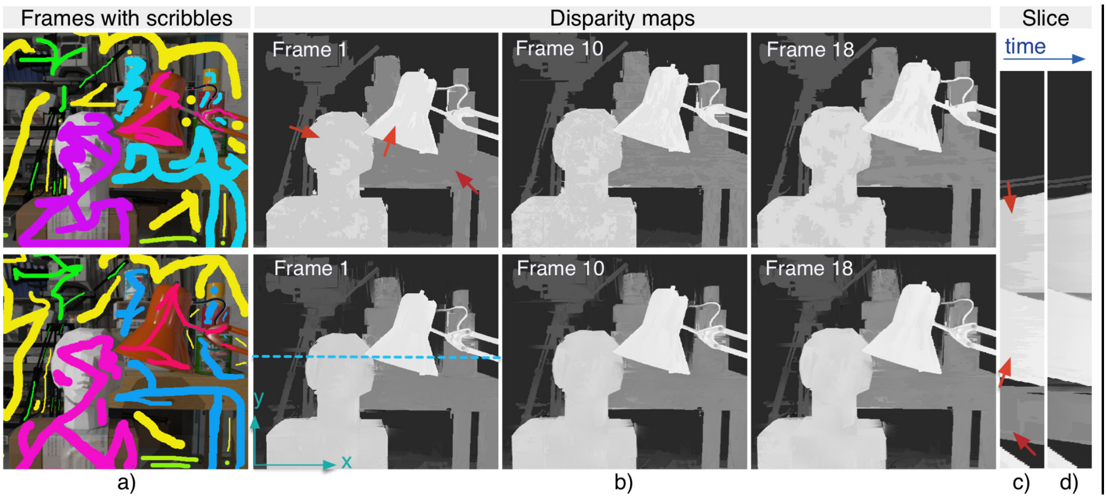


Figure 4.6: Segment-wise disparity interpolation. *a)* Annotated first (*top*) and last frame (*bottom*) of a video. Scribble *hues* encode disparity. The hues of the scribbles on the lamp, table and head change slightly. *b)* Disparity maps (foreground: *white*, background: *black*): Joint segmentation and disparity propagation of the video (*top*) and subsequent segment-wise disparity interpolation (*bottom*). Marked (*red arrows*) objects contain multiple disparities. *c)-d)* Temporal slice for a scan-line (i.e., $y = 231$, *blue line*) in disparity video, before *c)* and after *d)* segment-wise disparity interpolation. Note that *d)* is smoother than *c)*. Original 2D video from [102].

only of pixels with these two disparities. Figure 4.6 shows an example of this case, in which three foreground objects (lamp, table and head) change their disparities over time. The disparities that were defined in the first frame are dominant in the first frames compared to the disparities that were defined in the last frame, and vice versa. In the intermediate frame, these objects contain a more balanced mixture of these disparities. Since the disparity changes computed so far are abrupt, these segments have to be processed further to obtain a spatially and temporally smooth conversion result (Figure 4.2, *disparity interpolation & refinement*).

To interpolate disparities over time, we apply a spatio-temporal filtering technique on the current disparity video. In particular, we extend the *guided filter* [59] (GF), which was originally developed to process images, to perform spatio-temporal video filtering. GF locally smoothes a given image, but preserves its colors at edges detected in a second input image. In order to perform edge- and motion-preserving smoothing of video content, we use three-dimensional instead of the originally proposed two-dimensional filter windows (and kernels). In particular, the smoothed disparity d'_i, which is associated with the not filtered disparity d_i at pixel i in the current disparity video, is determined by a weighted average of d_i's neighboring pixels [59]:

$$d'_i = \sum_j W_{i,j}(I) d_j, \tag{4.3}$$

whose weights are determined from the 2D input video's pixels I_i by:

$$W_{i,j} = \frac{1}{|\omega|^2} \sum_{k:(i,j)\in\omega_k} (1 + (I_i - \mu_k)^T (\Sigma_k + \epsilon U)^{-1} (I_j - \mu_k)). \tag{4.4}$$

Here, ω_k is a spatio-temporal window, with spatial radius r_s and temporal radius r_t, that is centered around a pixel k. i and j are pixels in ω_k. $|\omega|$ is the number of pixels in ω_k. I_i and I_j are color vectors (3×1). μ_k (3×1) is the mean color vector in ω_k and Σ_k (3×3) is the covariance matrix. U is the 3×3 identity matrix. The filter's sensitivity can be controlled by ϵ (a higher ϵ leads to smoother videos). Analogue to its spatial version [35, 59], the spatio-temporal GF can be implemented as a series of local *box filters* (i.e., computes mean of pixels in a filter window) and, thus, has as well a linear time complexity. Specifically, in Equation 4.3's implicit computation that is given below, the summations correspond to box filters implemented as in [35]:

$$a_k = (\Sigma_k + \epsilon U)^{-1}\Big(\frac{1}{|\omega|}\sum_{i\in\omega_k} I_i d_i - \mu_k \overline{d_k}\Big), \tag{4.5}$$

$$b_k = \overline{d_k} + a_k^T \mu_k, \tag{4.6}$$

$$d_i' = \overline{a_i}^T I_i + \overline{b_i}. \tag{4.7}$$

Here, $\overline{d_k}$ is the mean of all disparities d_i in ω_k. $\overline{a_i}$ (3×1) and $\overline{b_i}$ are the means of the linear coefficients a_k (3×1) and b_k in ω_i, i.e., $\overline{a_i} = \frac{1}{|\omega|}\sum_{k\in\omega_i} a_k$ and $\overline{b_i} = \frac{1}{|\omega|}\sum_{k\in\omega_i} b_k$.

We apply the spatio-temporal GF on each spatio-temporal segment that contains a mixture of disparities. It is applied per segment, excluding the remaining pixels of the video. Due to GF's edge-preserving properties, we smooth the disparity video of a segment in areas that have similar colors in the input video and preserve disparities at spatio-temporal edges in the input video. By choosing r_t as large as a quarter of the number of frames contained in a particular segment, we are able to obtain smooth disparity changes over time. In order to preserve edges at segment borders despite the potentially large window size, only disparities within such segments are smoothed. Considering the previously discussed example, i.e., Figure 4.6, it can be seen that this segment-wise interpolation step smoothes temporal disparity changes over time. This is especially visible in the corresponding time slices, in which the abrupt temporal change in disparity in Figure 4.6 *c)* is replaced by a smooth temporal change in disparity in Figure 4.6 *d)*.

A second step (optional) refines the disparity video by applying GF (with smaller r_s and r_t) to the entire disparity video. As above, the disparity video is filtered under the guidance of the input video. This means that the disparities of pixels with similar colors are smoothed, while disparities at spatio-temporal color edges in the input video are preserved. Textured surfaces of constant disparity keep their disparities, while homogenous surfaces with different disparities are smoothed. Furthermore, GF is sensible to fine image structures. Hence, it can reveal details in the disparity video that have not been captured before (Figure 4.4 *d)*, plants).

4.3 Efficient Implementation Using Multicore Technology

Although the 2D-to-3D conversion described above adopts two efficient algorithms [53] and [59], its unoptimized implementation and the large amount of data given by a 2D video can lead to long runtimes (e.g., 450 seconds for a video with 20 frames with a resolution of 600×255 pixels). Thus, we further propose an optimized implementation of the 2D-to-3D conversion algorithm.

As discussed in Section 2.2, salability and efficiency were already addressed by various 2D-to-3D conversion algorithms. A common strategy for reducing their runtimes is a reduction of the given data, e.g., by prior downsampling of the video [56] or segmentation of pixels to groups that are subsequently processed together [44]. Another strategy is the parallel processing of computationally expensive tasks on the GPU [163], which can lead to a major reduction of an algorithm's runtime. We use the latter strategy to reduce the runtime of our semi-automatic 2D-to-3D conversion algorithm. Thus, we re-implement the most computationally expensive parts of the joint segmentation and propagation algorithm (Section 4.3.1) and the interpolation and refinement step (Section 4.3.2) on the GPU, using NVDIA CUDA [108]. CUDA is a parallel computing platform and programming model by NVDIA that allows the simultaneous execution of a large number of (independent) programming tasks, *threads*, on different processing units. After transferring given data from the CPU to the GPU, the same programming task can be simultaneously performed on different portions of the data, i.e., each thread processes a different portion of the given data. To take full advantage of the GPU's abilities, not only an algorithm's decomposition in parallel computations, but also the CUDA architecture, i.e., memory management, has to be taken into account. For example, data that is stored in the GPU's global memory, which can be accessed by all threads, is accessed via 128-byte memory transactions and is accordingly organized into *memory segments*. Ideally, threads running in parallel access different data portions that are, however, stored in the same memory segment (*coalesced memory access*). When accessing different memory segments in each thread (*uncoalesecd memory access*), more memory transactions are necessary and more unused data per thread has to be loaded from the memory (*memory transfer overhead*). This can lead to increased runtimes and, thus, should be avoided by, e.g., optimizing memory-access patterns or adding (unused) data (*padding*) to ideally divide data among threads. While the brief description given above only scratches the surface of the CUDA architecture and its usage for accelerating computer vision algorithms, interested readers are referred to the CUDA programming guide [108] for a more comprehensive description. In context of memory management, it is worth to note that the GPU's memory is typically smaller than the CPU's memory, which hinders the parallel execution of large data like videos. Our optimized implementation further takes this limited capacity of the GPU's onboard memory into account. We propose a simple clip-based 2D-to-3D conversion algorithm for long videos (Section 4.3.3). Our algorithm splits long videos into smaller sub-videos (*clips*) and processes them sequentially while incorporating methods for handling borders between clips.

The main contribution of the optimized version of our 2D-to-3D conversion algorithm is a significant reduction of the runtime while maintaining the quality of its results. In particular, evaluations (Section 4.4) show that it accelerates the 2D-to-3D conversion algorithm by a factor of approximately 226 per frame (resolution of 660×336 pixels).

4.3.1 Joint Segmentation and Propagation

We reduce the execution time of the segmentation and propagation algorithm by implementing computationally expensive parts on the GPU, where they are processed in parallel. As discussed in Section 4.2.2, the joint segmentation and disparity propagation essentially consists of three steps: (1) generating the graph-based representation of the initially given data, (2) pixel-wise over-segmentation and (3) generation of a region-graph for region merging. The region-graph is then iteratively segmented by repeatedly applying steps (2) and (3). Since the over-segmentation in step (2) is generated iteratively, i.e., merging operations depend on the previously performed merging operations, it is not suited for parallelization and remains unchanged.

We parallelize step (1) by processing each pixel of a given video in one thread. Specifically, for each pixel i, we compute edges e_{ij} and their weights w_{ij} for its right, bottom and temporal neighbors j. Thus, the generation of the graph-based representation including the computation of the edge weights is performed in parallel.

The optimization of step (3) is more complex. It comprises the selection of edges between pixels of different segments from the initial edges e_{ij} in (1) and the computation of their additional weights wr_{ij} that are based on χ^2 distances between histograms. First, the *LAB* histograms and the per-frame OF histograms for each segment are initialized in the GPU's global memory and built in parallel. Each pixel is processed in one thread. It increases a bin in the histogram corresponding to its associated segment. To prevent writing conflicts, this is implemented with CUDA's atomic functions. Next, the edges of the region-graph are generated in parallel. As previously described, edges in the region-graph (Figure 4.5) have two weights, the edge weights w_{ij} that express the similarity of the vertices (i.e., border pixels) and were already computed in (1), and the segment edge weights wr_{ij} that express the similarity of their associated segments. The calculation of the segment weights wr_{ij} is very time consuming, since they depend on χ^2 distances between histograms. Thus, reducing the time needed for these distance calculations and their number leads to a significant reduction of the overall runtime. Each edge e_{ij} that connects two border pixels between two specific segments has the same weight wr_{ij}. Redundant histograms distance calculations for these e_{ij} between two specific segments should be avoided. We calculate the histogram distances for each neighboring segment pair only once and store them in a distance buffer. For that purpose, during edge generation, we first determine which unique distances need to be calculated and select the relevant e_{ij} from (1). Next, all required histogram distances can be calculated in parallel, i.e., one distance calculation per thread, on the GPU and stored in the distance buffer. The final segment weights wr_{ij} can be calculated in parallel, i.e., each edge in one thread, using the pre-calculated distances from the distance buffer.

In each of the parallel processing tasks described above the memory-access patterns are coalesced. Thus, no further memory optimization has to be performed.

4.3.2 Disparity Interpolation and Refinement

We reduce the runtime of the interpolation and the refinement steps (Section 4.2.3) by adapting an optimized GPU implementation of GF [59] that is used in [62]. GF can be implemented as a sequence of box filters. Since box filters are *separable*, i.e., can be applied independently for each direction, their filtering operation can be performed in parallel. The box filters are computed by consecutive running sums in horizontal (x), vertical (y) and temporal (t) direction [35]. For each direction, these sums are computed in parallel by processing each row, column or temporal vector in a thread. For a (padded) video that is given in row-major order and stored in global memory, y- and t-direction are coalesced and, thus, not require memory optimization. However, the access pattern is uncoalesced when filtering in x-direction, which can cause elevated runtimes. Thus, we compute the running sums in x-direction by first re-arranging, i.e., transposing, the video frames in global memory and subsequently performing the computations for the x-direction in y-direction in the re-arranged frames. After determining the running sums for the x-direction in this uncoalesced memory access pattern, the video frames are re-arranged to their original form.

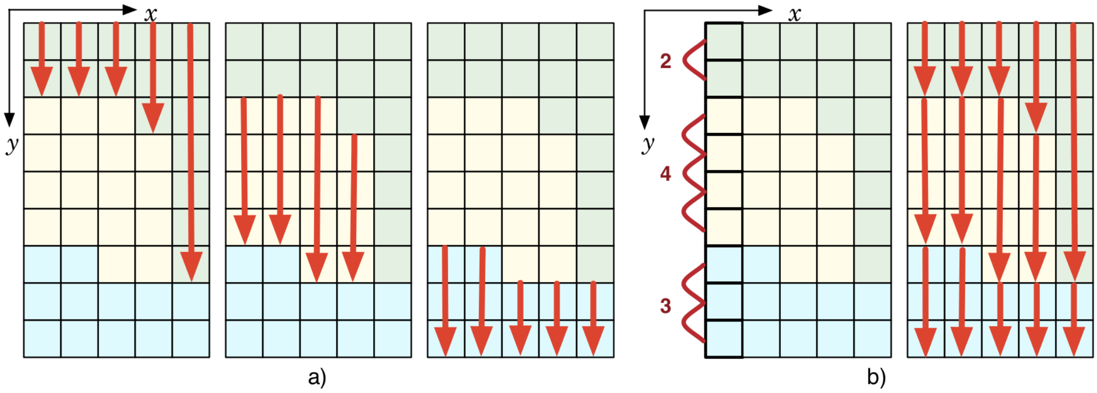


Figure 4.7: Efficient disparity interpolation, i.e., segment-wise filtering in y-direction. *a*) Initial approach: All three segments (*color-coded*) are filtered (*red arrows*) in a separate filtering step. *b)* Optimized approach: First, count pixels per segment (e.g., *red numbers* correspond to first column). Then, filter (*red arrows*) each segment separately in a single filtering step.

While in the refinement step GF's CUDA implementation of [62] is sufficient, this is not the case when performing segment-wise disparity interpolation. In the interpolation step disparities are smoothed over time by applying the GF on each spatio-temporal segment of a video in a separate filter step (Figure 4.7 *a)*) that masks out pixels that are not in the current segment. Hence, the runtime increases proportionally with the number of segments in the video[2]. Thus, we propose an efficient segment-wise GF that is capable of filtering all segments in a single filter step[3] (Figure 4.7 *b)*). For each direction, the segment-wise filter first counts the number of pixels within the current segment and then filters (i.e., box filter) the covered pixels until the segment border of the current segment is reached. Then, we proceed with the next segment until all segments have been processed. In this manner, we are able to perform a segment-wise filtering in a single filtering step that is nearly independent of the number of segments in the video. Note that through the additional effort of counting pixels within segments the runtime slightly increases with the number of segments.

4.3.3 Clip-based 2D-to-3D Conversion

Due to the limited capacity of the GPU's onboard memory only short videos can be processed at once (depending on the video resolution and the GPU's memory). This limitation affects the joint segmentation and propagation as well as the interpolation and refinement steps. We address this problem by partitioning videos that are too long to be processed at once into shorter sub-videos (*clips*) and processing them sequentially on the GPU. To preserve temporal coherence between clips, we employ the standard strategy of including a fraction of the frames from the previous clip in the current one. Specifically, for the interpolation and refinement step, the overlapping part between clips equals

---

[2] If, independent of a segment's scope, the entire video is filtered for each segment, the runtime increases proportionally by the factor of N with each segment. Here, N denotes the number of pixels of a video. (However, in practice, segments that contain only a single disparity can be skipped.)

[3] In the optimized segment-wise filtering step all segments are filtered with the same filter parameters. r_t is set to a quarter of the number of frames contained in the processed video.

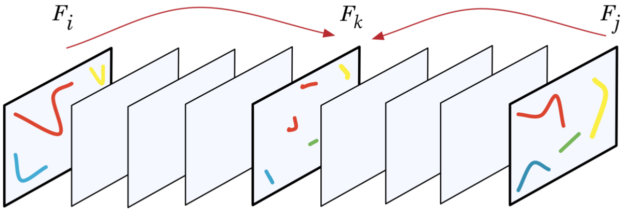


Figure 4.8: Illustration of clip-based disparity propagation. A video with the frames F_i to F_j is split into two clips: F_i to F_k and F_k to F_j. To enable temporal disparity changes, the clip-based 2D-to-3D conversion has to take disparity scribbles (*colored lines*) from the first frame F_i and the last frame F_j into account. Disparity scribbles for the last frame of the first clip, i.e., F_k, can be provided through feature matching (indicated by *red arrows*). See text for details.

the radius of the temporal filter window that was chosen for the respective filter procedure. Thus, the discussed filter operations in the interpolation and refinement step can be performed across clips.

When handling clip-borders for the joint segmentation and propagation step, the last frame of a clip equals the first frame of the following clip. During the segmentation of a clip each pixel receives a segment label, which indicates to which segment it belongs. Since, especially in the context of segment-wise interpolation, segments should extend across clips, the consistency of segment labels between clips has to be considered. Thus, additionally to the frame itself, we also copy the already obtained segment labels and disparities for all pixels in the last frame of the previous clip (i.e., vertices in the graph-based representation of the previous clip) to their corresponding pixels in the first frame of the following clip. For simplicity's sake, in our implementation, the segmentation in one clip is performed independently of the already formed segments, i.e., the computation of a segment's internal variation is only based on the current clip[4] (Section 4.2.2 and Section 4.3.2). Similarly to the propagation of disparities, when merging a vertex with a previous clip's segment label with a vertex that does not posses this information, the merging process additionally copies the segment label to the latter vertex. Therefore, segments can extend across clips. Note that this does not hinder the segmentation from generating a new segment (with a new label that was not available in the previous clip).

In order to propagate disparities between clips and model disparity changes in time, predefined scribble disparities in the first and last frame of each clip are needed. However, these disparities are only available in the first frame F_i and last frame F_j of the whole video (Figure 4.8). Hence, we copy them to each clip's border frames by using SIFT (*Scale Invariant Feature Transform* [98]). Specifically, to copy the scribble disparities from F_i and F_j to an intermediate frame F_k, we proceed as follows (Figure 4.8): First, we extract SIFT features [98] from all three frames. We then match the features between F_i and F_k, using only robust matches[5] [98]. If the pixels at the feature positions in F_i contain disparities, we copy them to the corresponding pixels in F_k. To obtain as many disparities as possible, we include neighboring pixels within a fixed radius (e.g., ten pixels) if their color differences to the corresponding pixels are below a fixed

---

[4] Transferring the internal variations from previous clips might further improve the segmentation and propagation result.

[5] The similarity of robust SIFT feature matches is below a threshold T_f [98]. T_f is automatically set to two times the smallest match similarity that was observed for the SIFT features in a video.

threshold (e.g., $T_c = 0.1$). Next, we copy the scribble disparities from F_j to F_k in the same manner. If a pixel in F_k already received a disparity in the previous step, we use the average of the disparities from F_i and F_j weighted by the respective distance to F_k. It has to be noted that the quality of the 2D-to-3D conversion results is limited by the quality of the feature matching results. In this simple solution, a small number of features typically leads to more propagation errors in the final results (i.e., missing information from scribbles in intermediate frames)[6].

4.4 Experimental Results and Evaluation

We applied our 2D-to-3D conversion algorithm to a variety of monoscopic video shots, including sport scenes (Figure 4.9 *a)* and *b)*), broadcast videos (Figure 4.9 *a)*, *c)* and *d)*, Figure 4.10 *a)* and *b)*), animated (Figure 4.9 *c)*) and self-recorded (using a conventional camcorder) videos (Figure 4.9 *b)*). The videos shown in Figure 4.9 consist of 15, 15, 8 and 71 frames, respectively. The videos shown in Figure 4.10 contain 20 and eight frames, respectively. In each video the first and the last frames had to be annotated to obtain the shown results. On average, 23 frames with an average resolution of 595×334 pixels were annotated with disparity scribbles in the range of $[50, 247]$. We obtained temporally coherent disparity videos, which reflect temporal disparity changes due to objects' motion (Figure 4.10 *a)* and *b)*). As shown in Figure 4.9, Figure 4.10 and Figure 4.11, our results adapt well to the corresponding scenes and objects. They contain homogenous disparity areas with hard disparity edges as well as plausible disparities in slanted or on rounded surfaces. The conversion algorithm is able to capture fine structures and small objects (Figure 4.9 *b)* and *c)*, Figure 4.10 *a)*). It can be seen that our algorithm deals with partial occlusions (Figure 4.10 *a)* and *d)*) and dynamic scenes. In case of full occlusions, the proposed algorithm uses neighboring segments to guess the disparity. However, one full occlusion of an object with constant disparity in the entire video can still lead to plausible results, assuming the respective reference data in the first and last frame are available. We observed limitations when the segmentation algorithm fails and noticed halos in the disparity maps near some color edges (disadvantage of using [59]). In the examples given in Figure 4.11, we used our conversion results *b)* and the given monoscopic video *a)* to generate a second, novel view *c)* for two of the test videos. The shown novel views were generated with a professional stereoscopic software [144] that can also be used to adjust the desired depth effect and create virtual camera perspectives from 2D and disparity data. These examples show that our generated disparity maps are well suited for further processing concerning (auto-)stereoscopic viewing conditions.

Quantitative evaluation and comparison to [56] We quantitatively compare our algorithm to Guttmann et al.'s semi-automatic 2D-to-3D conversion algorithm [56][7] on five test videos that were recorded by a stereoscopic camera. In Appendix A, we provide visual examples for these videos and their reference solutions. To obtain their reference solutions, we apply a stereo method [12], which derives a disparity video for each input video[8]. Similar to the regular user input (Section 4.2.1), we draw scribbles in the first and the last frame of the

---

[6] Lowering T_f can increase the number of matches, but also increases the likelihood of false matches which might lead to propagations of wrong disparities.

[7] We use the parameters that were suggested by the authors in [56].

[8] Note that we only use one of the views for evaluating our algorithm.

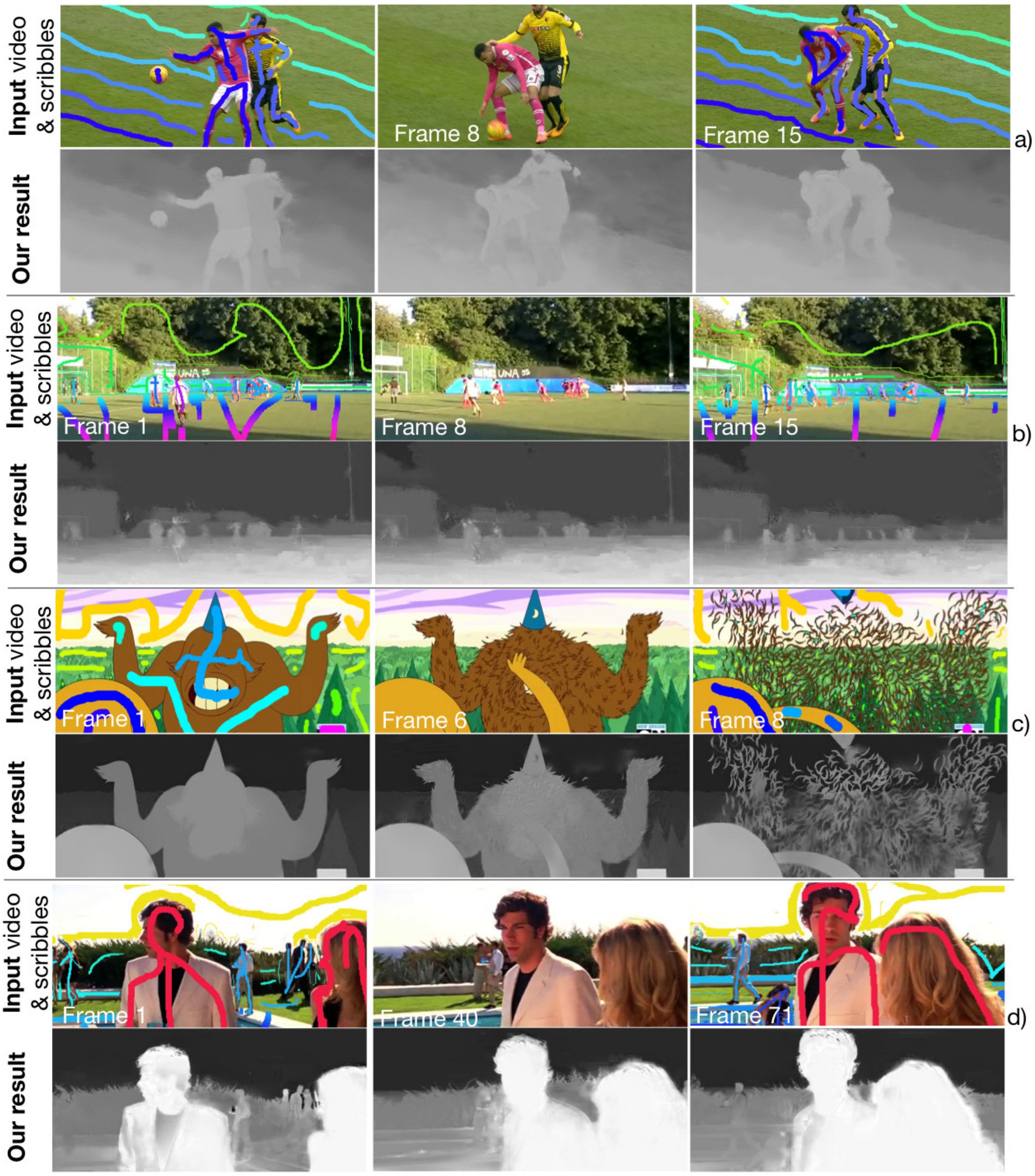


Figure 4.9: Our 2D-to-3D conversion results. Input video with scribbles (*top*): Scribble *hues* encode disparity. Our obtained disparity maps (*bottom*): Foreground: *bright*, background: *dark*. Copyright of original videos: *a)* BBC. *c)* Cartoon Network. *d)* Warner Bros.

videos. Likewise the scribbles define which disparities should be propagated. To make a comparison with the reference solution possible, the disparities for the marked pixels

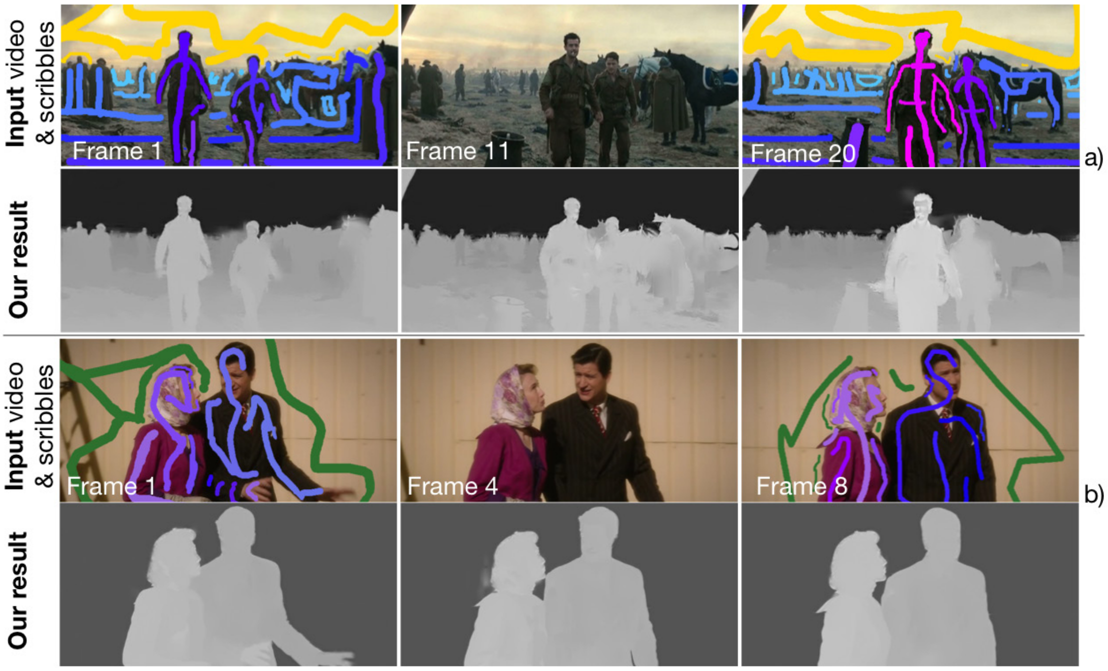


Figure 4.10: Our 2D-to-3D conversion results with temporal disparity changes. Input video with scribbles (*top*): Scribble *hues* encode disparity. Our disparity maps (*bottom*): Foreground: *bright*, background: *dark*. Our results capture the temporal disparity changes of the persons that move towards the camera. Copyright of original videos: *a)* Universal Pictures. *b)* ABC Studios.

Table 4.1: Quantitative evaluation and comparison to [56]. Our algorithm with and without the usage of optical flow (OF) (*left*) and our implementation of Guttmann et al.'s algorithm [56] (*right*) are compared to the reference solution [12]. The table lists the mean squared error (MSE) (i.e., averaged over all pixels) of the disparities. The MSEs are multiplied by 100.

Video	**MSE**, with OF	**MSE**, without OF	**MSE** for [56], with OF
City	0.32	0.56	10.30
Parade	0.23	0.23	6.87
Palace	1.01	1.39	8.51
Stairs	0.19	0.31	1.56
Football	0.23	0.33	4.95

are defined as the disparities of the reference data at the scribble positions[9]. Table 4.1 lists the *mean squared error* (MSE) averaged over all pixels of a video for our 2D-to-3D conversion algorithm and our implementation of Guttmann et al.'s algorithm [56]. More precisely, the MSE between each pair of corresponding pixels i, one in the reference

---

[9] As mentioned in Section 4.2.1 we work with disparity maps that were normalized with respect to the entire video shot, and, hence, also propagate normalized disparities, i.e., $\in [0, 1]$.

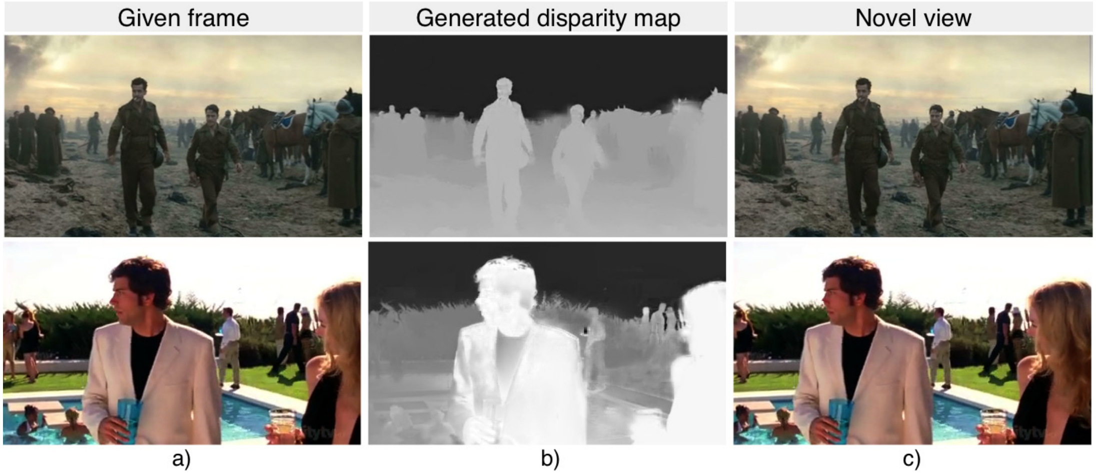


Figure 4.11: Novel views generated from proposed conversion results. *a)* Input frame. *b)* Corresponding generated disparity map (foreground: *bright*, background: *dark*). *c)* Novel view generated by using *a)* and *b)* with a software that can be used for novel view generation [144]. The conversion results used in this example correspond to videos shown in Figures 4.9 and 4.10. Copyright of original video: *First row:* Universal Pictures. *Second row:* Warner Bros.

solution $d_{ref,i} \in [0,1]$ and one in the conversion result $d'_i \in [0,1]$, is determined by:

$$MSE = \frac{1}{N} \sum_{i \in d_{ref}} (d'_{ref,i} - d_i)^2, \tag{4.8}$$

where N is the number of pixels in the reference solution. We evaluate our algorithm in two versions, one version with and one version without making use of OF in the graph-based representation of the video and as a feature in the segmentation. When employing OF, we use the same OF fields as [56], i.e., dense OF fields obtained by [110]. Therefore, the performed comparison is independent of the motion information.

Figure 4.12 shows the results for two test videos (Figure 4.12, *City* (*top*), *Stairs* (*bottom*)). It can be seen (Figure 4.12, Table 4.1) that our algorithm produces disparity maps of high quality, i.e., which are close to the reference solution. More importantly, we outperform the previous work by Guttmann et al. [56] using the same user input for both algorithms (Figure 4.12, Table 4.1). Our algorithm adapts better to the underlying scenes and generates disparity maps in which disparity edges at object borders are preserved. This is also true for longer videos (e.g., Figure 4.12, *Stairs* with 45 frames). Contrary, due to the increased sparsity of the user input in longer videos (i.e., number of annotated pixels compared to not annotated ones) [30, 66], Guttmann et al.'s algorithm [56] severely over-smoothes their disparity maps. As an additional advantage over [56], we are able to spare OF and still obtain similarly good results (Table 4.1). When we examine the error maps in detail, we notice limitations at borders of moving objects (Figure 4.12 *b)-d)*) and when the segmentation algorithm encounters problems, such as objects with similar color (and motion). Considering the former limitation, the refinement step reduces the error at motion discontinuities. In this context, it is important to recall that the comparison is performed with a stereo generated reference solution which, as well, might contain disparity errors near objects borders. Figure 4.12 *b)-d)* illustrates that our approach is able to reflect disparity changes due to object's motion in its result. Note that for the object that moves in depth (e.g., Figure 4.12 *c)*, *Stairs*, person), the difference in the error maps is caused from different velocities of the disparity changes in the reference solution and our disparity videos. Since we assume linear disparity changes from the first to the last frame, deviations from this model (e.g., irregular movement) are not taken into account.

Quantitative evaluation of region merging step's influence on conversion quality. In [53], Grundmann et al. point out that their additionally performed region merging step leads to better segmentation results than a segmentation without it. In this experiment, we test if our 2D-to-3D conversion results behave analogously. To this end, we compare two versions of our 2D-to-3D conversion algorithm: (1) the *full conversion* algorithm as described from Section 4.2.1 to Section 4.2.3 and (2) a faster *shortened version* that is only based on pixel-wise merging, i.e., without further merging of segments according to a region-graph. Instead we merge low-cost edges in the pixel-graph to achieve a conversion that is based on the same number of segments as in (1). Similar to the previous evaluation, our conversions are performed using normalized ground truth (GT) values at scribble positions. Motivated by [76], we place scribbles automatically within spatially connected regions of constant ($T0$) disparity in the GT (*disparity regions*) and within disparity regions that can contain disparity edges that differ by one ($T1$). Hence, placing a *dot scribble* (DS), i.e., a dot shaped brush stroke (e.g., Figure 4.14), in the center of each region results in two sets of input scribbles, i.e., *DS T0*, which is denser, and *DS T1*, which is sparser. Analogously, we generate *thinning scribbles* (TS) by thinning disparity regions, which results in scribbles that more closely resemble typical user-annotations (e.g., Figure 4.13).

In Table 4.2 (*top*), we compare our converted disparity maps to GT disparities from the Middlebury stereo benchmark [134] (i.e., 35 left views). For these images, the following parameters are used for (1) and for (2): segmentation parameters $\{\tau = 0.3, T_{minsize} = 20,$ (7 iterations for (1))$\}$, segment-wise GF parameters $\{r_s = 9, \epsilon = 0.0001\}$ and GF parameters for refinement $\{r_s = 3, \epsilon = 0.0001\}$. In Table 4.2 (*bottom*), the comparisons are performed on five videos from the Sintel dataset[10] [27] and two videos from the Tsukuba dataset [102] that were provided with depth GT [27] and disparity GT [102]. Appendix A provides visual examples for the image, videos and their GT. Scribbles were automatically placed in the first and last frames of each video and are initialized with normalized and inverted GT depths [27] or normalized GT disparities [102]. For these videos, following parameters are used for (1) and for (2): segmentation parameters $\{\tau = 0.3, T_{minsize} = 110,$ (7 iterations for (1))$\}$, segment-wise GF parameters $\{r_s = 12, \epsilon = 0.0001\}$ and GF parameters for refinement $\{r_s = 9, r_t = 2, \epsilon = 0.0001\}$.

Table 4.2 confirms that, in case of the tested images and videos, the full conversion algorithm with region merging is closer to the GT than the shortened conversion without region

---

[10] The authors provided us with color videos and corresponding rendered depth data, before officially extending the Sintel Flow dataset [27] to additionally provide depth and disparity ground truth.

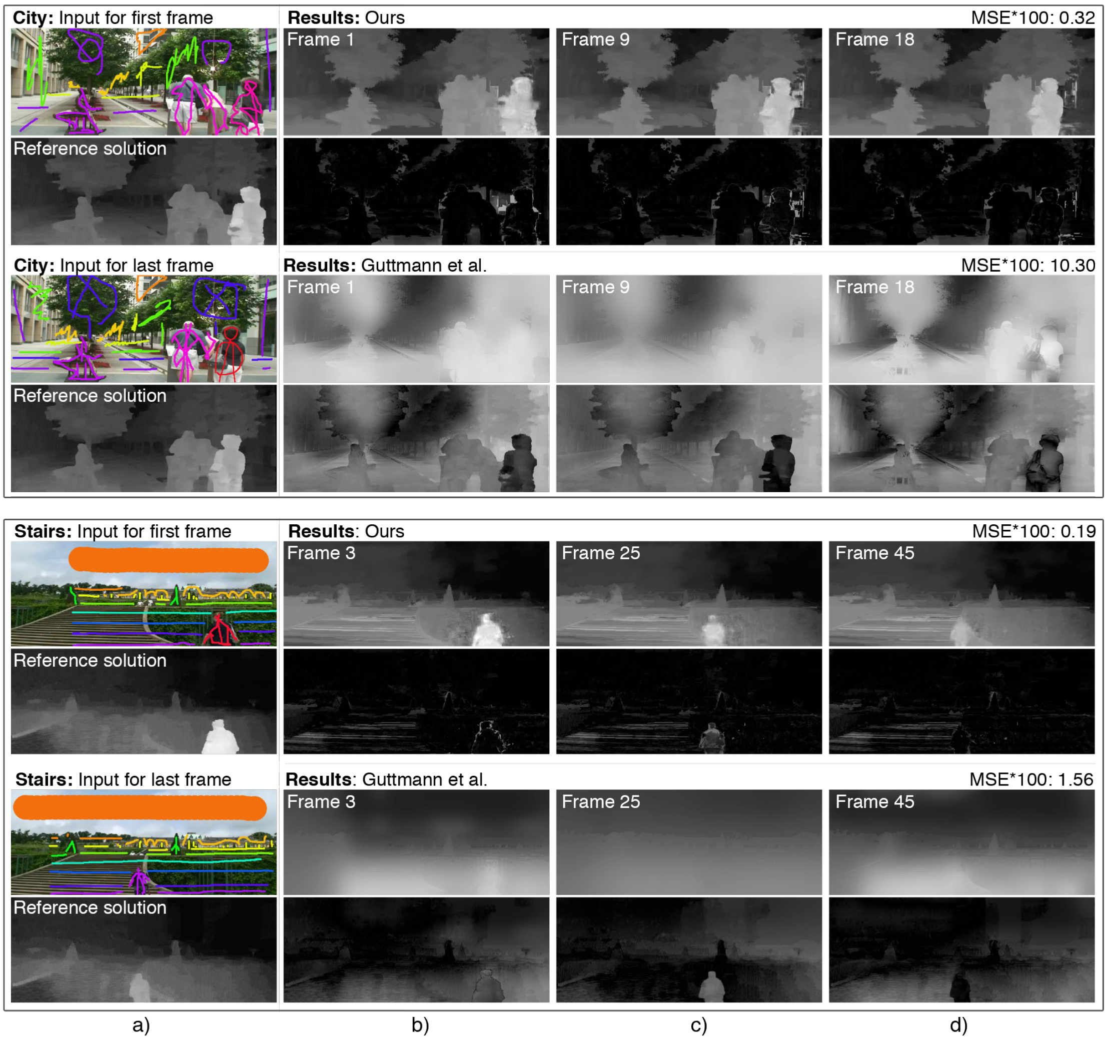


Figure 4.12: Quantitative evaluation and comparison to [56]. Per video (*City*, *Stairs*): *a)* Scribbles (*first* and *third row*) and reference solution (*second* and *fourth row*) of first and last frame. In the reference solution: *White* fore- and *black* background. *b)-d)* Our result (*first row*) and corresponding error map (*second row*). In the error map, *dark* and *white* pixels denote low and high errors, respectively. Disparities (*third row*) obtained by our implementation of [56] and corresponding error map (*fourth row*). Our result: hard edges. [56]: Over-smoothed edges.

merging. For the dataset that contains images (Table 4.2, *top*) and for the dataset that contains videos (Table 4.2, *bottom*) the MSEs of the full conversion are throughout lower than those of the shortened conversion. Figure 4.13 shows conversion results obtained with the full and the shortened 2D-to-3D conversion algorithm for TS T0 and TS T1. It can be seen that the disparity maps generated by the full conversion adapt better to the objects in the scene and discriminate between ambiguous image regions (e.g., color similarities between

Table 4.2: Quantitative evaluation of region merging step's influence for images from [134] and videos from [27, 102]. Results from the full and the shortened conversion algorithms are compared to GT. The table lists the MSE averaged over the dataset and multiplied by 100. The conversion was performed based on computer-generated scribbles (*TS* and *DS*) (see text for details).

MSE	**Full conversion**				**Shortened conversion**			
2D content	**TS T0**	**TS T1**	**DS T0**	**DS T1**	**TS T0**	**TS T1**	**DS T0**	**DS T1**
images [134]	0.08	0.31	0.53	2.97	0.08	0.42	0.64	3.71
videos [27, 102]	0.65	0.77	0.94	2.97	0.67	0.88	1.09	3.91

doll and map). For videos, this can also be observed in Figure 4.14. In this particular scene color similarities (e.g., brown rock and brown hair or beard) and large motions pose a challenge to the conversion algorithm. These observations indicate that the segment-based similarity measure that is used in the full conversion algorithm is more stable concerning the mentioned color similarities than the local pixel-wise measures that are used in the shortened conversion algorithm. Hence, in case of our task of 2D-to-3D conversion, Grundmann et al.'s [53] observation is confirmed. Furthermore, Table 4.2 demonstrates that denser scribble input (i.e., T0) leads to better conversion results than sparser scribble input (i.e., T1).

Computational efficiency evaluation of optimized 2D-to-3D conversion algorithm. We evaluate the optimized 2D-to-3D conversion algorithm from Section 4.3 by comparing its runtimes to its not optimized version from Section 4.2 and by comparing the conversion results against GT. For these experiments we used a PC with an Intel Xenon E5 processor with 3.6GHz, 32GB RAM and a NVIDIA GeForce GTX 680 GPU. The runtimes of the joint segmentation and propagation step and the runtimes of the interpolation step of our optimized GPU implementation (C++/CUDA) and the unoptimized implementation (C++) are compared separately. On the one hand, the evaluation in terms of runtimes is performed on a video with 20 frames (600 × 255 pixels), a downsampled sub-video from *Alley1* (Appendix A, Sintel dataset[11] [27]). For this video we also provide runtimes consumed by the interpolation step for different segmentation granularities. On the other hand, the computational efficiency is evaluated on eight additional videos without considering different segmentation granularities.

For the joint segmentation and propagation step, the unoptimized and the optimized implementation consume 450.10 and 9.95 seconds, respectively when processing the single test video mentioned above. Hence, for this video, the optimized implementation is 45 times faster than the unoptimized one. Table 4.3 gives a more detailed efficiency evaluation of this step by listing runtimes for eight videos. In this table the optimized implementation is, on the average, 35 times faster than the unoptimized implementation.

Figure 4.15 reports the runtimes of a CUDA[12] implementation of the initial segment-wise

---

[11] At the time the evaluations were performed, no GT data was available for this test video.

[12] In the initial, not optimized implementation of the proposed 2D-to-3D conversion algorithm, the segment-wise interpolation is implemented in C++. However, in this experiment a CUDA implementation is used to highlight the algorithmic contribution of its optimized version. Figure 4.15 lists the runtimes of the CUDA implementation. The runtimes of the C++ implementation of the initial segment-wise interpolation are 1945 and 4200 seconds for a segmentation into 1063 and 2295 segments, respectively.

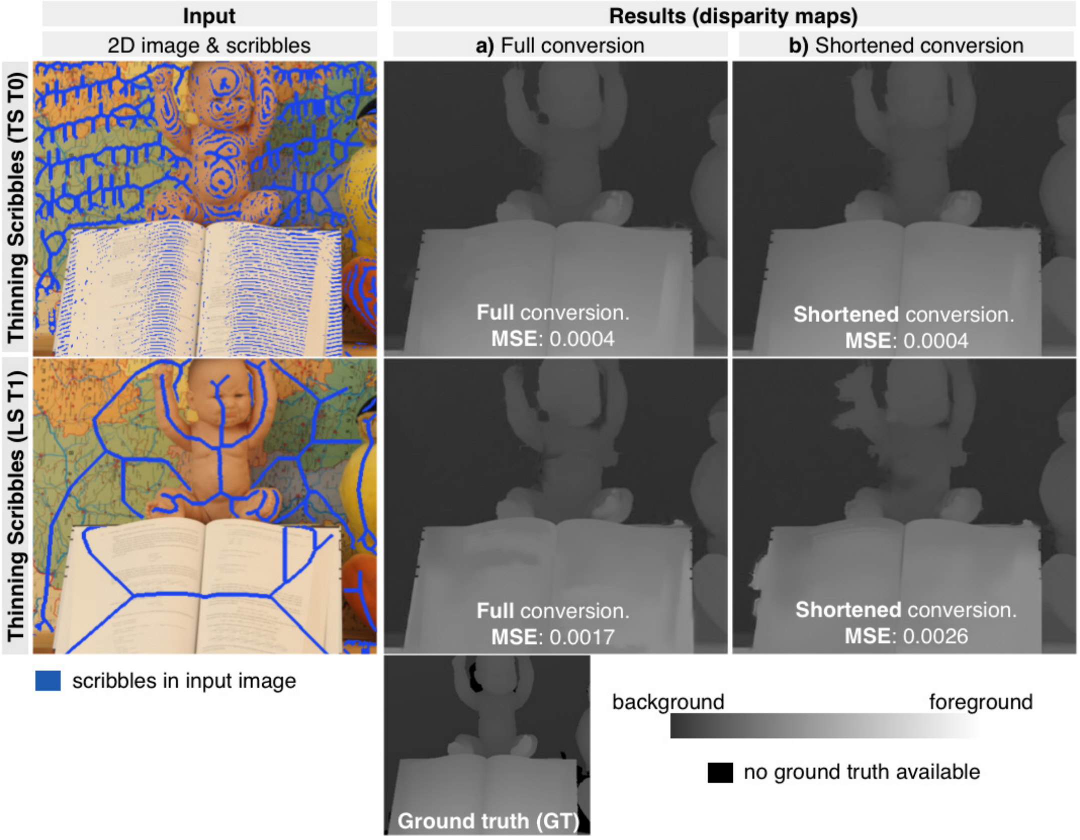

Input
2D image & scribbles
Results (disparity maps)
a) Full conversion
b) Shortened conversion
Thinning Scribbles (TS T0)
Thinning Scribbles (LS T1)
Full conversion. MSE: 0.0004
Shortened conversion. MSE: 0.0004
Full conversion. MSE: 0.0017
Shortened conversion. MSE: 0.0026
scribbles in input image
Ground truth (GT)
background
foreground
no ground truth available

Figure 4.13: Evaluation of region merging step's influence. Results for *Baby2* of our *a)* full and *b)* shortened 2D-to-3D conversion algorithm. The results are based on GT disparities (*bottom*) at computer-generated scribble positions (*blue*), i.e. *TS T0* and *T1*. Original image from [134].

interpolation approach (i.e., each segment is filtered in a separate filtering step), and the optimized CUDA implementation (i.e., all segments are filtered in one filtering step) when processing the single test video mentioned above. Both segment-wise GF implementations are applied to segmentations of the original video into 1063 and 2295 segments (Figure 4.15). Table 4.4 list these runtimes for eight additional test videos, however, without considering different segmentation granularities. As shown in Figure 4.15 and Table 4.4, the optimized implementation consumes significantly less time than the initial approach, in which the runtime of segment-wise filtering increases proportionally with the number of segments. Contrarily, the runtime of the optimized approach increases only slightly with an increasing number of segments (caused by counting pixels within a segment).

When considering the single video mentioned in the beginning the optimized 2D-to-3D conversion algorithm, i.e., the joint segmentation and propagation and the optimized segment-wise interpolation steps implemented in CUDA, consumes 10.57 seconds according the runtimes given above (i.e., joint segmentation and propagation: 9.95 seconds, optimized segment-wise interpolation for 0.62 seconds in case of 1063 segments). The

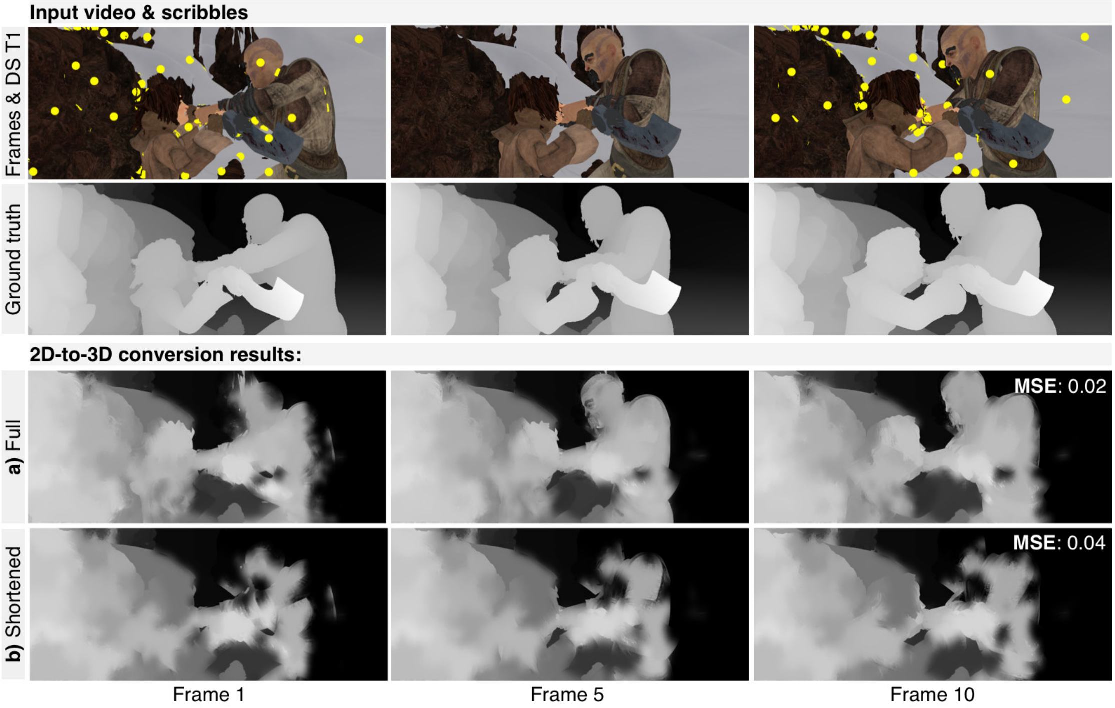


Figure 4.14: Evaluation of the region merging step's influence. Results for *Ambush5* of our *a)* full and *b)* shortened 2D-to-3D conversion algorithm. These results are based on GT values at computer-generated scribble positions (*yellow*), i.e., *DS T1*. Front: *bright*, Back: *dark*. Original video from [27].

unoptimized 2D-to-3D conversion algorithm, i.e., the joint segmentation and propagation and the initial segment-wise interpolation septs implemented in C++, consumes 2395.10 seconds (i.e., joint segmentation and propagation: 450.10 seconds, optimized segment-wise interpolation for 1945 seconds in case of 1063 segments). Thus, for this video, our optimized 2D-to-3D conversion accelerates the unoptimized one by a factor of 266.

Figure 4.16 additionally compares four implementations of the box filter, which is a principal component of GF [59]. We list runtimes for a C++ and three CUDA implementations: (1) a not memory-optimized naive implementation (uncoalesced memory-access), (2) an implementation that uses the GPU's faster texture memory to achieve computational performance gains despite the uncoalesced memory-access in x-direction, and (3) our implementation that transposes video frames for coalesced memory-access in x-direction. The box filter implementation that performs memory optimization via texture memory is provided by NVIDIA Computing SDK 4.1 [108]. Since the texture memory is restricted to reading after its initialization, its usage causes frequent data transfers between CPU and GPU when applying multiple box filters in a row (as it is done in GF). Contrary, our implementation is entirely performed on the GPU and has lower runtimes in Figure 4.16.

Finally, Table 4.5 lists the runtime of the entire optimized 2D-to-3D conversion algorithm (i.e., joint segmentation and propagation, segment-wise interpolation and the optional

Table 4.3: Computational efficiency evaluation of joint segmentation and propagation. Comparison of runtimes between the unoptimized C++ implementation and the optimized CUDA implementation of the joint segmentation and propagation step. The joint segmentation and propagation step of the unoptimized and optimized algorithm take the same input videos, user input, disparities and parameters. The table lists the runtimes in seconds.

Video	**Resolution**	**Clips**	**Runtime**, unoptimized	**Runtime**, optimized
City	699 × 282 × 19	1	439.82	11.39
Parade	689 × 282 × 11	1	286.20	6.93
Palace	702 × 278 × 10	1	267.80	6.69
Stairs	702 × 278 × 20	1	589.30	11.73
Football	669 × 282 × 21	2	974.10	26.74
Child	600 × 338 × 21	2	530.10	27.76
Tsukuba50	640 × 480 × 17	1	728.96	15.24
Tsukuba380	640 × 480 × 18	1	705.05	15.96

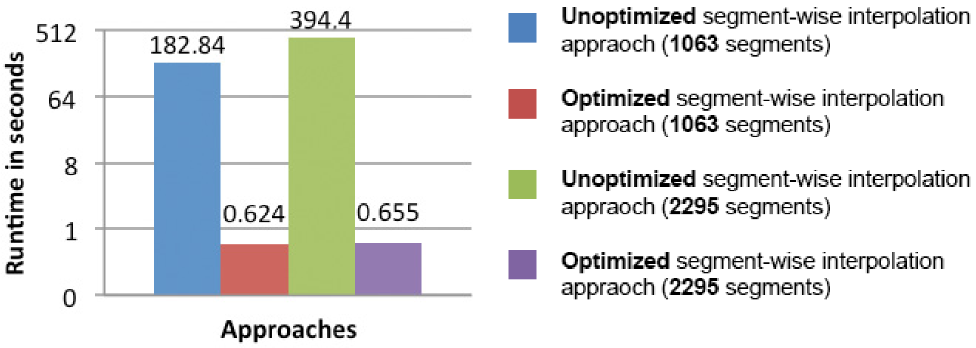


Figure 4.15: Computational efficiency evaluation of segment-wise interpolation. Runtime comparisons between a CUDA implementation of the initial segment-wise interpolation approach (i.e., each segment is filtered in a separate step) and a CUDA implementation of the optimized segment-wise interpolation approach (i.e., all segments are separately filtered in one filtering step). The comparison is performed for a segmentation into 1063 and into 2295 segments.

refinement step) for 11 test videos. On average, for a video shot with a resolution of 653 × 363 pixels per frame, our optimized algorithm has a runtime of 0.99 seconds per frame (average of Table 4.5). In this evaluation, videos that are too long to be processed on the GPU at once are partitioned into clips (e.g., *Interview* is divided into six clips). In these cases, the additionally performed feature matching leads to slightly longer runtimes.

Quantitative evaluation of optimized 2D-to-3D conversion algorithm. Since a main concern is not only to reduce the runtime of the proposed 2D-to-3D conversion algorithm, but also at the same time to maintain the quality of its results, the disparity videos generated with the optimized version of the algorithm are evaluated as well. We compare the results of our optimized conversion algorithm with reference solutions from three different test sets. (Appendix A provides visual examples of these videos and their reference solutions.):

Table 4.4: Computational efficiency evaluation of segment-wise interpolation. Comparison of runtimes (in seconds) between the CUDA implementation of the initial segment-wise interpolation approach (each segment is filtered in a separate filtering step) and the CUDA implementation of the optimized segment-wise interpolation approach (all segments are separately filtered in one filtering step). Note that these approaches for each video are applied on the same number of segments.

Video	**Resolution**	**Clips**	**Runtime**, unoptimized	**Runtime**, optimized
City	699 × 282 × 19	1	1.36	0.62
Parade	689 × 282 × 11	1	0.36	0.14
Palace	702 × 278 × 10	1	0.34	0.13
Stairs	702 × 278 × 20	1	0.75	0.22
Football	669 × 282 × 21	2	1.70	0.44
Child	600 × 338 × 21	2	2.15	1.06
Tsukuba50	640 × 480 × 17	1	1.11	0.27
Tsukuba380	640 × 480 × 18	1	1.12	0.36

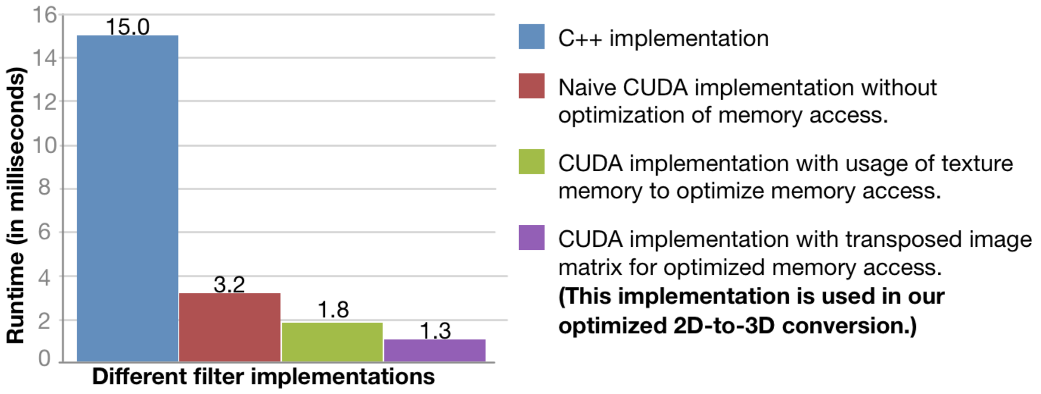


Figure 4.16: Computational efficiency evaluation of memory optimization performed in segment-wise interpolation. The chart lists the runtimes (in milliseconds) of different box filter implementations in CUDA. The different filter implementations were applied on an image with the resolution of 1024×1024 pixels. Note that GF [59], which is used in the interpolation step of the 2D-to-3D conversion algorithm, is implemented as a series of box filters.

- *Recorded stereo dataset*, which consists of the five test videos that we used to evaluate the unoptimized algorithm (Table 4.1 and Figure 4.12), i.e., *City*, *Parade*, *Palace*, *Stairs*, *Football*. The reference solutions are stereo estimated [12] disparity videos.
- *Long video dataset*, which consists of three videos (with 21 to 101 frames) taken from [20, 42], i.e., *Child* (Figure 4.17), *Head*, *Interview*. These videos are computer-generated (*Child*, *Head*) or recorded (*Interview*) and provided with GT disparity and depth maps, respectively. The depth map of *Interview* was recorded with a special camera that is also able to record depth [42]. GT from [20] is only provided for a few frames and, thus, conversion results are only evaluated at these frames.
- *Tsukuba dataset*, which consists of three videos that were taken from the new Tsukuba

Table 4.5: Quantitative evaluation of optimized 2D-to-3D conversion algorithm. MSE to reference solutions and runtime (in seconds) for the optimized 2D-to-3D conversion approach (i.e., including refinement step, excluding calculation time of OF). MSEs are multiplied by 100.

Video	**Resolution**	**Clips**	**MSE**, with OF	**MSE**, without OF	**Runtime**
City	699 × 282 × 19	1	0.08	0.09	12.61
Parade	689 × 282 × 11	1	0.12	0.13	7.40
Palace	702 × 278 × 10	1	0.17	0.17	7.16
Stairs	702 × 278 × 20	1	0.12	0.12	12.73
Football	669 × 282 × 21	2	0.08	0.10	29.00
Child	600 × 338 × 21	2	0.07	0.07	31.46
Head	600 × 330 × 81	4	0.44	0.52	51.69
Interview	600 × 480 × 101	6	0.37	0.43	111.85
Tsukuba50	640 × 480 × 17	1	0.02	0.02	16.61
Tsukuba380	640 × 480 × 18	1	0.16	0.18	17.44
Tsukuba1	640 × 480 × 100	8	0.11	0.11	113.82

dataset [102], i.e., *Tsukuba1*, *Tsukuba50* (Figure 4.6), *Tsukuba380.* These computer-generated videos are provided with corresponding GT disparity videos.

To evaluate the quality of our results we compare them to their respective reference solutions. We employ the same strategy as above to enable the comparison of our propagation result with the respective reference solution, i.e., the conversion algorithm propagates the reference solution at user-provided scribble positions instead of the disparity given by the scribbles. In case of the recorded stereo dataset, we use the same input data, including scribbles, as in the evaluation of the not optimized 2D-to-3D conversion algorithm (i.e., Table 4.1). Table 4.5 lists the MSE averaged over all pixels of the respective video. The videos were converted with the optimized 2D-to-3D conversion algorithm with and without OF (from [110]). Figure 4.17 exemplarily shows the results for the video *City* (with OF). As in the unoptimized conversion, in its optimized version only a few scribbles are sufficient to generate temporal consistent disparity maps that are near to the reference data (Figure 4.17, Table 4.5). When further comparing the MSEs of the optimized algorithm (i.e., Table 4.5) with the MSEs of the unoptimized algorithm (i.e., Table 4.1), it can be seen that both implementations of the algorithm obtain results of high conversion quality, i.e., results which are close to their corresponding reference solutions.

Evaluation of clip-based 2D-to-3D conversion algorithm. We further evaluate the impact of our clip-based processing on the quality of the conversion results and the runtime. To this end, we apply our algorithm to six videos[13]. For each video we perform three 2D-to-3D conversions: (1) conversion as a whole video and (2) clip-based conversions for partitions into two clips and (3) into three clips (Table 4.6). In all investigated videos the overall runtime increases with the number of clips. This is caused by the additionally performed feature matching that propagates scribble disparities to the first and last frame of each

---

[13] The videos from our long video dataset, i.e., *Child*, *Head* and *Interview*, and *Tsukuba380* were excluded from this experiment. Due to their memory requirements, they can only be processed in multiple clips.

Table 4.6: Quantitative evaluation and computational efficiency evaluation of clip-based 2D-to-3D conversion algorithm (with OF). MSE (multiplied by 100) and runtime (in seconds) for conversion as a whole video (*1C*) and clip-based for partitions into two (*2C*) and three clips (*3C*).

Video	**MSE**			**Runtime**		
	1C	**2C**	**3C**	**1C**	**2C**	**3C**
City	0.08	0.11	0.16	12.61	13.24	14.01
Parade	0.12	0.17	0.27	7.40	9.15	10.82
Stairs	0.12	0.72	0.21	12.73	12.87	13.51
Palace	0.17	0.17	0.20	7.16	8.83	10.71
Tsukuba50	0.02	0.02	0.02	16.61	17.10	18.20
Tsukuba380	0.16	0.16	0.16	17.44	18.12	19.07

clip. Table 4.6 further indicates that our clip-based algorithm works better for computer-generated videos than for video recordings. In case of the recorded videos *City*, *Parade*, *Stairs* and *Castle* the MSE increases with the number of clips, while remaining unchanged for the computer-generated videos *Tsukuba50* and *Tsukuba380*.

4.5 Summary

This chapter has presented a semi-automatic 2D-to-3D conversion approach for dynamic 2D video shots. Our main contribution was to adopt a segmentation algorithm for disparity propagation, which enabled us to process video shots containing camera and object motion. The joint segmentation and propagation approach supports multiple disparities within segments and ensures hard disparity edges at segment borders. A subsequent step interpolates disparities within segments over time and optionally refines the final disparity video, enforcing disparity edges to be consistent with color edges in the input video. Quantitative evaluations have demonstrated that our proposed algorithm generates disparity maps of high conversion quality. Our conversion-generated disparity maps capture the structure of the corresponding 2D content, contain plausible disparities for slanted surfaces and capture smooth disparity changes over time. Our experiments, visually and quantitatively, show that we have outperformed a well-established semi-automatic 2D-to-3D conversion approach on a set of recorded videos using the same user input. In this context, a key advantage of our approach was a significant reduction of over-smoothing in our 2D-to-3D conversion results.

This chapter has further discussed an optimized implementation of the presented semi-automatic 2D-to-3D conversion algorithm. In this context, our first contribution was a significant reduction of the runtimes while maintaining the quality of the resulting disparity videos. By implementing computationally expensive parts of our algorithm on the GPU, we have achieved an acceleration of the conversion algorithm by a factor of 226 for a video with the resolution of 600×255 pixels. As a second contribution, we have generated an optimized implementation of the underlying segmentation algorithm [53] and an efficient method for segment-wise filtering that is based on an existing edge-preserving filter [59]. Furthermore, we have used a simple approach for overcoming the limitation posed by the GPU's onboard memory. It partitions long videos into clips, which are processed sequentially, while incorporating methods for handling borders between clips. Since it relies on feature matching, the quality of the results depends on the quality of the matching results.

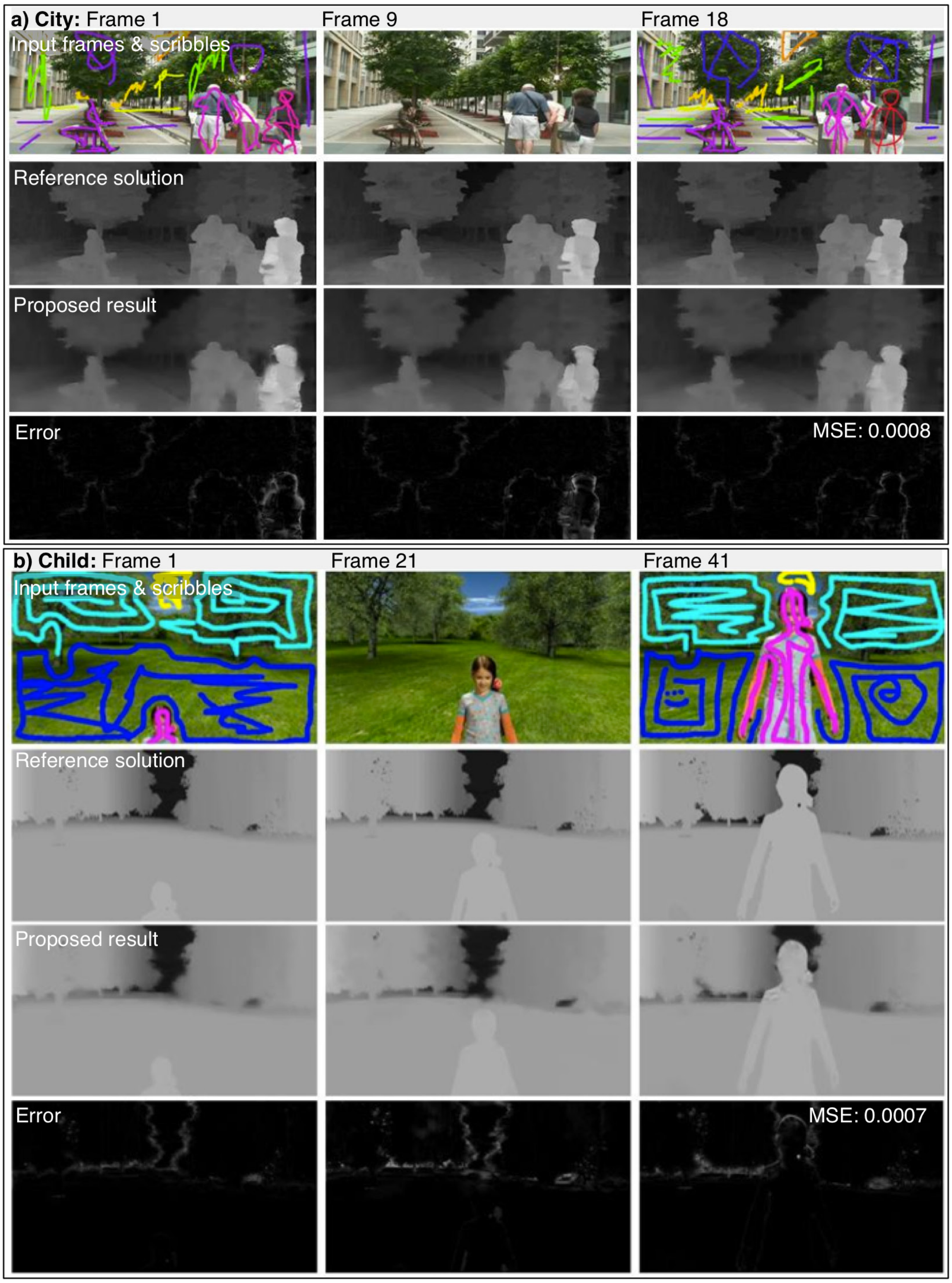


Figure 4.17: Evaluation of optimized 2D-to-3D conversion algorithm. For *a) City* and *b) Child*: *First row*: Frames of input video with scribbles in its first and last frame. *Second row*: Reference solution (*white*: foreground, *black*: background). *Third row*: Our conversion results. *Fourth row*: Corresponding error (*dark* small and *white* large errors). *b)* Original video from [20].

CHAPTER 5

Cost Volume Filtering for Video Segmentation and 2D-to-3D Conversion

5.1 Introduction

This chapter describes the second semi-automatic 2D-to-3D conversion algorithm that is proposed in this thesis. In some similarity to our previous segmentation-based 2D-to-3D conversion algorithm (Chapter 4) it also exploits video segmentation techniques to preserve disparity edges at object borders. Contrary to our previous segmentation-based 2D-to-3D conversion algorithm, which uses an automatic video segmentation technique, an interactive video object segmentation algorithm based on *cost volume filtering* (CVF) is used. In this chapter, we first propose a fast interactive video object segmentation algorithm that is able to segment a video at interactive rates (i.e., 250 fps for frames with a resolution of 620×360 pixels) and then extend it to perform 2D-to-3D conversions. This extension considers special requirements of the application of 2D-to-3D conversion (e.g., multiple smooth disparities within objects instead of fixed assignments to fore- and background). Both algorithms (the proposed interactive object segmentation algorithm and the proposed 2D-to-3D conversion algorithm) are embedded in an efficient optimization framework, i.e., the *cost volume filtering* framework [125], that has been used to solve several label-based optimization problems in the field of computer vision, including stereo matching, OF estimation and image segmentation [62, 82, 125].

The interactive video object segmentation algorithm that is proposed in this thesis (Section 5.2.1) allows users to extract objects from a video using only a few foreground scribbles. Its main contribution is the extension of the image segmentation approach via CVF of [125] to the temporal domain. More precisely, the initial CVF approach [125] solves the binary image segmentation problem in three steps: (1) First, based on color models that were initialized through foreground scribbles, Rhemann et al. generate a cost map (or in case of a video a cost volume) that contains each pixel's probability of belonging to the foreground. (2) Smoothing the cost map with an edge-preserving filter [59] aggregates the costs across neighboring pixels with similar colors. (3) Finally, pixels are assigned to the fore- or background according to the smoothed costs. While this framework enables spatially smooth pixel assignments, in its initial version it lacks temporal aggregation of costs. When being applied to a task such as video segmentation or 2D-to-3D video conversion, which both call for a labeling that is *temporally* as well as spatially smooth, the initial CVF approach can cause flickering. We achieve temporally coherent segmentation results by additionally aggregating the costs across consecutive frames. This is similar to [62], where spatio-temporal filtering was used in the domains of stereo matching and OF estimation. The video object segmentation can be further refined in a temporally coherent *alpha matting* step that accounts for transparencies at object borders. More precisely, alpha matting algorithms strive to recover an additional transparency map (denoted *alpha matte*) that describes the partial transparency of a foreground object's pixels (and the corresponding color of the foreground object). This is of particular importance in image areas where fore- and background colors are blended into each other (i.e., for *mixed pixels*). Object segmentation algorithms can not clearly assign these image areas to either the fore- or the background by a binary decision. Thus, in these image areas, alpha matting algorithms perform a "*soft segmentation*", i.e., determine a percentage that defines a partial coverage by the foreground and store this percentage in the mentioned transparency map.

Additionally to the image segmentation algorithm [125] mentioned above, related (interactive) image segmentation algorithms (e.g., [4, 19, 32, 43, 97, 139]) and (interactive) video segmentation algorithms (e.g., [4, 36, 53, 122, 160]) were already discussed in Section 3.2.2. Closely related algorithms that focus on interactive video object segmentation include [4]. In [4], the segmentation is based on the geodesic distance of a pixel to fore- and background scribbles. The geodesic distance is computed as the weight of the shortest path in a cost volume that was generated from color models. This approach requires the cost volume to be of high quality. Noise in the cost volume can accumulate over geodesic paths and lead to errors at object borders. In contrast, our approach smoothes the cost volume under guidance of the input video and yields spatially and temporally coherent segmentations, in which label changes coincide with spatio-temporal edges in the input video. Bai et al. [5] propose a framework that is able to process scenes that contain similar colors in the fore- and the background. However, it relies on accurate manual segmentation of keyframes, i.e., boundary-based image annotation. In contrast, the video object segmentation algorithm that is presented in this thesis, requires sparse and less accurate user input in the form of foreground scribbles (Section 5.2.1.1). Drawing a scribble triggers a local CVF-based optimization process that assigns pixels in all frames of a video to either the fore- or the background. As we have discussed in Section 3.2.2, various video object segmentation approaches rely on global optimization (e.g., [36, 88, 160]). Although these approaches can potentially leverage the totality of video pixels in support of computing a segmentation, processing them at the same time leads to high computational cost. Wang et al. [160] reduce the runtime of their segmentation algorithm by jointly processing groups of pixels. However, the requisite pre-processing step is itself costly (e.g., video with 720 × 480 × 175 pixels: 39 minutes [160]). In contrast, our local approach achieves runtimes of 250 fps (620 × 360 pixels).

As mentioned above, our interactive video object segmentation approach is extended to perform 2D-to-3D conversions. It considers special requirements of this application, including the extension to multiple scribble labels (i.e., multiple disparities instead of only fore- and background) and smooth disparity variations within objects that are obtained with a disparity blending approach. The main contribution of our 2D-to-3D conversion algorithm is the introduction of temporal disparity change models, which interpolate disparities over time. These models allow us not only to capture temporal disparity changes, but also to generate *perceptually coherent* disparity maps with respect to object motion. In other words, when capturing the disparity change of objects that move towards or farther away from the camera, our algorithm ensures that the disparities of these moving objects harmonize with those of nearby objects. More precisely, the temporal interpolation of the moving object's disparities is performed in accordance with occlusions between this object and nearby objects. Thus, by addressing the problem of perceptual coherence in the context of motion-caused occlusions, our 2D-to-3D conversion algorithm takes a step towards the generation of perceptually coherent disparity maps.

While the importance of enabling temporal disparity changes was already stressed by a few existing semi-automatic 2D-to-3D conversion algorithms (e.g., [56, 117]), we are not aware of any semi-automatic 2D-to-3D conversion algorithm that also considers the problem of generating perceptually coherent disparity maps. Liao et al.'s [92] automatic 2D-to-3D conversion algorithm uses automatically estimated disparities, but provides the option to combine them with user-given depth order cues for complex scenes. The algorithm ensures perceptual coherence in reference to a single, prior extracted moving foreground object, but could principally be extended to consider multiple objects similar to our algorithm. It propagates disparities with a global optimization that incorporates inequality constraints between selected pairs of neighboring pixels. These inequality constraints restrict the disparity of all pixels that are adjacent to the foreground object to be lower than the disparity of the foreground object. Liao et al. [92] further analyze object motion to detect expanding or shrinking objects and infer a disparity change depending on the object size. Contrary to [92], we aim for disparity maps that are perceptually coherent with respect to multiple moving objects. We achieve this by inferring a rough depth order [112, 154] between multiple objects which is then combined with the user-given disparities.

In Section 5.2.1, we discuss our CVF-based interactive video object segmentation algorithm. Our CVF-based 2D-to-3D video conversion algorithm is discussed in Section 5.2.2. In Section 5.3.1 and Section 5.3.2, we test our algorithms on various videos and compare them to previous algorithms. Evaluations and experiments of the video object segmentation algorithm (Section 5.3.1) show that it outperforms previous video object segmentation algorithms with similar runtime capabilities as ours [4, 125]. Experimental evaluations with our 2D-to-3D conversion algorithm (Section 5.3.2) demonstrate that our CVF-based 2D-to-3D conversion algorithm is competitive when being compared to related 2D-to-3D conversion approaches [56, 118].

5.2 Proposed Algorithm

This section first discusses our video object segmentation algorithm (Section 5.2.1) and the CVF framework for videos (with temporal cost aggregation) that uses two fixed labels, i.e., fore- and background. Then, the 2D-to-3D conversion algorithm (Section 5.2.2) is described. For this application we use a more general CVF framework that considers multiple labels (i.e., disparities).

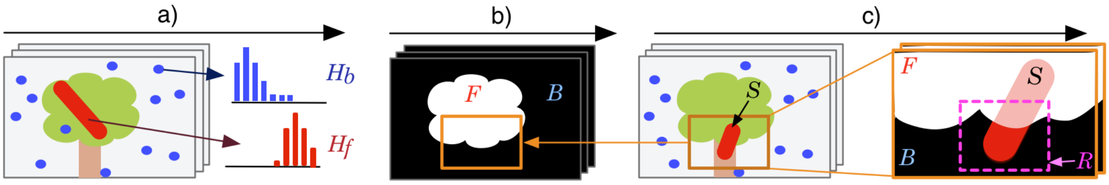


Figure 5.1: Illustration of progressive labeling (after [97]). *a)* First foreground scribble: The foreground model (H_f) is based on the marked pixels (*red scribble*). The background model (H_b) is based on randomly chosen pixels (*blue dots*). *b)* Obtained segmentation: Foreground (F) *white*, background (B) *black*. *c)* Second scribble: Scribble (S). H_f is based on newly marked pixels ($B \cap S$) and local foreground pixels ($R \cap F$) in a bounding box (R).

5.2.1 Interactive Video Object Segmentation via Cost Volume Filtering

The proposed video object segmentation and matting framework comprises three components. First, to enable a user to extract objects from a video, a scribble-based user interface is implemented (Section 5.2.1.1). Users draw on frames to indicate that the marked pixels belong to the foreground (or background). After drawing a scribble, a fast optimization (Section 5.2.1.2) based on spatio-temporal CVF is triggered. The filtered cost volume is then thresholded to get the binary segmentation, i.e., an assignment of each pixel to one of the labels $\mathcal{L} = (F, B)$, foreground F or background B. Finally, an optional matting step can account for mixed pixels at object borders. Below, these components are discussed in more detail.

5.2.1.1 Scribble-based User Interface

Aiming for little user interaction, we offer a scribble-based user interface that supports progressive editing. With this user interface, the user draws scribbles on an arbitrary frame to indicate which pixels belong to the foreground object that should be extracted from a video. This annotation initializes the segmentation process, i.e., color[1] models are built from the user-marked pixels. These color models, i.e., color histograms that sum up to one, are then used as data cost in our segmentation algorithm. This user interface follows three basic concepts: foreground selection, progressive labeling and local editing. These concepts are borrowed from [97] and are in this thesis transferred to the temporal domain. Below, they are discussed in more detail.

Foreground selection. In general, only foreground scribbles are necessary to extract a foreground object from a video.[2] This is more intuitive and reduces the amount of interaction (i.e., fewer scribbles, no need for the user to change the scribble label). Given a foreground scribble, the foreground color histogram H_f is built from the marked pixels (Figure 5.1 *a)*, *red scribble*). The background color histogram H_b is built from random samples (i.e., 1200 pixels) from the same frame (Figure 5.1 *a)*, *blue dots*). Our algorithm automatically and randomly choses these background samples from all pixels in the same frame that are not

---

[1] Other features (e.g., motion vectors [4] or texture [79]) could be additionally used.

[2] It is possible to add background scribbles and reversing the roles of fore- and background in the algorithm.

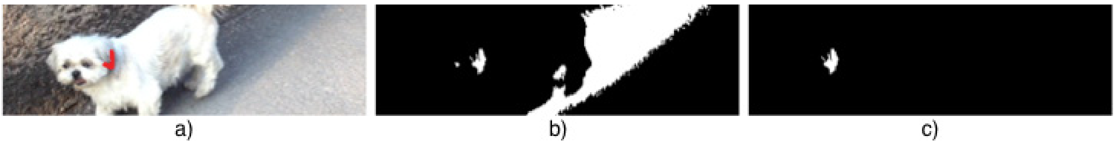


Figure 5.2: Effect of local editing. *a)* Frame and foreground scribble (*red*). *b)-c)* Binary segmentation that accepts *b)* all and *c)* only local changes. Foreground: *white*, background: *black*.

labeled as foreground pixels. Although they might not all be true background samples, the model can be further refined with subsequent user interactions.

Progressive labeling. Based on the foreground and the background histogram a binary segmentation is generated (Figure 5.1 *b)*). This segmentation, with foreground F and background B, can be expanded by adding new scribbles in any frame of the video (Figure 5.1 *c)*). When a new scribble S is drawn, the histograms are updated. H_f is re-built from the newly marked foreground pixels $B \cap S$ and *local foreground pixels*, which are available from previous interactions. Given a bounding box[3] R around $B \cap S$, these local foreground pixels are defined as $R \cap F$. To avoid that the local foreground pixels' colors are dominant in H_f, we lower their contribution in H_f based on their spatial distance to S. Next we update the background histogram H_b. This is done by removing samples from H_b that were assigned to F in the previous interaction (e.g., Figure 5.1 *a)*, background sample in green area belongs to F in *b)* and, thus, has to be replaced). These samples are replaced by randomly chosen samples from B. The progressive labeling preserves the segmentation from previous interactions by only expanding either the foreground or the background. Specifically, when adding a foreground scribble, the previous result is updated by switching only labels of background pixels.

Local editing. Based on the observation that users often want to change the segmentation only locally, we try to avoid unwanted changes far away from local user input. We implement the concept of local editing in two ways: Firstly, we build the foreground color histograms locally (as discussed above). We do this because color re-occurs quite frequently in a video and, hence, global color models would be ambiguous. In contrast, the color information is more unique in a small spatio-temporal window. Secondly, a scribble can only affect the segmentation locally. In particular, if a scribble generates a segmentation with regions that are spatio-temporally disconnected from the scribble (26-connected neighborhood), we remove the disconnected regions. This is illustrated in Figure 5.2, where the user marks the dog's head with a foreground scribble (Figure 5.2 *a)*). Due to color ambiguities, regions of the background are erroneously assigned to the foreground (Figure 5.2 *b)*). However, removing regions from the resulting binary segmentation that are not spatio-temporally connected to the scribble, leads to a better result (Figure 5.2 *c)*).

---

[3] We additionally expand the bounding box by 40 pixels in the spatial and two frames in the temporal domain.

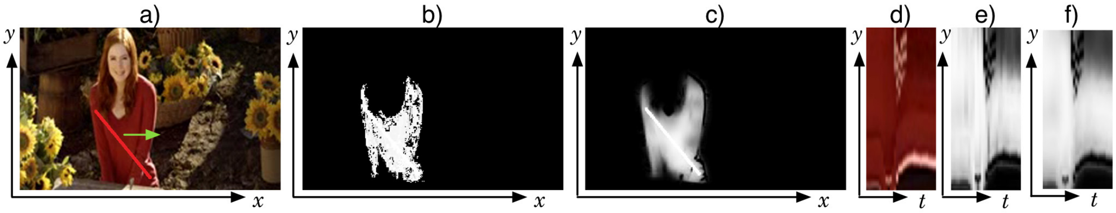


Figure 5.3: Temporally coherent cost volume filtering. *a)* Frame from guidance video *Who* with foreground scribble (*red*). *b)* Corresponding xy-slice from cost volume and *c)* from spatio-temporally filtered cost volume (high probability: *bright*, low probability: *dark*). *d)* yt-slice at position of the *green arrow* from *a)*, *e)* from per-frame filtered cost volume and *f)* from spatio-temporally filtered cost volume. Copyright of original video: BBC Worldwide Ltd.

5.2.1.2 Spatio-temporal Video Segmentation

Given the foreground color model H_f and the background color model H_b, we assign all pixels of a video to either the fore- or the background. The assignment to the labels should ideally be spatio-temporally coherent and fast. To achieve these goals, we follow three steps [125]: (1) build and (2) filter a cost volume and (3) assign labels according to the cost volume.

Cost volume generation. In the context of video object segmentation, a *spatio-temporal cost volume* P (Figure 5.3 *b)*, xy-slice for fixed t) contains the probabilities (or costs[4]) $p_i \in [0,1]$ that a pixel $i = (x,y,t)$ belongs to the foreground (or the background, i.e., $1 - p_i$). This probability is based on the comparison of the frequencies of i's bin in H_f and H_b:

$$p_i = \frac{H_f(i)}{(H_f(i) + H_b(i))}. \tag{5.1}$$

Previously marked pixels have already been assigned by the user and are accordingly set to 0 or 1 to indicate the assignment to the foreground or the background, respectively. The result of this procedure — the three-dimensional (x,y,t) cost volume P — is itself a segmentation cue (e.g., Figure 5.3 *b)*). However, naively using this cost volume for extracting the foreground ($P > 0.5$) would result in a binary segmentation that is spatially and temporally not coherent. These incoherencies are primarily caused by color ambiguities, missing colors in the color models or noise (e.g., Figure 5.3 *b)*). To account for these problems, a smoothness assumption is commonly implemented to aggregate probabilities (or data costs) from neighboring pixels that are similar in terms of color.

Spatio-temporal filtering. For smoothing we use an efficient, edge-preserving filtering technique to locally smooth the costs of similar neighbor pixels. When filtering the cost volume in a frame-to-frame manner [125], holes are smoothed in the individual frames — the segmentation is spatially more coherent — and edges in the cost volume are aligned with spatial edges in the input video (Figure 5.3 *c)*). However, filtering frames independently from each other does not prevent flickering (Figure 5.3 *d)-f)*). To obtain a spatially and

---

[4] The probability $p_i \in [0,1]$ in the cost volume corresponds to the cost $(1 - p_i) \in [0,1]$.

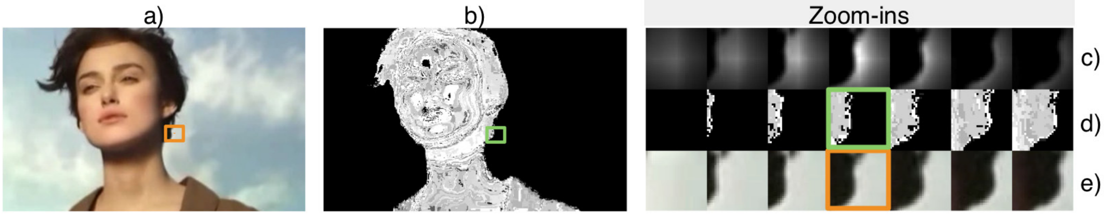


Figure 5.4: Spatio-temporal filter kernel. *a)* Frame from guidance video *Girl*. *b)* Corresponding xy-slice from P (high p_i: *bright*, low p_i: *dark*). *d)-e) Zoom-in* on the marked windows and at temporally neighboring positions. *c)* Corresponding filter weights ($r_s = 15$, $r_t = 3$, $\epsilon = 0.002$). *Bright*: high weights, *dark*: low weights. Copyright of original video: Universal Pictures.

temporally coherent result, we use the temporally extended version of GF [59] that was discussed in Section 4.2.3. Its three-dimensional filter kernel additionally smoothes temporally close-by pixels of the cost volume which are similar (color) in a guidance video (the input video I). This means cost-edges that coincide with spatio-temporal video edges are preserved. The described behavior is realized by a weighted average, i.e.,

$$p'_i = \sum_j W_{i,j}(I) p_j, \tag{5.2}$$

whose weights are given by:

$$W_{i,j} = \frac{1}{|\omega|^2} \sum_{k:(i,j)\in\omega_k} (1 + (I_i - \mu_k)^T (\Sigma_k + \epsilon U)^{-1} (I_j - \mu_k)). \tag{5.3}$$

Here, ω_k is a spatio-temporal window, with spatial radius r_s and temporal radius r_t, that is centered around a pixel k. i and j are pixels in ω_k. $|\omega|$ is the number of pixels in ω_k. I_i and I_j are color vectors (3×1). μ_k (3×1) is the mean color vector in ω_k and Σ_k (3×3) is the covariance matrix. U is the 3×3 identity matrix. p' is the filtered probability of pixel i and belongs to the filtered cost volume P'. The filter's sensitivity can be controlled by ϵ (a higher ϵ leads to smoother videos). Figure 5.4 visualizes a filter kernel for guided video filtering. It can be seen that pixels that exhibit a similar color as the central pixel have high weights. Pixels on the other side of the spatio-temporal edge have low weights.

The spatio-temporal GF can be, analogously to its spatial version [59], implemented in linear time [62]. Instead of computing the weights explicitly., i.e., using Equation (5.3), their implicit computation can be performed according to the following linear equations:

$$a_k = (\Sigma_k + \epsilon U)^{-1} \Big(\frac{1}{|\omega|} \sum_{i\in\omega_k} I_i p_i - \mu_k \overline{p_k} \Big), \tag{5.4}$$

$$b_k = \overline{p_k} + a_k^T \mu_k, \tag{5.5}$$

$$p'_i = \overline{a_i}^T I_i + \overline{b_i}. \tag{5.6}$$

Here, the filtered cost volume P' contains the filtered probabilities p'_i for each pixel i. $\overline{p_k}$ is the mean of P in ω_k. $\overline{a_i}$ (3×1) and $\overline{b_i}$ are the means of the linear coefficients a_k (3×1) and

b_k in ω_i, i.e., $\overline{a_i} = \frac{1}{|\omega|} \sum\limits_{k \in \omega_i} a_k$ and $\overline{b_i} = \frac{1}{|\omega|} \sum\limits_{k \in \omega_i} b_k$. The summations in the implicit weight computation, i.e., in Equations (5.4), (5.5) and (5.6), can be implemented as box filters. When implementing these box filters using the sliding window technique [35], the filter output can be computed in $O(N)$ with N being the number of pixels in a video. When filtering the cost volume, we additionally experimented with a temporal weighting [126] of a kernel's time slices, meaning that we give higher weights to frames near the kernel's central frame than to frames at window borders. In particular, in the summations in Equations (5.4), (5.5) and (5.6), the regular box filter was replaced by a box filter whose kernel's time slices are weighted as described above.

Fore- and background assignment. Finally, we apply a threshold (0.5) to the filtered cost volume P'. This yields an assignment of each pixel to either the fore- F or the background B:

$$\mathcal{S} = \begin{cases} F \text{ if } P' > 0.5, \\ B \text{ if } P' \leq 0.5. \end{cases} \tag{5.7}$$

To obtain a soft segmentation, which accounts for mixed pixels at object borders, an additional, temporally coherent matting step can be performed. It has been shown that the spatial GF approximates an alpha matting algorithm [59], when being applied to binary segmentations. Thus, we apply (analogously to Equations (5.2) - (5.6)) the spatio-temporal GF to the previously obtained binary video segmentation $\mathcal{S}$. More precisely, the thresholded cost volume (also denoted *binary segmentation map*), in which foreground pixels are assigned to one and background pixels are assigned to zero, is filtered with guidance of the input video I. This leads to temporally smooth matting results.

5.2.2 Semi-automatic 2D-to-3D Conversion via Cost Volume Filtering

In this section we extend our video object segmentation algorithm to perform 2D-to-3D conversions. Figure 5.5 gives an overview of the conversion algorithm's main components. It comprises six components, which transform the video object segmentation algorithm to a 2D-to-3D conversion algorithm. Compared to object segmentation, 2D-to-3D conversion typically requires (1) *multiple scribble labels* (i.e., multiple disparities). The increased number of labels might come with a larger ambiguity between their color models, which is addressed by (2) an additional *spatio-temporal closeness constraint* and (3) a *3D connectivity constraint.* We further investigate an extension of the CVF framework [125] by performing motion guided filtering to obtain (4) *improved temporal coherence* in our conversion results. Finally, we address the cardboard effect by introducing (5) smooth label changes within objects using a simple *disparity blending* approach. The focus of our presented 2D-to-3D conversion algorithm is to not only obtain temporally coherent, but also perceptually coherent disparity maps. To this end, we introduce (6) *temporal disparity change models* for the user-drawn scribbles that interpolate disparities in accordance with observed occlusions from nearby moving objects. In the following, we first discuss the basic 2D-to-3D conversion algorithm (Section 5.2.2.1), which includes components (1) to (5) (i.e., MS, STC, CON, CVF and disparity assignment in Figure 5.5). Then, we present component (6) (i.e., DC in Figure 5.5) in Section 5.2.2.2. Appendix B.1 further summarizes components (1) to (6) in the form of their program flow.

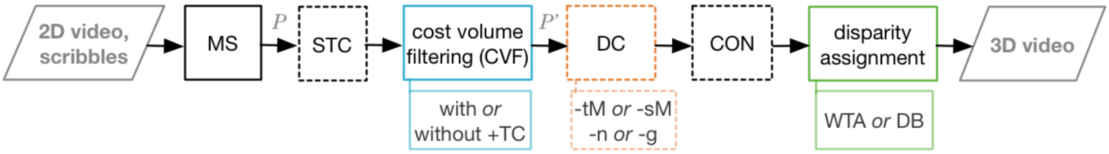


Figure 5.5: Overview of semi-automatic 2D-to-3D conversion algorithm. Generation of cost volume P from multiple scribbles (*MS*). Spatio-temporal closeness constraint (*STC*). Cost volume filtering (CVF) without or with motion guided filtering (*+TC*) to obtain P'. Temporal disparity change models (DC) that correct naive interpolations (*-n*) by guided interpolations (*-g*). They can alternatively be applied with respect to time (*-tM*) or segment size and motion (*-sM*). 3D connectivity constraint (*CON*). Final disparity assignment using a winner-takes-all (*WTA*) or a disparity blending (*DB*) scheme. *Dashed* components can be disabled.

5.2.2.1 Basic 2D-to-3D conversion algorithm

The proposed 2D-to-3D conversion algorithm uses a scribble-based user interface similarly to the user interface used in our object segmentation algorithm (Section 5.2.1.1). In case of 2D-to-3D conversion users provide initial disparities in the first (and last) frame of a video shot by drawing scribbles (Figure 5.6 *a)*). As in the user interface described in Section 4.2.1, a scribble's hue is used to assign a disparity to the pixels covered by the scribble. For the algorithm discussed in this section, a single scribble is used to indicate a single disparity. The 2D-to-3D conversion is triggered by the user, when she or he is satisfied with the annotations. Below, we describe the basic components of the 2D-to-3D conversion algorithm, i.e., all components except for the temporal disparity change models that are discussed in a separate section (Section 5.2.2.2).

Multiple scribbles (MS). User-provided scribbles $\mathcal{L} = (S_1, ..., S_L)$ indicate corresponding user-selected disparities $\mathcal{D} = (D_1, ..., D_L)$ (e.g., Figure 5.6). Since we aim to capture temporal disparity changes, users can indicate them by performing appropriate scribble annotations in the first and the last frame of a video. Based on the idea that scribbles which are contained in the same spatio-temporal segment belong to the same object, we use segmentation information and color comparisons to group and match scribbles. For that purpose, we use a motion segmentation algorithm [48] (applied without the in [48] given depth information as additional input).[5] The used motion segmentation algorithm [48] leverages long-term motion information, i.e., dense point trajectories from an OF based tracker [48, 82, 131, 145], that can also be used in later components of our 2D-to-3D conversion algorithm (e.g., +TC in Figure 5.5). Since the resulting segments are used to identify matching scribble pairs, and, hence, should not partition a single scribble into multiple segments, we initially merge all trajectories that belong to the same scribble S_l ($l \in [1, L]$) to the same segment and then proceed with the motion segmentation algorithm. The result of this process is a segmentation in which each scribble belongs to exactly one segment. The segmentation might also contain segments without any or with more than one scribble. First, we *group* scribbles that indicate the same disparity and are in the same

---

[5] It should be noted that this is different from the segmentation algorithm described in Section 5.2.1.

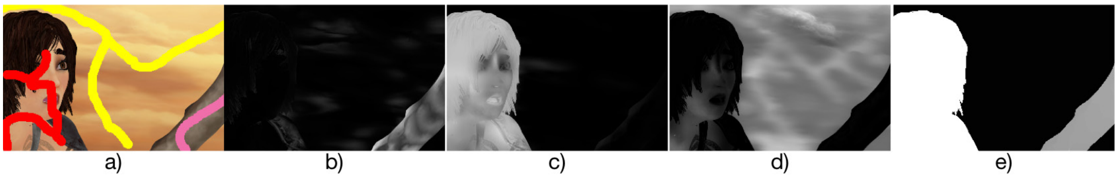


Figure 5.6: Cost volume with multiple scribble labels. *a)* First frame ($t = 1$) of input video with scribble-based annotation, i.e., three user-drawn disparity scribbles. Scribble *hues* encode disparities (i.e., the *redder* a scribble the closer it is to the camera). *b)-d)* Corresponding filtered cost volume slices $P'(.,.,1,l)$ for each scribble $\dot{S}_l$. Specifically, *b)*, *c)* and *d)* correspond to the *pink*, *red* and *yellow* scribble, respectively (*dark*: low p_i, *bright*: high p_i). *e)* Accordingly obtained disparity map (*dark* background, *bright* foreground). Original video from [27].

frame and segment. Then, we *match* scribbles that are located in the same spatio-temporal segment, but in different frames (i.e., the first and last frame). The resulting *scribble pairs* can indicate different disparities of one and the same object at different points in time. In the following, each scribble pair or set of grouped scribbles (that is not contained in a scribble pair) is referred to as $\dot{S}_l$ and the set of all $\dot{S}_l$ as $\dot{\mathcal{L}}$. Analogue to $\mathcal{D}$, each $\dot{S}_l$ has disparities that are referred to as $\dot{D}_l \in \dot{\mathcal{D}}$. All scribbles S_l that belong to a set of grouped scribble or scribble pair $\dot{S}_l$ are processed jointly.

The extension of [22] to multiple labels is straightforward. As in [62, 125] a more general CVF framework is used. Specifically, a four-dimensional cost volume $P(x, y, t, l)$ is built, in which the additional dimension, i.e., l, corresponds to the set of scribbles $\dot{\mathcal{L}} = (\dot{S}_1, ..., \dot{S}_L)$ with their corresponding disparities $\dot{\mathcal{D}} = (\dot{D}_1, ..., \dot{D}_L)$ (Figure 5.6). Analog to the case of segmentation (Section 5.2.1.2), for each scribble $\dot{S}_l$ a color model $H_{f,l}$ is built from the marked pixels in the 2D video. The color model $H_{b,l}$ for $\dot{S}_l$ captures the pixel colors from the remaining scribbles, i.e., $\dot{\mathcal{L}} \setminus \dot{S}_l$. The probability $p_{i,l}$ for each pixel $i = (x, y, t)$ and label l is computed according to Equation (5.1). The smoothness assumption (in component CVF) is incorporated by applying the temporally extended version of GF on each cost volume slice $P(.,.,t,l)$ for a fixed l analogously to Equations (5.2) - (5.6). Finally (in disparity assignment component), each pixel i can be assigned to a disparity d_i, i.e., a disparity $\dot{D}_l \in \dot{\mathcal{D}}$ from a single scribble $\dot{S}_l \in \dot{\mathcal{L}}$, by employing a *winner-takes-all* (WTA) strategy on the filtered cost volume P' and its probabilities $p'_{i,l}$:

$$d_i = \operatorname*{arg\,max}_{\dot{S}_l \in \dot{\mathcal{L}}, \dot{D}_l \in \dot{\mathcal{D}}} p'_{i,l}, \tag{5.8}$$

where at the current stage of the algorithm the disparity given in the first frame is used for scribble pairs. The current result (Figure 5.6 *e)*) equals an interactive multi-label segmentation $\mathcal{R}$ in which each segment is additionally assigned to one of the fixed disparities $\dot{D}_l$.

Spatio-temporal closeness constraint (STC). As mentioned in Section 5.2.1.1, with scribble-based annotations users typically want to indicate local assignments rather than global ones. This is also true in our application of 2D-to-3D conversion and may be complicated by color ambiguities between objects at different disparities (e.g., Figure 5.7 *a)*, bears). Thus, we constrain a scribble's influence to each pixel in the video by their spatio-temporal closeness. In some similarity to [163], our STC is implemented in form of a confidence weight $p_{close,l} \in]0,1]$ that is computed per scribble $\dot{S}_l$ and for each pixel. This confidence weight is computed from a (truncated)[6] distance transform [17] (using Chessboard distance) M_l of $\dot{S}_l$ and is applied to the cost volume before filtering its probabilities (i.e., $P(.,.,t,l) := P(.,.,t,l)p_{close,l}$). Figure 5.7 exemplarily shows the effect in a conversion result and a cost volume when applying STC. The changes in the cost volume can be observed best in the *zoom-ins* on $P'(.,.,t,1)$ and $P'(.,.,t,2)$ for the bear on the left (marked *yellow*). After applying STC (Figure 5.7 *b)*, $P'(.,.,t,1)$), the *red* scribble $\dot{S}_1$ has lower probabilities than before (Figure 5.7 *a)*, $P'(.,.,t,1)$). This means, in this example, $p_{close,l}$ reduced the probability of erroneously assigning the distant bear on the left to disparities (from $\dot{S}_1$) that were meant only for the closer bear on the right. As a result, erroneous disparity assignments in the conversion results (Figure 5.7 *a)*) that were caused by color ambiguities in the 2D image are reduced after applying STC (Figure 5.7 *b)*).

While the computation of $p_{close,l}$ is straightforward for a single frame that contains user-drawn scribbles (e.g., first frame), remaining frames require the additional step of scribble tracking. This is done by following OF vectors (trajectories from MS) at the pixels that are marked by the scribble throughout the video. While this tracking strategy might be less accurate than, e.g., object tracking, it is typically more efficient and has shown to be sufficient when being used in 2D-to-3D conversion algorithms (e.g., [117]). After scribble tracking, M_l and $p_{close,l}$ are computed based on the tracked scribbles $\dot{S}_l$ and subsequently applied to the cost volume slice $P(.,.,t,l)$ of the current frame t, i.e., $P(.,.,t,l) := P(.,.,t,l)p_{close,l}$. The result of STC is an updated P that can be smoothed and used to obtain a different $\mathcal{R}$ and disparity map than when only using MS (Figure 5.5).

Improved temporal coherence (+TC). In our object segmentation algorithm and in [62], the smoothness assumption is implemented as a spatio-temporal filtering step that smoothes the cost volume P under guidance of the input video I to obtain P'. The filtering is performed within a spatio-temporal filter window of fixed size (i.e., r_s and r_t) and, thus, uses motion between frames only implicitly. While this local approximation of the smoothness assumption and the implicit usage of motion is sufficient for most scenes, it is less robust for scenes that contain fast moving objects (e.g., Figure 5.8 *a)*, motion between frames up to 100 pixels). If the movement of an object exceeds the size of the filter window[7], the filtering step does not include pixels of the same object from different frames (e.g., Figure 5.8 *b), bottom*). Thus, the object is filtered independently for each frame, leading to reduced temporal coherence and possibly to erroneous results (e.g., Figure 5.8 *b)*, *top*).

We address this issue by incorporating motion in the filtering process, i.e., allowing the filter

---

[6] In principle, the truncation value t_{close} (in pixels) can regulate the STC's influence on the result. Instead of using a single fixed t_{close} for all scribbles, it is also possible to choose a t_{close} for each scribble individually. Although it is possible to put effort in optimizing t_{close}, for the sake of minimal user input, we use a single t_{close} that is set to the maximal possible value (i.e., frame diagonal) for all scribbles, except when we explicitly mention otherwise.

[7] It is worth to note that the choice of larger filter windows can solve the problem described above to some extent. However, since color might re-occur frequently in a video or image, larger windows also increase the probability of color ambiguities within a filter window and, hence, unwanted aggregations of probabilities.

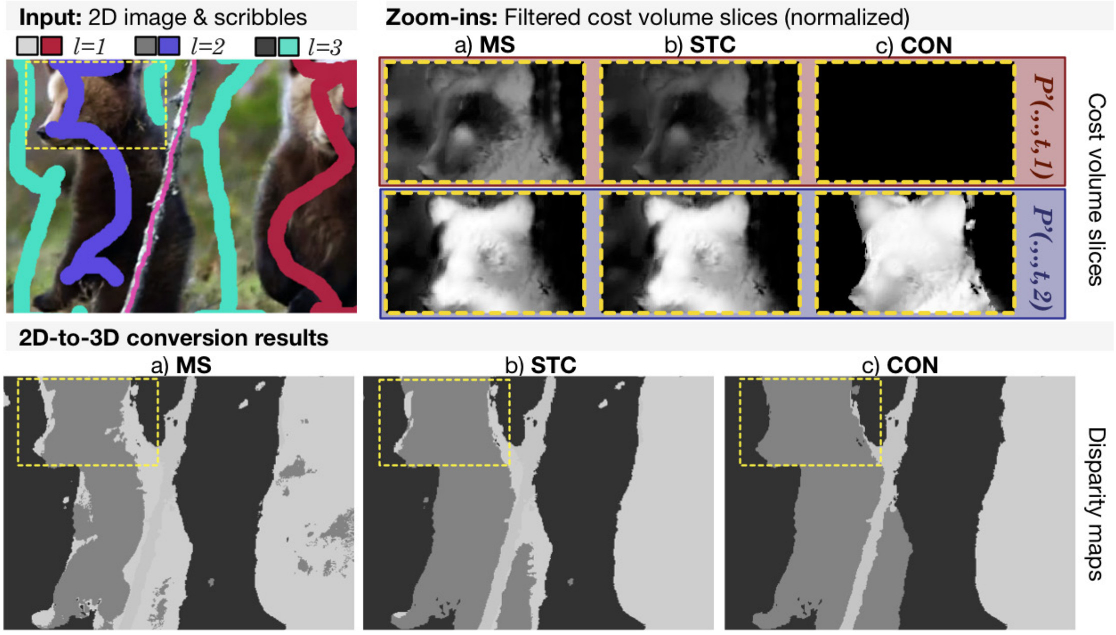

Input: 2D image & scribbles
l=1
l=2
l=3
Zoom-ins: Filtered cost volume slices (normalized)
a) MS
b) STC
c) CON
P'(.,.,t,1)
P'(.,.,t,2)
Cost volume slices
2D-to-3D conversion results
a) MS
b) STC
c) CON
Disparity maps

Figure 5.7: Effect of spatio-temporal closeness constraint and 3D connectivity constraint. Input frame t with scribbles including the *red* scribble $\dot{S}_1$ ($l = 1$) and the *violet* scribble $\dot{S}_2$ ($l = 2$). Our conversion-generated disparity maps (*bottom*): *Bright* fore- and *dark* background. *Zoom-ins* on the filtered cost volume slices $P'(.,.,t,l)$ in the *yellow* marked areas for $\dot{S}_1$ and $\dot{S}_2$. For better visualization of the effect caused by *b)* STC and *c)* the additionally applied CON over *a)* MS, P' is normalized to sum up to the value of 1 across l. In *a)* $P'(.,.,t,1)$ contains higher probabilities (is *brighter*) than in *b)* and *c)*. Copyright of original video: Discovery Communications.

window to adjust its spatial position between frames according to the motion in a video (e.g., Figure 5.8 *c)*, *bottom*). For scenes with fast motion, the maintained temporal support from neighboring frames can lead to temporally more coherent results (e.g., Figure 5.8 *c)*, *top*). We implement the smoothness assumption similar to [82], i.e., based on the same idea, but using a different filter operation. More precicely, we modify the temporal box filter that is used in the implicit weight computation of GF [59]. In our *motion guided filtering* it is applied along OF trajectories [82, 131, 145] (i.e., trajectories from MS). Figure 5.9 illustrates our filtering process. While the horizontal (Figure 5.9 *a)*) and vertical (Figure 5.9 *b)*) filtering steps of the motion guided box filter equal the respective steps of the common GF, the temporal box filter is applied differently (Figure 5.9 *c)*). Instead of computing the average of constant spatial pixel positions in different frames, the temporal average is built from pixels that belong to the same trajectory. It is worth to note that a temporal filtering step starts and ends with a certain trajectory. Motion guided filtering can be applied instead of the common GF that is used in CVF (Figure 5.5).

3D connectivity constraint (CON). We (optionally) enforce that disparity assignments and their corresponding user-provided scribbles are spatio-temporally connected in 3D to avoid

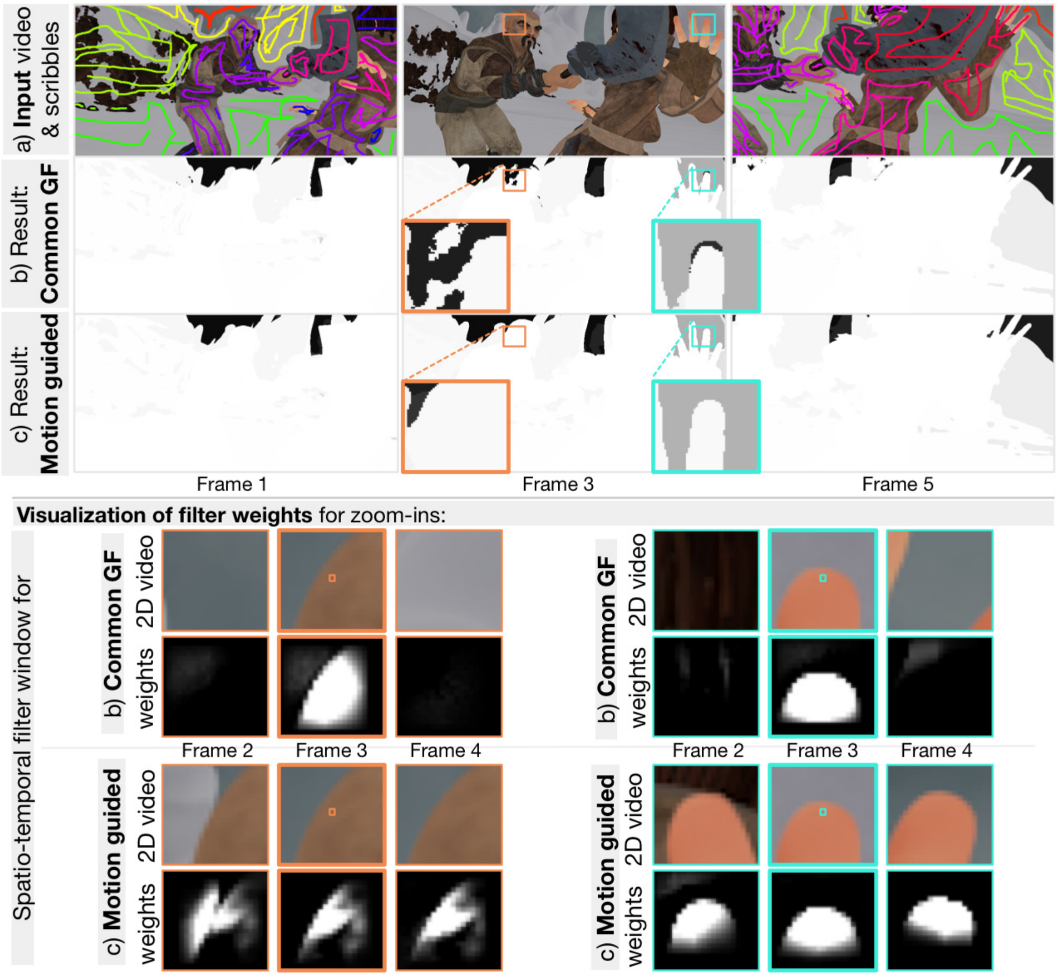


Figure 5.8: Improved temporal coherence. *Top: a)* Input video with disparity scribbles (disparity is encoded by *hue*). 2D-to-3D conversion result without *b)* and with *c)* including motion in the filtering process (front: *bright*, back: *dark*). *Zoom-ins* highlight areas where *c)* has a better result than *b)*. *Below*: Filter weights (with $r_t = 1$, $r_s = 9$, $\epsilon = 0.0016$) for marked center pixels within areas highlighted above (high weight: *bright*, low weight: *dark*). In *b)* the support of neighboring frames is lost. In *c)* the filter window is adjusted according to motion. Original video from [27].

unwanted changes in areas not connected to the user input. This connectivity constraint operates on the filtered cost volume P' and its current WTA disparity map (i.e., the pixel disparities d_i obtained after applying WTA at this stage of the algorithm). Essentially, it reduces filtered probabilities $p'_{i,l}$ in P' if they result in a WTA disparity map in which i's disparity (e.g., $\dot{D}_l$) is not connected to its corresponding scribble (e.g., $\dot{S}_l$). We consider a pixel i in a frame connected to a particular scribble (e.g., $\dot{S}_l$), if the frame contains a *connectivity path* (8-connected neighborhood) that connects all pixels, including i, with the same disparity (e.g., $d_i = \dot{D}_l$ from $\dot{S}_l$) and the scribble (e.g., $\dot{S}_l$ or tracked pixels

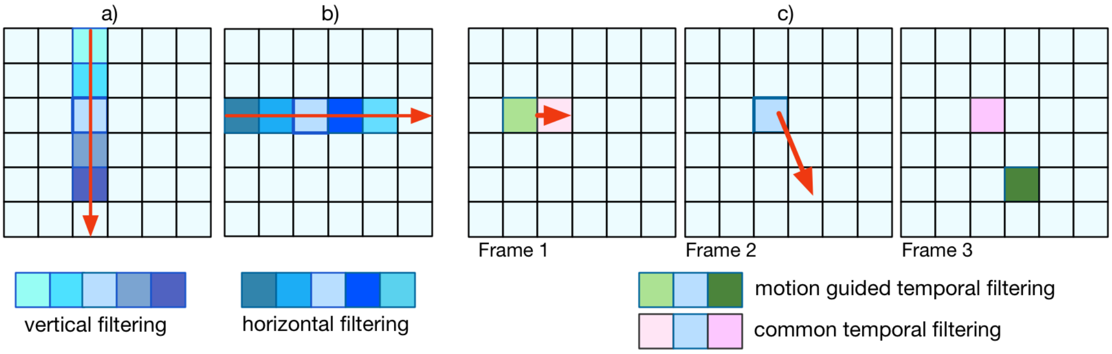


Figure 5.9: Motion guided filtering. The spatio-temporal box filter that is used in GF [59] is separable and, thus, can be implemented as three 1D filters that use the sliding window technique [35]. The *a)* vertical and *b)* horizontal 1D filtering steps of the common spatio-temporal box filter equal the respective steps of the motion guided filtering (*red arrows* in *a)* and *b)*). The *c)* temporal 1D filtering step varies. While the common version filters straight through a video, its motion guided version filters along motion vectors (*red arrows* in *c)*), i.e., trajectories [82].

from $\dot{S}_l$). Inspired by a similar formulation in [14], a connectivity path consists of pixels that either are assigned (i) to the same (e.g., $d_i = \dot{D}_l$ from $\dot{S}_l$) or (ii) to a larger disparity (e.g.,., $d_i > \dot{D}_l$). The reason for condition (ii) is to consider the common case that a connected object in the background is occluded by foreground objects. We implement CON by consecutively applying it on each scribble $\dot{S}_l$. Due to condition (ii) scribbles $\dot{S}_l$ with larger disparities $\dot{D}_l$ are processed first to exclude their invalid assignments from subsequent computations. We reduce $p'_{i,l}$ of disconnected pixels i that violate CON by an appropriate factor to reach a value close to zero. The result of CON is an updated P', which can be processed further (Figure 5.5). When recomputing the disparity map from P', pixels might be assigned to a different disparity than before.

Figure 5.7 illustrates the effect of CON by showing a conversion-generated disparity map with (Figure 5.7 *c)*) and without (Figure 5.7 *b)*) applying CON. For the *yellow* marked areas, *zoom-ins* in the cost volume slices $P'(.,.,t,l)$ for the scribbles $\dot{S}_1$ (*red*) and $\dot{S}_2$ (*violet*) are shown. Since there is no connectivity path that connects $\dot{S}_1$ in the disparity map in Figure 5.7 *b)* to the left bear, CON reduces the probability that this bear is assigned to $\dot{S}_1$'s disparity (Figure 5.7 *c)*, $P(.,.,t,1)$). As a result, erroneous assignments in the disparity maps in Figure 5.7 *b)* that are not connected to $\dot{S}_1$ are removed in Figure 5.7 *c)*.

Disparity blending (DB). Since objects might be rounded or slanted, they can exhibit multiple disparities that blend into each other. We support this case by (optionally) substituting the WTA scheme with a simple blending approach (i.e., disparity blending approach) similar to [170]. Specifically, a final disparity d_i for a pixel i is assigned to a weighted average of the scribble disparities $\dot{D}_l$ derived from the filtered probabilities $p'_{i,l}$ according to:

$$d_i = \frac{\sum_l p'_{i,l} \dot{D}_l}{\sum_l p'_{i,l}}. \tag{5.9}$$

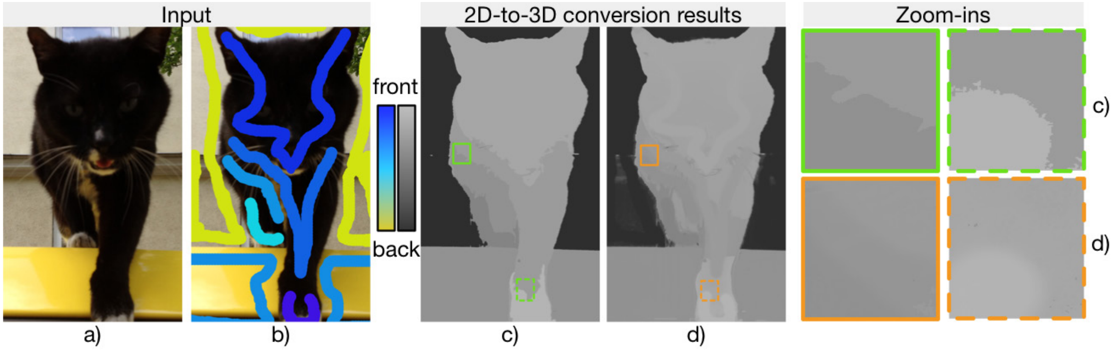


Figure 5.10: Smooth disparity changes. *a)* Original image *b)* with disparity scribbles (the *bluer* a scribble the larger its disparity). 2D-to-3D conversion result with *c)* WTA and *d)* with DB ($n = 2$). Foreground: *bright*, background: *dark*. DB causes smoother disparity transitions than WTA and maintains hard disparity edges near object borders that exhibit different colors.

Here, multiple disparities $\dot{D}_l$ with high $p'_{i,l}$ can influence the final disparity d_i of a pixel i, while disparities $\dot{D}_l$ with low $p'_{i,l}$ hardly contribute to d_i. We further restrict this averaging to the disparities $\dot{D}_l$ with the n highest $p_{i,l}$ for each pixel i. An example of a disparity map that was computed using DB ($n = 2$), compared to a disparity map that was generated with the WTA scheme, is shown in Figure 5.10 *d)* and *c)*, respectively. It can be seen that the blending approach generates a smoother disparity map than the WTA scheme.

5.2.2.2 Temporal disparity change models

To capture motion towards or away from the camera in our 2D-to-3D conversion results, we introduce temporal disparity change models for the user-given scribble pairs. Specifically, for each scribble pair a *disparity change model* (DC) has to specify an interpolation between its disparity given in the first and its disparity given in the last frame of the video (Figure 5.5). For a scribble pair $\dot{S}_l = \{S_l, S_k\}$ with disparities $\dot{D}_l = \{D_l, D_k\}$, a DC specifies the interpolation between D_l and D_k. Naively, this interpolation could be performed linearly according to the closeness of the regarded frame to the first and the last frame. While this solution might work in some cases, it has a major drawback – the resulting disparity maps might be perceptually incoherent. Figure 5.11 shows a failure case for such a *naive interpolation*. In this example, a small dragon moves from behind the large dragon's wing to the front, but is assigned to a disparity that still places it behind the wing. We address this issue by performing the temporal disparity interpolation in accordance with observed occlusions that were caused by nearby objects. The underlying basic idea is that, if an object A (e.g., small dragon in Figure 5.11) moves in front of another object B (e.g., large dragon's wing in Figure 5.11) in frame t – i.e., if A occludes B in frame t – we can conclude that in t A has a larger disparity than B. Thus, when interpolating A's disparity over time it should exceed B's disparity in frame t, i.e., should be restricted by B's disparity (as a lower boundary) in frame t. We implement this idea by, first, determining *a rough depth order* in each frame according to the motion observed in the video. Then, this rough depth

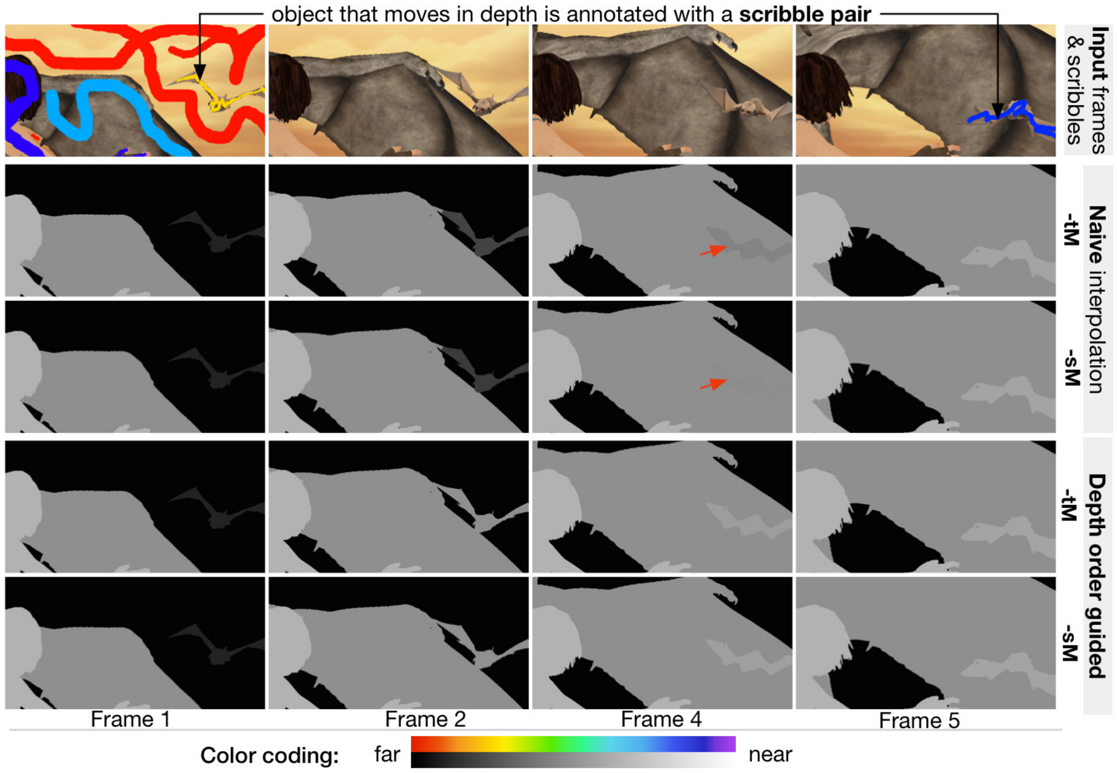
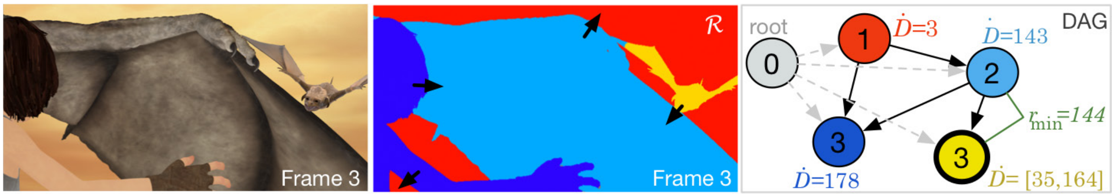


Figure 5.11: Temporal disparity change models. *First row*: Input video with color-coded disparity scribbles. The small dragon is annotated with a yellow scribble in the first and a blue scribble in the last frame to indicate a temporal disparity change. *Second to fifth row*: 2D-to-3D conversion results using different models, i.e., naive linear disparity interpolations (*second and third row*) and our depth order guided interpolations (*fourth and fifth row*). The interpolations can be applied with respect to time (*-tM*) and with respect to segment size and motion (*-sM*). The naive interpolations are perceptually incoherent (i.e., *red arrow*). Original video from [27].

order is used to define *disparity restrictions*, which are, finally, incorporated in the *depth order guided disparity interpolation*. These steps rely on motion estimated from the input video, the filtered cost volume, i.e., P', and intermediate results that can be derived from it (Figure 5.5).

Rough depth order. First, a rough depth order between segments is determined for each frame. We use segments that are derived by interpreting the current WTA result as a multi-label segmentation, i.e., $\mathcal{R}$. A segment $R_l \in \mathcal{R}$ in frame t belongs to exactly one scribble (pair) $\dot{S}_l$. We collect depth order cues between pairs of segments in each frame (Figure 5.12 *b)*). Similar to [112] we detect (dis)occlusions in videos by checking the consistency of the fore- and backward OF in conjunction with corresponding segment affiliations. More precisely, according to [112] a (dis)occlusion can be detected by identifying two forward (or two backward) motion vectors from $t-1$ (or $t+1$) that converge on the same pixel in t. By checking the segment affiliations of the three involved pixels (i.e., two pixels from the



a) b) c)

Figure 5.12: Disparity restriction example. *a)* Intermediate input frame from example in Figure 5.11. *b)* Corresponding multi-label segmentation $\mathcal{R}$: Segment colors correspond to scribble colors in the first frame in Figure 5.11. *Black arrows* visualize depth order cues, i.e., segment at the arrow's shaft occludes segment at its pointy end. *c)* DAG: Node colors correspond to segment colors. Directed edges (*black*) constitute an occlusion relation, i.e., parent node is occluded by child node. The *numbers* in the nodes are their depth level, i.e., $\lambda(R_l, t)$. The artificial *root* node (*gray*) and its *gray* edges were added to determine $\lambda(R_l, t)$. The minimum restriction $r_{\min} = 144$ of the *yellow* node is based on the *light blue* node. Original video from [27].

neighboring frame and one pixel in the regarded frame t) the (dis)occluding segment can be identified and recorded. For each frame, these depth cues are stored in a *directed acyclic graph* (DAG) [112, 154]. The nodes of this DAG correspond to segments $R_l \in \mathcal{R}$ (e.g., Figure 5.12 *b)* and *c)*). Directed edges $e_{l,k}$ between nodes, e.g., R_l and R_k, record an occlusion relation between them. In this connection, *child nodes* are closer to the camera than their *parent nodes* (e.g., dragon with *yellow* and wing with *light blue* node in Figure 5.12 *b)* and *c)*). When adding a depth ordering cue to the DAG, a *depth-first-search* [148] is triggered to identify cycles. In case of a cycle or conflicting pairwise occlusion cues for a segment, only the more frequent depth order cue is used [154].[8]

The global relative depth order for a frame t is captured by the hierarchy level (or *depth level*) of each node in the DAG with respect to an artificial root node (Figure 5.12 *c)*, *gray node*). This root node is connected to all other nodes by a directed edge that starts at the root node. The maximum number of nodes that have to be passed to reach R_l from the root node [9] is the depth level $\lambda(R_l, t)$ of R_l (e.g., Figure 5.12 *c)*, *numbers* in nodes). Hence, the larger the depth level of a segment (node), the closer it is to the camera.

Disparity restrictions. Given the DAG for a frame t, its depth levels λ for segments R_l and the user-provided disparity annotations $\dot{D}_l$, we define disparity restrictions that guide the temporal disparity interpolation. As stated above, we assume that only scribble pairs indicate a change in disparity over time. Scribbles without annotations in both the first and the last frame are associated with their fixed user-assigned disparities within the entire video. These fixed disparities can be exploited when deriving a scribble pair's *disparity restrictions*, i.e., a range $[r_{\min}(R_l, t), r_{\max}(R_l, t)]$ that defines a minimal and maximal allowed disparity that can be taken on by a specific scribble pair $\dot{S}_l$ (and corresponding segment R_l) in frame t. The restrictions for the minimal $r_{\min}(R_l, t)$ and maximal $r_{\max}(R_l, t)$ disparity are determined in ascending order of λ, i.e., restrictions of parent nodes (background)

---

[8] In principle, for segments not corresponding to scribble pairs, their given disparities could also resolve conflicts.

are computed before restrictions of their child nodes (foreground), as following:

$$r_{\max}(R_l,t) = \min_{R_k \in c(R_l,t)} \begin{cases} \dot{D}_k - |\lambda(R_l,t) - \lambda(R_k,t)| & \text{if } \dot{D}_k \text{ fixed,} \\ \dot{D}_{c(R_k,t)} - |\lambda(R_l,t) - \lambda(c(R_k,t))|) & \text{if } \dot{D}_k \text{ not fixed,} \\ Z & \text{if } \dot{D}_k \text{ not found.} \end{cases} \tag{5.10}$$

$$r_{\min}(R_l,t) = \max_{R_k \in p(R_l,t)} \begin{cases} \dot{D}_k + |\lambda(R_l,t) - \lambda(R_k,t)| & \text{if } \dot{D}_k \text{ fixed,} \\ r_{\min}(R_k,t) + 1 & \text{if } \dot{D}_k \text{ not fixed,} \\ 0 & \text{if } \dot{D}_k \text{ not found.} \end{cases} \tag{5.11}$$

Here, $p(R_l,t)$ are parent nodes and $c(R_l,t)$ are child nodes of a given node R_l in frame t. R_l and R_k are segments in t. Let us first consider the first cases in Equation (5.10) and Equation (5.11). Segment R_k is invoked by scribble $\dot{S}_k$ with a user-assigned fixed disparity $\dot{D}_k$. $\dot{S}_l$ is a scribble pair. In Equation (5.11), the minimum restriction for R_l is determined from the maximum disparity of all its parent nodes i.e., $\max_{R_k \in p(R_l,t)}(\dot{D}_k)$ (e.g., Figure 5.12 *c)*, *yellow* node derives disparity $r_{\min}$ from *light blue* node). Since R_l should have a larger disparity than R_k (R_l occludes R_k), $\dot{D}_k$ is increased by the difference in depth levels of the current and the found node, i.e., $|\lambda(R_l,t) - \lambda(R_k,t)|$. In Equation (5.10), the maximum restriction $r_{\max}(R_l,t)$ for R_l in t is determined analogously. When considering scenes that contain scribble pairs that occlude each other (e.g., due to a camera zoom), the focus in Equation (5.10) and Equation (5.11) is on their second and third cases. In these cases, segments R_l and R_k correspond to scribble pairs $\dot{S}_l$ and $\dot{S}_k$ and occlude each other. $r_{\max}(R_l,t)$ and $r_{\min}(R_l,t)$ are at first calculated momentarily and updated as soon as the disparity assignment for R_l is fixed (after interpolation). In Equation (5.10)'s second case, this is done by further traversing the child nodes of the occluding segment R_k until a child node $c(R_k,t)$ that corresponds to a scribble with a fixed disparity $\dot{D}_{c(R_k,t)}$ is found. Since $\dot{D}_l$ should be larger than $\dot{D}_k$ and $\dot{D}_{c(R_k,t)}$, $|\lambda(R_l,t) - \lambda(c(R_k,t))|$ is subtracted from $\dot{D}_{c(R_k,t)}$. If no child node with a fixed disparity was found, a fixed maximal disparity Z is used (in Equation (5.10)'s third case). In Equation (5.11), $r_{\min}(R_l,t)$ is determined from the minimum restrictions of R_l's parent nodes or uses the fixed minimal disparity 0 if no parent node was found. Hence, with the procedure described above, videos solely annotated with scribble pairs can also be processed.

Depth order guided disparity interpolation. The final disparity of each scribble pair can be determined by interpolating their user-given disparities over time while taking into account the restrictions that were defined above. To allow the update of momentarily set disparity restrictions, the interpolation is performed in ascending order of λ. We perform this temporal interpolation according to two different disparity change models, the *time-based disparity change model* (DC -tM) and the *segment-based disparity change model* (DC -sM). DC -tM interpolates user-assigned disparities (e.g., $\dot{D}_l = \{D_l, D_k\}$) of a scribble pair (e.g., $\dot{S}_l = \{S_l, S_k\}$) between the disparity at the first (e.g., D_l) and at the last frame (e.g., D_k). The final disparity $\dot{D}_{l,t}$ at frame t is initially given by a linear interpolation with respect to time, in which T is the total amount of frames in the video:

$$\dot{D}_{l,t} = D_l + \frac{D_k - D_l}{T} t. \tag{5.12}$$

DC -sM interpolates disparities with respect to a segment R_l's size $s(R_l, t)$ and its motion in t:

$$\dot{D}_{l,t} = D_l + \frac{D_k - D_l}{s(R_l, T) - s(R_l, 1)}(s(R_l, t) - s(R_l, 1)). \tag{5.13}$$

In particular, DC -sM considers irregularly moving objects by changing the disparity in conjunction with segment sizes (i.e., height in pixels) in different frames. A disparity is only updated (compared to the previous frame) if changes in the segment size and vertical movement [56] are observed. The latter idea comes from the observation that a change in disparity occurs more often in conjunction with vertical than with sole horizontal motion [56]. Note that DC -sM supports the more complex case, in which an object not only stops, but changes its movement in depth during the scene (e.g., a car moves towards the camera, stops and reverses, before it approaches the camera again).

DC -tM and DC -sM are combined with the previously obtained disparity restrictions. For that purpose, we perform a step-by-step interpolation for the currently considered DC. To begin, the naive interpolation is performed between two fixed data points D_l and D_k, i.e., the user-given disparities of $\dot{D}_l = \{D_l, D_k\}$ in the first and the last frame, and might produce percepetual incoherencies (e.g., Figure 5.11, *naive interpolations*). Then, the interpolation is further guided according to the disparity restrictions. We perform a recursive disparity verification and adjustment step for each frame in order to remove the mentioned incoherencies (e.g., Figure 5.11, *guided interpolations*). More precisely, we compare the current disparity of each scribble pair (e.g., $\dot{S}_l$) with the upper and lower bounds that are provided by its disparity restrictions $[r_{\min}(R_l, t), r_{\max}(R_l, t)]$. If the current disparity violates these restrictions, it is adjusted to the closest disparity within the allowed disparity range. This adjustment adds another disparity data point to the interpolation and triggers an according update (i.e., recomputation) of the disparities of preceding and following frames. These recomputed disparities are again recursively verified and, if necessary, adjusted until only valid disparities are used. Thus, after this procedure the disparity change model of each scribble pair specifies a disparity that is consistent with the previously extracted disparity restrictions in each frame.

Figure 5.13 illustrates this procedure for the scene shown in Figure 5.11 for DC -tM (Figure 5.13 *a)* and *b)*) and DC -sM (Figure 5.13 *c)* and *d)*). In this example, the disparity of the small dragon, who's scribble pair contains a *yellow* scribble in the first and a *blue* scribble in the last frame in Figure 5.11, is interpolated over time. The colors of the restrictions (i.e., *blue rectangles* and *red line* in Figure 5.13) correspond to the colors of the scribbles in Figure 5.11. The *yellow* line in Figure 5.13 illustrates the performed step-by-step interpolation, before (Figure 5.13 *a)* and *c)*) and after verification and adjustment (Figure 5.13 *b)* and *d)*). As shown (Figure 5.11, *depth order guided interpolation* and Figure 5.13 *b)* and *d)*), the final disparity maps are perceptually coherent. Contrary to the results of a naive interpolation (Figure 5.11, *naive interpolation* and Figure 5.13 *a)* and *c)*), the small dragon is assigned to a plausible disparity that conforms with the depth order in the scene.

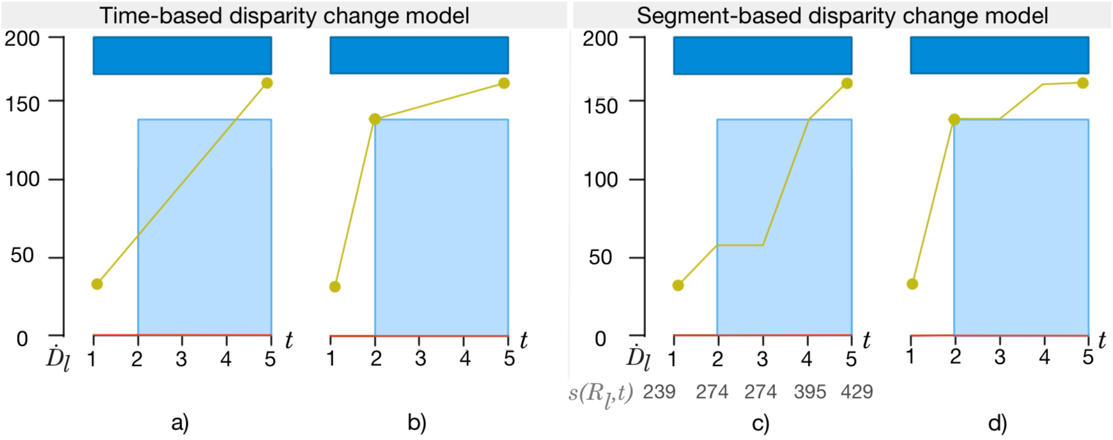


Figure 5.13: Step-by-step interpolation in temporal disparity change models: *a)-b)* time-based (-tM) and *c)-d)* segment-based (-sM) disparity change model (DC). DC -sM considers segment sizes $s(R_l, t)$. The disparities at frames t in *a)* and *c)* are based on a naive interpolation of $\dot{D}_l$ without incorporating disparity restrictions, i.e., *a)* on Equation (5.12) and *c)* on Equation (5.13). Since in this toy example, the segment size in the second and third frame are the same, so is the disparity in *c)*. These disparities (*yellow line*) conflict with the disparity restrictions (*blue rectangles*, *red line*). Thus, they are adjusted in *b)* and *d)*: A new data point (*yellow point*) is added and the interpolation between its neighboring data points is recomputed and verified.

5.3 Experimental Results and Evaluation

In this section we evaluate our interactive video object segmentation algorithm (Section 5.2.1) and our 2D-to-3D conversion algorithm (Section 5.2.2). We show various results that were obtained by our algorithms. The evaluation of the video object segmentation algorithm (Section 5.3.1) focuses on the improved temporal coherence that was obtained by extending the image segmentation via CVF [125] to the temporal domain. We further perform visual comparisons to a related video object segmentation algorithm [4]. For the 2D-to-3D conversion algorithm (Section 5.3.2), we systematically evaluate the impact of each component. Furthermore, we compare our results with those from related conversion algorithms including [56] and [118].

5.3.1 Video Object Segmentation Results

Our interactive video object segmentation algorithm was implemented and tested on a 2.4 GHz Intel Core 2 Quad PC with a GeForce GTX40 graphics card. We used CUDA for the GPU implementation. In this configuration, our implementation requires four milliseconds to segment a frame with a resolution of 620×360 pixels. Throughout our tests, we use the following constant filter parameters to obtain a binary segmentation: $r_s = 11$, $r_t = 2$, $\epsilon = 0.002$. The color histograms use 32 bins per channel. Figure 5.14 shows segmentation and alpha matting (soft segmentation) results that are generated by our algorithm. It was applied on various video shots, including broadcast videos (Figure 5.14 *a)*, *d)-f)*), a computer-generated video (Figure 5.14 *b)*) and a shot that was captured with the camera of a mobile phone (Figure 5.14 *c)*). The videos

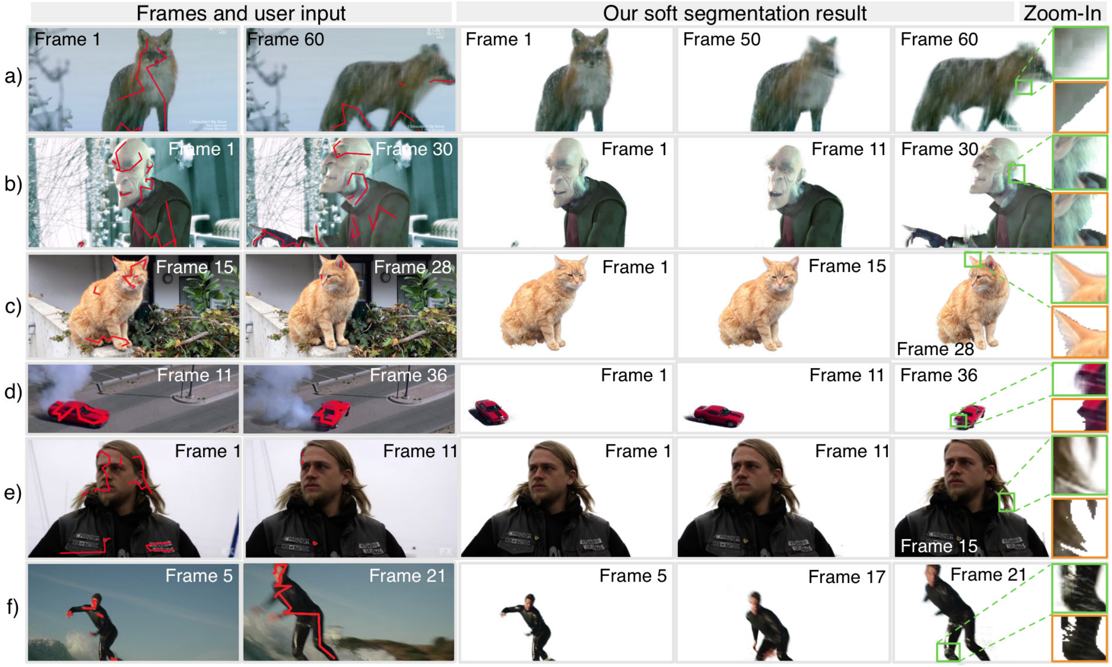


Figure 5.14: Our video object segmentations. *Left:* Frames and user input: Foreground scribbles (*red*). *Right:* Corresponding segmentations with alpha matting. *Zoom-ins*: Results before (*bottom*) and after (*top*) alpha matting. Copyright of original videos: *a)* Discovery Communications, *b)* Blender Foundation, *d)* Sony Pictures, *e)* Twentieth Century Fox, *f)* ABC Studios.

consist of 60 (Figure 5.14 *a)*), 31 (Figure 5.14 *b)*), 30 (Figure 5.14 *c)*), 41 (Figure 5.14 *d)*), 15 (Figure 5.14 *e)*) and 21 (Figure 5.14 *f)*) frames, respectively. In each video only two frames had to be annotated (Figure 5.14, *first two columns*) to obtain the results shown (Figure 5.14, *last three columns*). On average, eight scribbles were placed in a video of 33 frames. Note that we used only foreground scribbles. We obtained spatially and temporally coherent segmentations, in which temporal and spatial edges coincide with color edges in the corresponding input video. As can be seen in the *zoom-ins* (Figure 5.14, *right*), the additional matting step further refines our segmentations by capturing fine details and transparencies. The test videos include examples with similar colors in the fore- and background (e.g., Figure 5.14 *b)*), which would lead to errors, when only using the data cost (i.e., P) without smoothing (Figure 5.15). We observed limitations in scenes with weak borders between fore- and background regions. Figure 5.16 shows an example of such a failure case, where the color of a ground patch matches the color of the bison's horn, which causes it to be erroneously assigned to the foreground.

Comparison to per-frame filtering [125]. We compare our approach, which aggregates probabilities spatially as well as across frames, with the per-frame CVF approach of [125] that uses a two dimensional (spatial) kernel. For both approaches we use the same user input and color models, which result in the same cost volume. Except for the additional temporal extension, we use the same parameters (i.e., $r_s = 11$, $\epsilon = 0.002$) as for the filter

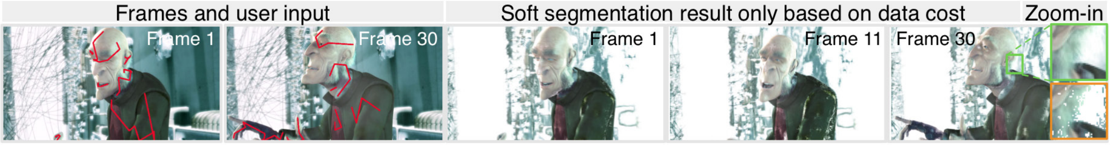


Figure 5.15: Video object segmentation based only on data cost (no smoothing). *Left*: Frames with *red* foreground scribbles. *Right*: Soft segmentation result. The *zoom-ins* show the results before (*bottom*) and after (*top*) alpha matting. Copyright of original video: Blender Foundation.

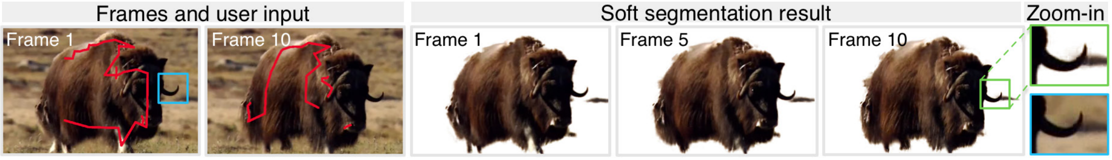


Figure 5.16: Failure case of proposed video object segmentation algorithm. *Left*: Two frames with *red* foreground scribbles. *Right*: Soft segmentation result. *Zoom-ins*: Result (*top*), input (*bottom*). Copyright of original video: Discovery Communications.

in [125]. We observe that the filtered cost volume and, more importantly, the segmentation results are temporally more coherent when using our proposed approach. Figure 5.17 gives an example for this case. Contrary to the per-frame approach [125] (Figure 5.17 *b)*), our approach achieves a largely flicker-free segmentation (Figure 5.17 *c)*). We quantitatively compare our binary segmentation maps (e.g., Figure 5.17 *b)* and *c)*) using a measure for temporal coherence [83], which we refer to as *Flickering Error* (FE):

$$FE_i(t) = \frac{|a_i - a_j|}{|I_i - I_j| + 1}. \tag{5.14}$$

This measure detects label changes ($a \in \{0, 1\}$, 0 background, 1 foreground) of temporally neighboring pixels i and j. The error for such a label change is given by the pixels' color similarity I_i, I_j. Accordingly, the error for label changes at similar pixels is higher than for label changes that go along with color changes. It can be seen (Table 5.1, Figure 5.17) that filtering the cost volume with our spatio-temporal approach significantly reduces the error (e.g., for *Bear* reduction of 56 percent from *per-frame* [125] to *spatio-temporal*), which indicates temporally more coherent results. An additional temporal weighting (Table 5.1, *spatio-temporal + weighting*) can lead to improvements over the approach without weighting. In Table 5.2, we further compare the soft segmentation results that were generated from the same binary segmentations, but by using the different filtering approaches discussed above (*per-frame* [125], *spatio-temporal*, *spatio-temporal + weighting*) to approximate the alpha matting process. The same filter parameters (i.e., $r_s = 5$, $r_t = 1$, $\epsilon = 0.00001$ for alpha matting[9]) are used for all filtering approaches, except for the nonexistent temporal radius in case of [125]. Table 5.2 shows that our spatio-temporal filtering approach increases the

---

[9] In practice, different scenes typically require different filter parameters. The parameters also depend on the size of the image areas in which mixed pixels at object borders are expected. For instance a close-up of a person with long hair (typical cause of mixed pixels) usually requires larger filter windows than a wide-shot of the same person.

Table 5.1: Binary segmentation – comparison of per-frame [125] and spatio-temporal approach. Averaged Flickering Error [83] for binary segmentations of eight videos. From *top* to *bottom*: The segmentations were computed by thresholding the *not filtered* P, the *per-frame* and the *spatio-temporally filtered* P without and *with weighting*. The errors are multiplied by 1000.

Averaged Flickering Error	*Bear*	*Board*	*Temple*	*Who*	*Girl*	*Surf*	*Arms*	*Fish*
not filtered	3.23	5.67	0.12	1.18	3.64	3.23	4.43	4.23
per-frame [125]	1.03	0.74	0.31	0.19	0.93	0.25	1.35	1.14
spatio-temporal	0.45	0.32	0.20	0.10	0.64	0.17	0.52	0.66
spatio-temporal + weighting	0.43	0.31	0.20	0.09	0.61	0.17	0.47	0.62

Table 5.2: Soft segmentation – comparison of per-frame [125] and spatio-temporal alpha matting. Averaged Flickering Error [83] for the alpha mattes of eight videos. From *top* to *bottom*: Error for the input, i.e., a binary segmentation (*not filtered*). The alpha mattes were computed by *per-frame* filtered binary segmentation maps [125] and the *spatio-temporally filtered* binary segmentation maps without and *with weighting*. The listed errors are multiplied by 1000.

Averaged Flickering Error	*Bear*	*Board*	*Temple*	*Who*	*Girl*	*Surf*	*Arms*	*Fish*
not filtered	1.50	0.64	0.20	0.10	0.64	0.17	0.52	0.66
per-frame [125]	1.81	0.73	0.57	0.27	0.72	0.28	0.61	0.87
spatio-temporal	1.61	0.74	0.42	0.24	0.68	0.26	0.67	0.82
spatio-temporal + weighting	1.60	0.72	0.41	0.24	0.67	0.26	0.66	0.81

temporal coherence for the alpha matting step. We also observe a failure case, i.e., for *Arms*, which is shown in Figure 5.18. In this particular example, large movements (approximately 60 pixels) and color similarities between successive frames cause subtle temporal alpha matting artifacts (Figure 5.18, *zoom-ins*) when applying our spatio-temporal approach.

Comparison to geodesic segmentation [4]. Due to the lack of segmentation ground truth, we visually compare the results of our binary segmentation to those of a geodesic segmentation framework (Figure 5.19). We use our re-implementation of [4] to generate the shown binary segmentations. Our algorithm outperforms the geodesic segmentation algorithm [4] using the same foreground scribble as input. While our background model is based on randomly chosen samples, in [4] users initialize the background model with an additional background scribble. As can be seen in Figure 5.19, our segmentation results adapt better to the scenes than those of [4]. In contrast to the results obtained by the re-implemented geodesic algorithm (Figure 5.19 *b)*), in our results (Figure 5.19 *a)*) label changes coincide with spatio-temporal edges in the input video. This is, in fact, a result of guided filtering, which smoothes probabilities in homogenous regions of the input video.

5.3.2 2D-to-3D Conversion Results

Our CVF-based 2D-to-3D conversion algorithm was tested on a 2.4. GHz Intel Core 2 Quad PC. Contrary to our video object segmentation algorithm, for the 2D-to-3D conversion algorithm we

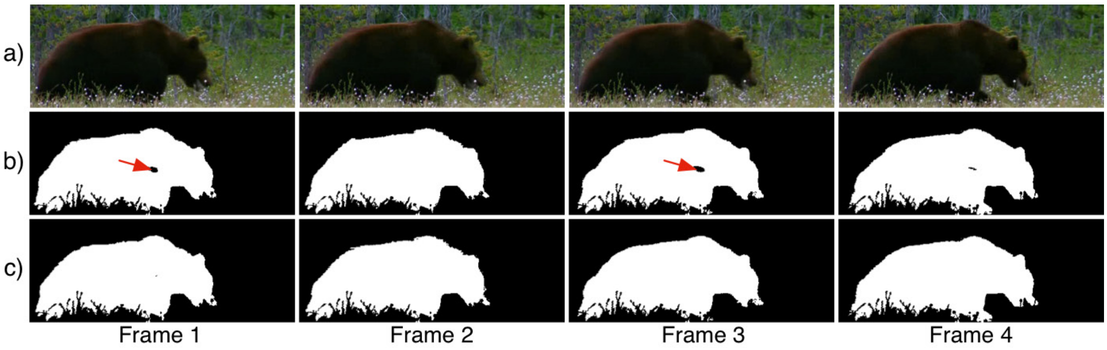


Figure 5.17: Visual comparison of proposed video object segmentation algorithm to [125]. *a)* Frames from *Bear*. *b)* Binary segmentation maps from per-frame cost volume filtering [125] and *c)* our spatio-temporal cost volume filtering. *Red arrows*: Region that flickers. Foreground: *white*, background: *black*. Copyright of original video: Discovery Communications.

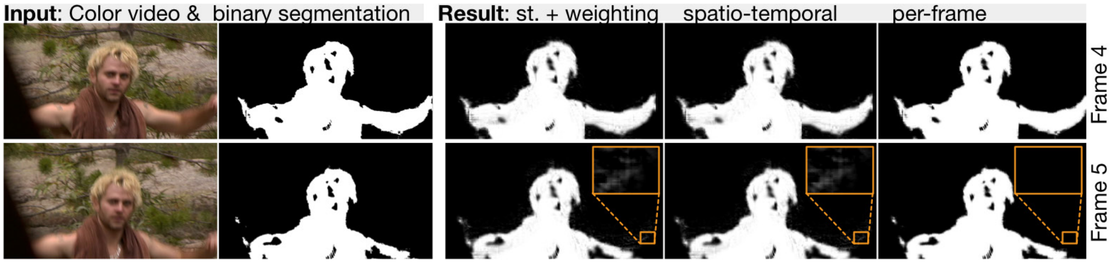

Input: Color video & binary segmentation
Result: st. + weighting
spatio-temporal
per-frame
Frame 4
Frame 5

Figure 5.18: Failure case of spatio-temporal alpha matting. *Left*: Input frames and input binary segmentation map. *Right*: Alpha mattes computed by *per-frame* [125] and spatio-temporal filtering of the segmentation maps with (*st. + weighting*) and without weighting (*spatio-temporal*). *Zoom-ins*: Temporal alpha matting artifacts, which are not visible when filtering only per-frame. Copyright of original video: Buena Vista Home Entertainment and Touchstone Television.

use a regular C++ implementation that runs on the CPU. In this configuration our algorithm requires 100 seconds to generate a disparity video from a monoscopic video that consists of ten frames (resolution of 1024×436 pixels) and was annotated with five scribbles. This runtime was measured excluding the computation of OF, but including all previously discussed components of our conversion algorithm (i.e., MS, STC, CON, DC -tM and +TC). The most computationally expensive components are CON and +TC, which consume 40 and 48 seconds, respectively. The usage of common GF (without motion) instead of our motion guided filtering in +TC reduces the total runtime to 67 seconds. It is worth noting that a CUDA implementation of the conversion algorithm would have similar runtime properties as the CUDA implementation of our object segmentation algorithm (Section 5.2.1.2). Specifically, only by using the CUDA implementation of GF instead of its currently used C++ implementation (using common GF), we would approximately gain a speed-up by a factor of 1.3[10] Other components of our algorithm,

---

[10] The speed-up factor was determined by comparing the total runtime of our algorithm with its currently used C++ implementation of the common GF and the total runtime of our algorithm with our CUDA implementation of

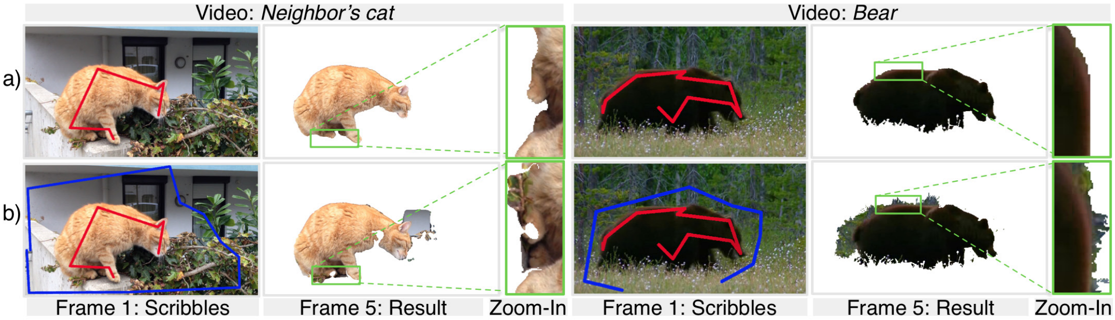


Figure 5.19: Visual comparison to [4]. For each video: Frame and user input (*left*). Binary segmentation of a frame (*right*) for *a)* the proposed algorithm and *b)* for [4]. The user input differs only in the additional background scribble (*blue*) for [4]. The same foreground scribble (*red*) is used. Copyright of original video *Bear*: Discovery Communications.

e.g., the WTA scheme [62, 125], are as well suitable for parallel execution on the GPU.

Our CVF-based 2D-to-3D conversion algorithm was tested with the following filter parameters to obtain disparity maps for an annotated monoscopic video: $r_s = 11$, $r_t = 2$, $\epsilon = 0.0016$. The color histograms use 32 bins per color channel (i.e., RGB). STC uses a global t_{close} which is set to the maximal possible distance within a frame for all scribbles. All shown results that use DB are obtained with $n = 2$. Figure 5.20 shows disparity maps that were generated by our CVF-based 2D-to-3D conversion algorithm that was applied on four monoscopic videos, including shots from broadcast videos (Figure 5.20 *a)*, *b)* and *d)*) and a shot that was captured using the camera of a mobile phone (Figure 5.20 *c)*). The shown videos consist of 16, 8, 60 and 20 frames, respectively. In each video two frames had to be annotated to obtain the shown results (Figure 5.20). On the average, 22 scribbles were placed in a video of 26 frames with an average resolution of 713×402 pixels. These scribbles are, on the average, assigned eight different disparities in the range of $[51, 243]$. All shown results use STC and CON. The videos in Figure 5.20 *a)*, *b)* and *d)* contain objects that move in depth. Thus, we additionally use our proposed temporal disparity change models to capture these changes (i.e., DC -sM). We obtained spatially and temporally coherent disparity maps that adapt well to the corresponding scenes. As can be seen in Figure 5.20 *a)* and *b)* (women moving in depth) our algorithm captures temporal disparity changes in a perceptually coherent manner. The 2D-to-3D conversion results that were obtained using DB (e.g., Figure 5.20 *a)* and *b)*) contain plausible disparities on slanted or rounded surfaces. The videos in Figure 5.20 *c)* and *d)* are obtained using the WTA scheme, resulting in less smooth, but likewise satisfactory disparity maps. It can be seen that our proposed algorithm is able to capture fine details (e.g., violin bow in Figure 5.20 *c)* or hair in *d)*) and provides hard disparity edges that coincide with object borders in the scene. Figure 5.21 shows novel views that were generated from our disparity maps using a professional stereoscopic software [144]. Specifically, we used our conversion results in Figure 5.21 *b)* and the given monoscopic video frame in Figure 5.21 *a)* to generate a novel view in Figure 5.21 *c)* for two of the test videos

---

the common GF that was used in Section 5.2.1.2 and Section 4.3.2. These spatio-temporal filtering steps consumes 15.2 and 0.1 seconds for the C++ implementation and CUDA implementation of GF, respectively.

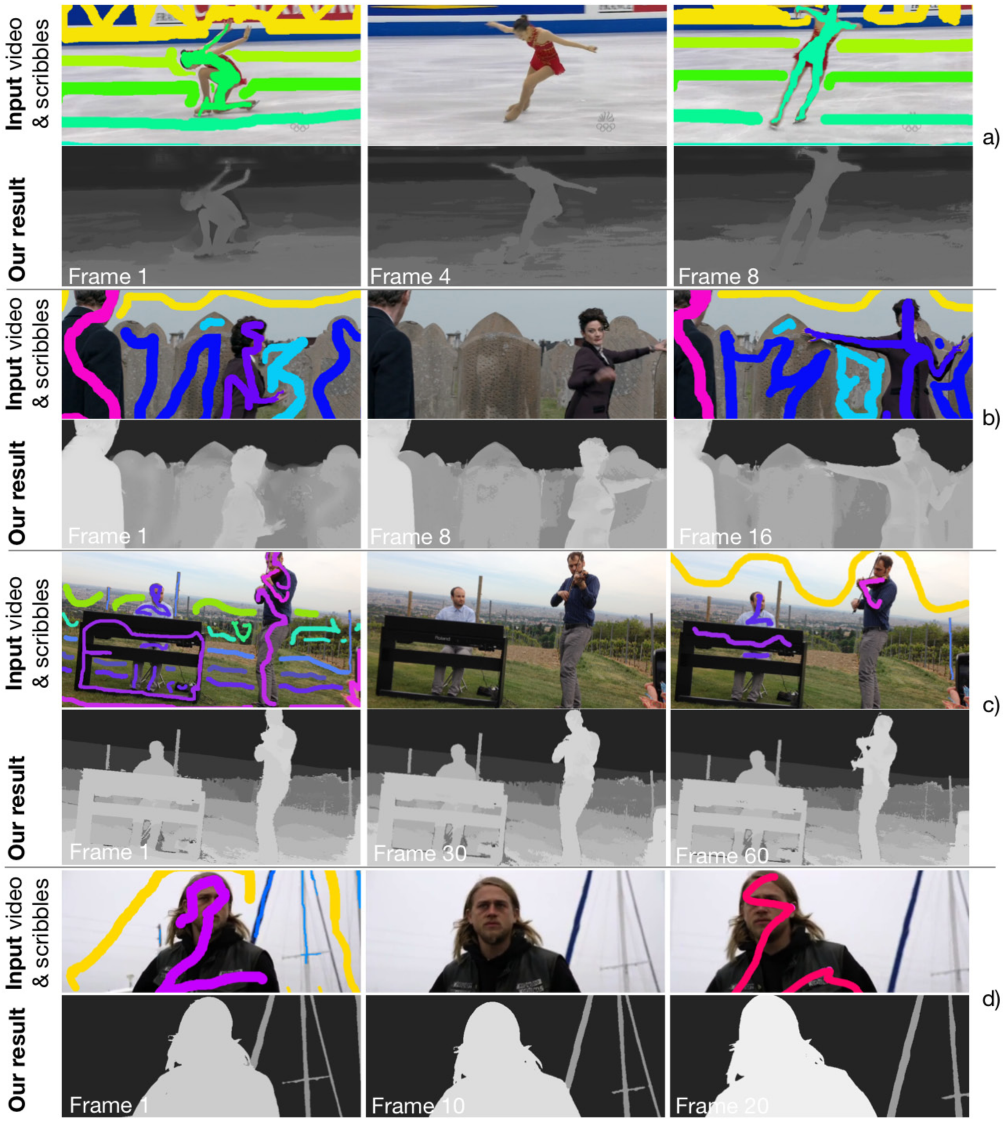


Figure 5.20: Our 2D-to-3D conversion results. Input video with disparity scribbles in the first and last frame of a video shot (*top*): Scribbles *hues* encode disparities. Obtained disparity maps for the shown example frames (*bottom*): *Bright* foreground, *dark* background. Copyright of original videos: *a)* NBC Olympics, *b)* BBC, *d)* Twentieth Century Fox.

shown in Figure 5.20. These examples demonstrate that our disparity maps are suited for further processing concerning (auto-)stereoscopic viewing conditions.

Comparison of different algorithm versions. We systematically evaluate the impact of the components of our 2D-to-3D conversion algorithm (Figure 5.5). We consider the following versions of our algorithm, in which different components are en- or disabled:

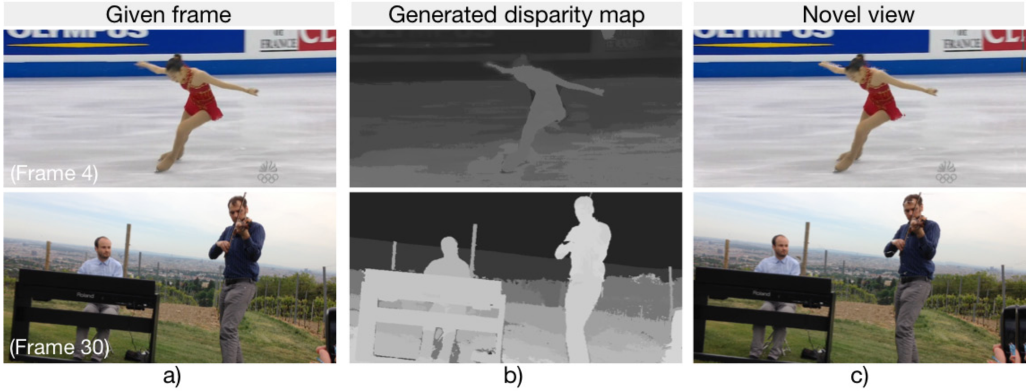


Figure 5.21: Novel views generated from our conversion results. *a)* Frame of given input video. *b)* Corresponding generated disparity map. *c)* Novel view generated from *a)* and *b)* using a professional stereoscopic software [144]. The conversion results that are used in this example correspond to videos shown in Figure 5.20. Copyright of original video *(top)*: NBC, Olympics.

- The *version MS* only enables the component MS (Figure 5.5). Thus, it is only based on color and cannot capture temporal disparity changes. It does not perform a temporal disparity interpolation and only assigns given disparities from the first frame.
- The *version STC* additionally applies the component STC (Figure 5.5).
- The version CON applies the components MS, STC and CON (Figure 5.5), but, as in the two previous versions, does not interpolate disparities over time.
- The *version DC -tM* additionally employs the time-based temporal disparity change model (Figure 5.5, DC -tM). Thus, given disparities from the first and last frame are interpolated guided by previously extracted depth order cues.
- The *version DC -sM* alternatively uses the segment-based temporal disparity change model (Figure 5.5, DC -sM) to interpolate disparities according to depth order cues.

For all versions mentioned above, we apply the common GF [59] during CVF [125] and our motion guided filtering (i.e., +TC). Since the evaluation results with WTA behave analogously to those with DB, we only list the former in Table 5.3. However, Appendix B (i.e., Table B.1) contains corresponding results for DB. The evaluations are performed on eight videos from [27] that were provided with depth GT (in meters)[11]. In Appendix A, we show visual examples of these videos and their GT. Like in our previously performed evaluations of 2D-to-3D conversion results, for each scribble we propagate the mean inverted[12] GT depth of all pixels that are marked by it. Table 5.3 reports the MSEs

---

[11] The authors of [27] provided us with GT depth data for test videos in [27]. These authors recently released an updated version of their data, i.e., the MPI Sintel Depth Training Data.

[12] Our CVF-based 2D-to-3D conversion algorithm assumes disparities with large values in the foreground and low values in the background. Hence, the GT depth has to be inverted before using it as input (and reference solution).

(Equation (4.8)) of the normalized inverted depth values[13] between the GT and our results. Our results in Table 5.3 were computed with GT OF from [27] and with estimated OF [96].

Let us first consider the results that were generated with WTA and GT OF (Table 5.3, *top*). When comparing the evaluation results of algorithm versions that use common GF with versions that use our motion guided filter (+TC), the latter on average reduce the MSEs by approximately 16 percent. Large improvements are observed for videos with fast motion, e.g. *Ambush2* in Figure 5.27 *a)* and *b)* (up to approximately 100 pixels between frames). This indicates that our motion guided filtering is important for videos with fast moving objects, while for videos with small motion common GF is sufficient. In some cases, we also observe slight degradations of the results when additionally applying +TC, e.g., *Sleeping1* in case of MS. These degradations can be explained by color ambiguities that are (as well) present at the adjusted positions of the filter windows. Table 5.3 further shows that both the closeness and the connectivity constraint have a positive effect on the results. Specifically, when comparing the MSEs of MS with those of STC and the MSEs of STC with those of CON, the additional constraint decreases the errors for nearly all test videos. As shown in Table 5.3, our temporal disparity interpolation can improve the results for videos that contain objects which exhibit motion in depth, compared to versions that do not capture temporal depth changes. The significantly reduced errors for *Shaman3* and *Sleeping1* especially stand out (i.e., from CON to DC -tM the MSEs decrease by a factor of 4.78 and 7.52, respectively). In both videos a camera zoom in the scene causes a large depth change for all objects (Figure 5.22). We also observe visual improvements for other videos that contain objects with smaller movement towards or farther away from the camera (e.g., *Temple3*, Figure 5.26). The MSEs of DC-tM and of DC-sM are similar. In this context, it is worth noting that the tested videos do not contain objects with irregular movement, e.g., objects that stop for a few frames, which are the focus of DC-sM.

As mentioned above, the results that were generated with WTA and with DB behave similarly (compare Table 5.3 and Table B.1). If enabling a component improves the WTA result, it typically also improves the DB result. Typically, videos that mainly contain objects that are more or less fronto-parallel (e.g., Figure 5.23 *b)*) perform better with WTA than DB. On the contrary, videos with slanted or rounded objects (e.g., Figure 5.23 *a)*) perform better with DB. Note that in the latter videos smooth transitions of scribble disparities (or, in the case of test videos from [27], depths) with overlapping color models are desired. While DB, in some cases, compensates for color ambiguities between close-by scribbles (e.g., Figure 5.23 *a)*, *green arrow*), in other cases DB's smoothing operation introduces errors for the same reason (e.g., Figure 5.23 *b)*, *red arrow*).

Comparison of naive and depth order guided interpolation. In the following, we compare equivalent versions of our algorithm that perform the temporal interpolation with and without taking depth order cues into account. Figure 5.24 and Figure 5.25 show examples of our 2D-to-3D conversion results that perform such a naive interpolation and corresponding results that use our depth order guided interpolation. In the shown examples, performing

---

[13] For each video, our conversion results and the GT are normalized by the same factor, i.e., their maximal depth. This prevents videos with large depth ranges from having larger errors than videos with narrow depth ranges.

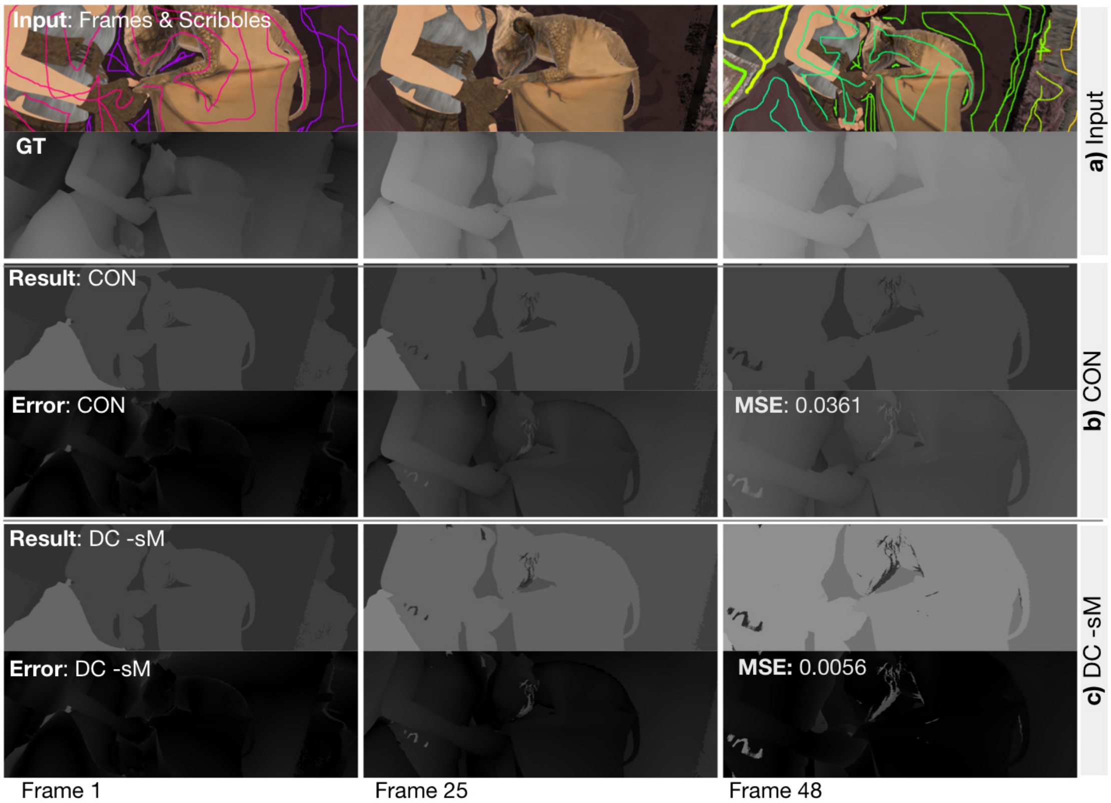


Figure 5.22: Comparison of proposed algorithm with dis- and enabled disparity change models (WTA, GT OF). *a)* Input video with scribbles (*top*) and depth GT (*bottom*). *b)* 2D-to-3D conversion results with disabled (i.e., CON) and *c)* with enabled disparity change models (i.e., DC - sM). Our results (*top*) and their errors to the GT (*bottom*). Error: *Black* small and *white* large errors. Result and GT: *Bright* foreground, *dark* background. Original video from [27].

naive interpolations for scribble pairs leads to perceptual conflicts (Figure 5.24, *red arrows*), while our depth guided interpolation improves the results concerning their perceptual coherence (Figure 5.24, *green arrows*). The achieved improvement is especially noticeable when comparing the results visually, while the corresponding quantitative evaluation results are very similar to each other. This may be explained by the fact that our depth order guided interpolation corrects naive interpolations that lead to a perceptual incoherency to the closest valid disparity (or, in the case of [27], inverted depth value). Nonetheless, we additionally provide quantitative evaluation results in Appendix B (Table B.2 and Table B.3).

Evaluation of sensitivity to quality of motion information. The MSEs of the algorithm versions in Table 5.3 were computed by using GT OF from [27] and by using estimated OF [96]. To better illustrate the difference in quality of the used OFs, we additionally list the *Endpoint Error* [6] (EE) that is computed between the GT OF and the estimated OF

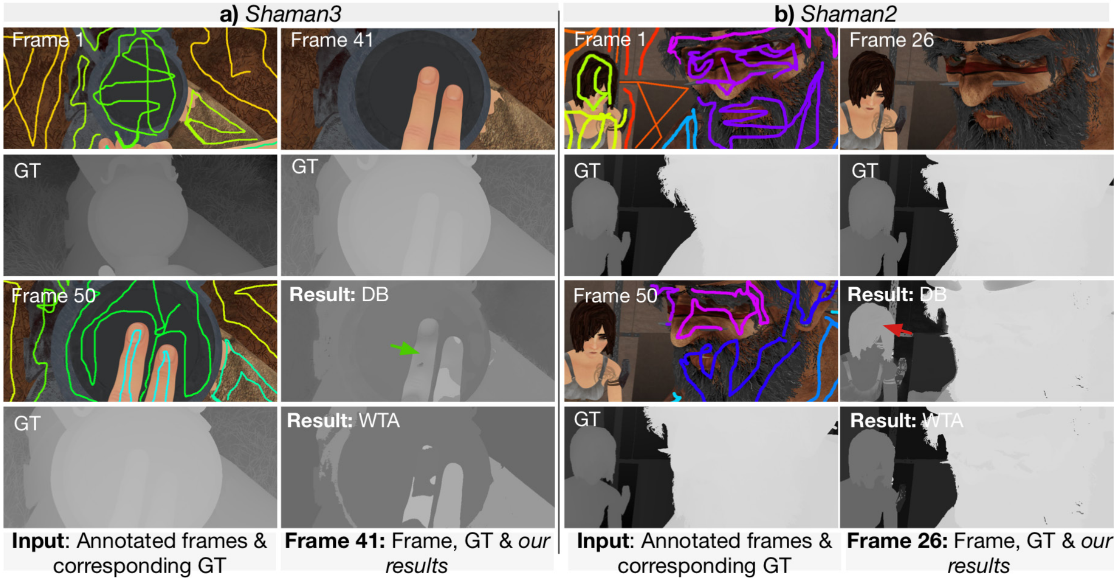

a) *Shaman3*
Frame 1
Frame 41
GT
GT
Frame 50
Result: DB
GT
Result: WTA
Input: Annotated frames & corresponding GT
Frame 41: Frame, GT & *our results*
b) *Shaman2*
Frame 1
Frame 26
GT
GT
Frame 50
Result: DB
GT
Result: WTA
Input: Annotated frames & corresponding GT
Frame 26: Frame, GT & *our results*

Figure 5.23: Comparison of assignment schemes (i.e., DB and WTA, GT OF). *a) Shaman3* (DC -tM). *b) Shaman2* (DC -sM). Per video: *Left*: Input frames with scribbles and depth GT. *Right*: Input frame, GT, our results with DB and WTA. (Front: *bright*, back: *dark*). In *a)* DB performs better (*green arrow*) than WTA, in *b)* vice versa (*red arrow*). Original videos from [27].

in Table 5.3. Clearly, the estimated OF is not ideal[14], i.e., on average the EE decreases by 12.75 percent when comparing the GT OF and the estimated OF.

When comparing the errors with GT OF and estimated OF in Table 5.3, on average the quality of our results is hardly affected by the change of used OF. When examining the results in detail, we observe that already the MSEs for MS differ. This is caused by the scribble matching and grouping results, which differ for different OF inputs. Consequently, the 2D-to-3D conversions are performed based on different color models. This also affects subsequent versions of the algorithm. Furthermore, erroneously matched scribble pairs might lead to perceptual conflicts in the conversion results that were generated using a naive interpolation technique. In fact, in this evaluation of our algorithm's sensitivity to the quality of used OF, the difference between the scribble matching and grouping results with GT OF and the scribble matching and grouping results with estimated OF emerged as a main reason for the de- and increases of MSEs (in Table 5.3, *bottom* versus *top*). In this context, Figure 5.26 and Figure 5.27 show our results for two videos, i.e., *Temple3* and *Ambush2*, that were obtained with GT OF and estimated OF. For *Temple3*, the results with estimated OF consider the scribbles on the wing (Figure 5.26, first frame *turquoise*, last frame *violet*) as a scribble pair, which is not the case with GT OF. The resulting additionally performed temporal interpolation leads to smaller MSEs with estimated OF. Contrary, for *Ambush2*, MSEs increase when using estimated OF instead of GT OF. Figure 5.27

---

[14] It should be noted that we did not optimize the parameters of the used optical flow estimation algorithm [96].

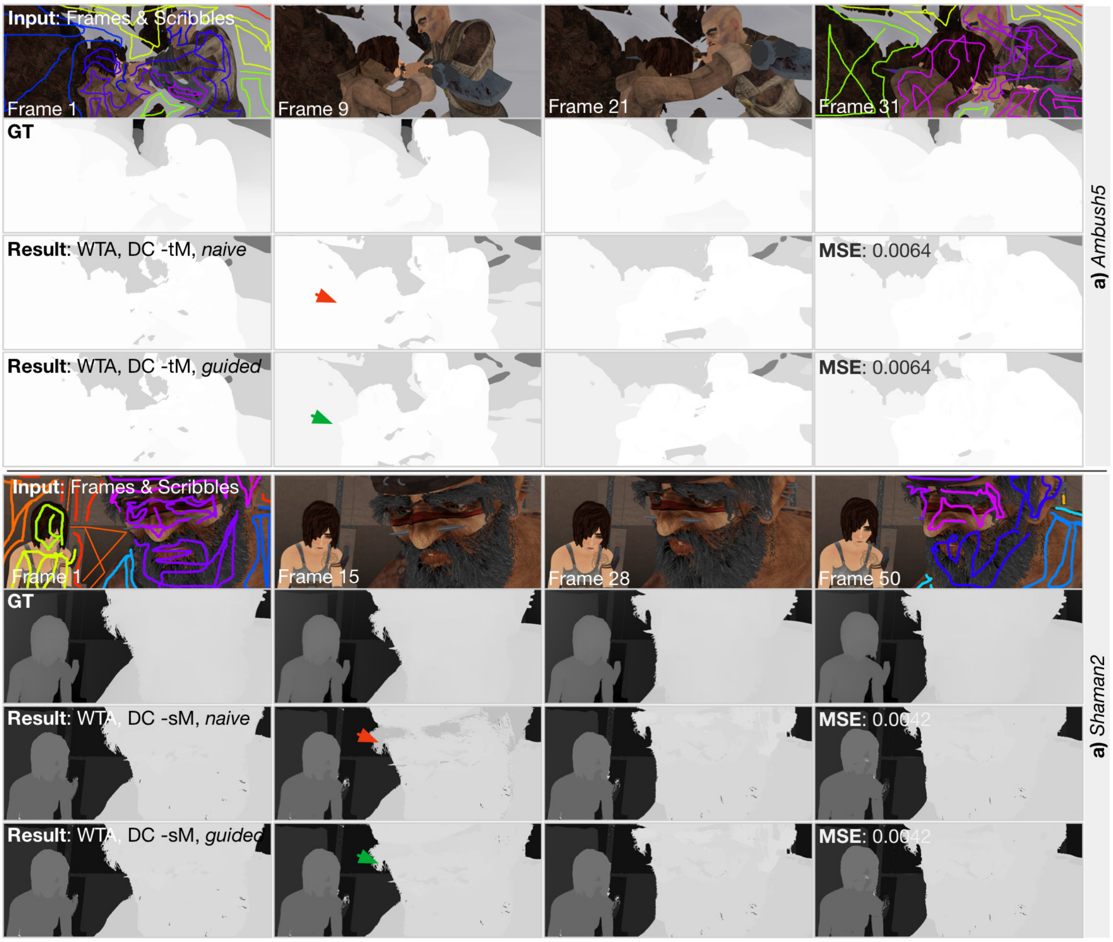

Input: Frames & Scribbles
Frame 1
Frame 9
Frame 21
Frame 31
GT
Result: WTA, DC -tM, naive
MSE: 0.0064
Result: WTA, DC -tM, guided
MSE: 0.0064
a) Ambush5
Input: Frames & Scribbles
Frame 1
Frame 15
Frame 28
Frame 50
GT
Result: WTA, DC -sM, naive
MSE: 0.0042
Result: WTA, DC -sM, guided
MSE: 0.0042
a) Shaman2

Figure 5.24: Comparison of interpolation techniques (GT OF). Input video with scribbles in the first and last frame and GT (*top*). Foreground: *bright*, background: *dark*. Our 2D-to-3D conversion results (*bottom*) that were obtained when using a *naive interpolation* and our *depth order guided interpolation* for *a) Ambush5* and *b) Shaman2*. Perceptual incoherencies (*red arrows*) are corrected (*green arrows*) by our guided interpolation. Original videos from [27].

highlights one difference in scribble matching, where only the scribble pair found with estimated OF causes a temporal interpolation (different depth in first and last frame). For both OFs, some scribbles (e.g., Figure 5.27, *blue arrow*) cannot be tracked throughout the video, thus, for these scribbles the components CON and STC cannot be applied.

As for GT OF, for estimated OF, we observe improvements when additionally enabling +TC, STC, CON and DC-sM or -tM (Table 5.3, *bottom*). Figure 5.25 and Figure 5.27 have already provided visual examples of such improvements when using estimated OF. In Figure 5.25, perceptual conflicts that arise in naive temporal interpolations are corrected by our depth order guided interpolations. In Figure 5.27 +TC improves the 2D-to-3D conversion results compared to a version with common GF.

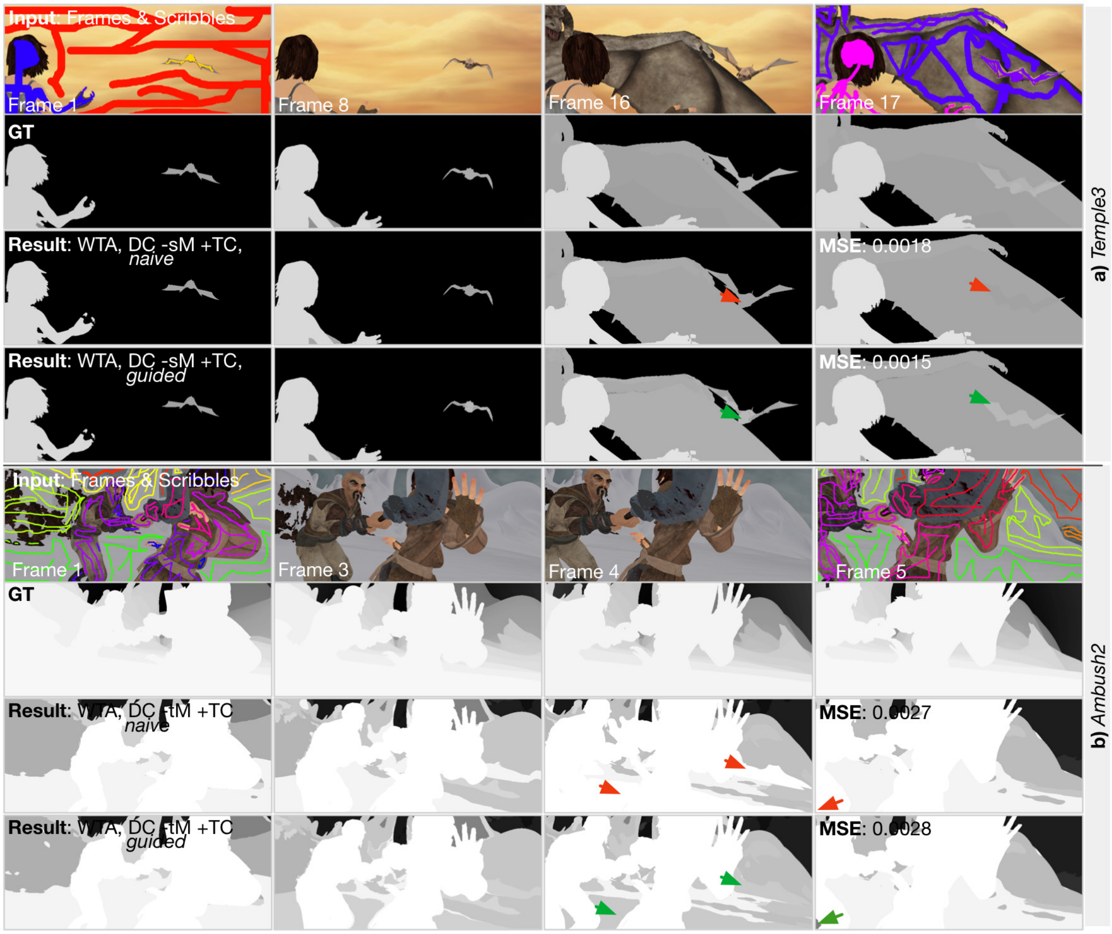

Input: Frames & Scribbles
Frame 1
Frame 8
Frame 16
Frame 17
GT
Result: WTA, DC -sM +TC, naive
MSE: 0.0018
Result: WTA, DC -sM +TC, guided
MSE: 0.0015
a) Temple3
Input: Frames & Scribbles
Frame 1
Frame 3
Frame 4
Frame 5
GT
Result: WTA, DC -tM +TC naive
MSE: 0.0027
Result: WTA, DC -tM +TC guided
MSE: 0.0028
b) Ambush2

Figure 5.25: Comparison of interpolation techniques (estimated OF). Input video with scribbles in the first and last frame and depth GT (*top*). Foreground: *bright*, background: *dark*. Our 2D-to-3D conversion results (*bottom*) that were obtained when using a *naive interpolation* and our *depth order guided interpolation* for *a) Temple3* and *b) Ambush2*. Perceptual incoherencies (*red arrows*) are corrected (*green arrows*) by our guided interpolation. Original videos from [27].

Evaluation of sensitivity to illumination effects. While the videos that were used in the previous experiments contain challenging color similarities, they were rendered (*albedo rendering*) without artistic and illumination effects such as artistic color corrections, motion blur, reflections and shadows. To test the sensitivity of our algorithm to such effects, we re-evaluate it using the same algorithm versions, scribbles, parameters, GT OF and scenes as in Table 5.3, but with *final renderings* that contain such effects. Appendix A provides visual examples for both renderings of the scenes and their depth GT. Table 5.4 lists the corresponding MSEs with WTA. When comparing the MSEs of the final rendering (Table 5.4) with those that were obtained with the albedo rendering (Table 5.3, *top*), a significant decrease in quality can be observed (i.e., factor 3.84 for WTA). In Table 5.4,

Table 5.3: Comparison of different algorithm versions (WTA). The table lists the mean squared error (MSE) of our results averaged over all pixels and multiplied by 100. Our algorithm is tested in different versions (see text) and with ground truth optical flow (GT OF) (*top*) and estimated OF (*bottom*). The endpoint error (EE) (*right*) measures the accuracy of the used OF.

MSE	using GT OF										**EE**
	MS		**STC**		**CON**		**DC -tM**		**DC -sM**		**OF**
		+TC		**+TC**		**+TC**		**+TC**		**+TC**	
Alley1	0.06	0.06	0.05	0.05	0.04	0.04	0.04	0.04	0.04	0.04	0.00
Ambush2	1.19	0.27	1.16	0.24	1.16	0.24	1.16	0.23	1.16	0.23	0.00
Ambush5	1.45	1.44	0.65	0.65	0.63	0.65	0.64	0.64	0.64	0.64	0.00
Ambush7	0.99	0.99	0.53	0.53	0.54	0.54	0.47	0.48	0.47	0.48	0.00
Shaman2	0.79	0.79	0.57	0.56	0.42	0.41	0.41	0.41	0.42	0.42	0.00
Shaman3	1.91	1.91	1.90	1.90	1.96	1.96	0.41	0.40	0.51	0.49	0.00
Sleeping1	3.65	3.66	3.62	3.62	3.61	3.61	0.48	0.48	0.56	0.57	0.00
Temple3	0.28	0.15	0.27	0.21	0.27	0.21	0.27	0.21	0.27	0.21	0.00

MSE	using estimated OF										**EE**
	MS		**STC**		**CON**		**DC -tM**		**DC -sM**		**OF**
		+TC		**+TC**		**+TC**		**+TC**		**+TC**	
Alley1	0.07	0.07	0.05	0.03	0.04	0.04	0.05	0.05	0.05	0.05	1.73
Ambush2	1.22	0.31	1.30	0.29	1.30	0.29	1.23	0.28	1.23	0.28	73.68
Ambush5	1.46	1.47	0.69	0.68	0.69	0.68	0.67	0.66	0.66	0.66	7.12
Ambush7	0.96	0.96	0.50	0.50	0.46	0.46	0.47	0.47	0.47	0.47	2.20
Shaman2	0.78	0.79	0.55	0.55	0.40	0.39	0.39	0.38	0.40	0.39	0.60
Shaman3	1.91	1.91	1.90	1.90	1.92	1.92	0.30	0.34	0.40	0.42	1.20
Sleeping1	3.65	3.66	3.61	3.61	3.61	3.61	0.56	0.56	0.60	0.61	2.20
Temple3	0.28	0.15	0.27	0.14	0.28	0.15	0.28	0.15	0.28	0.15	13.29

MS exhibits the largest MSEs for most videos and has large differences to corresponding MSEs in Table 5.3, *top*. The main reason for the increased MSEs in Table 5.4 are color ambiguities, object colors that were not covered by scribbles and, in some similarity to the previously performed evaluation of the sensitivity to motion information, wrongly paired or grouped scribbles. In Table 5.4, compared to MS, the MSEs are reduced with STC and CON. This indicates an increase of robustness to illumination effects when additionally constraining the results by spatial closeness and the 3D connectivity to the scribbles. However, when comparing the MSEs of STC and CON for the final rendering (Table 5.4) with those for the albedo rendering (Table 5.3, *top*), we conclude that illumination effects remain challenging. Figure 5.28 and Figure 5.29 versus Figure 5.26 visually compare the conversion results for different renderings of two test videos, i.e., *Ambush7* and *Temple3*. For *Ambush7*, Figure 5.28 shows that overlapping color models of scribbles (Figure 5.28 *b)*, handle and snow or glove) are especially challenging with DB. In such cases increasing the influence of the spatial constraint by adjusting t_{close} can lead to improved results (Figure 5.28, $t_{close} = 256$). Figure 5.29 provides a more detailed example for *Temple3*'s final rendering that can be compared with corresponding results for its albedo rendering in Figure 5.26. In addition to the obtained conversion result (Figure 5.29 *b)*), the correspond-

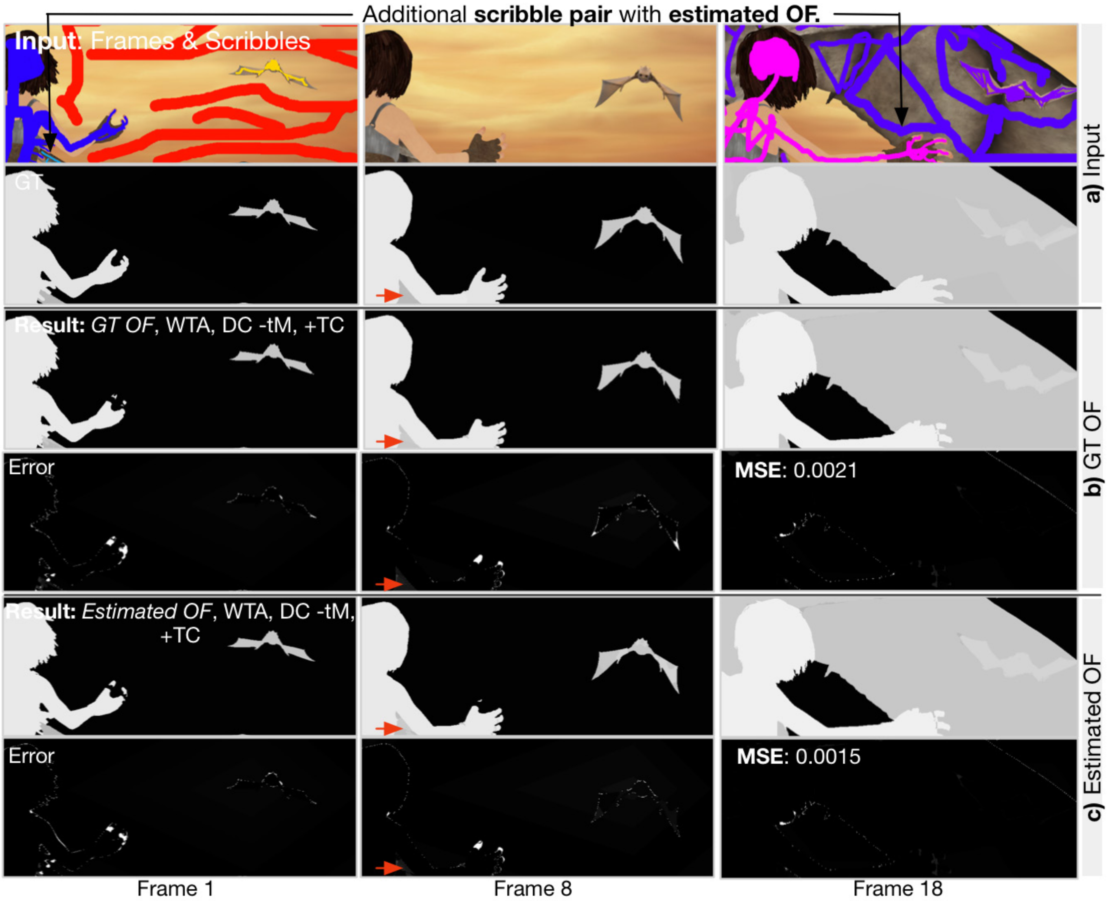

Additional scribble pair with estimated OF.
Input: Frames & Scribbles
GT
a) Input
Result: GT OF, WTA, DC -tM, +TC
Error
MSE: 0.0021
b) GT OF
Result: Estimated OF, WTA, DC -tM, +TC
Error
MSE: 0.0015
c) Estimated OF
Frame 1
Frame 8
Frame 18

Figure 5.26: Comparison of results with different motion information. *a)* Input video (*Temple3*) with scribbles and depth GT. Our results (DC -tM +TC, WTA) *b)* with GT OF and *c)* estimated OF: Conversion (*top*) and its error to the GT (*bottom*). *Red arrows*: Area that is closer to the GT with estimated OF. This is caused by an additional scribble pair (first frame *turquoise*, last frame *violet*) with estimated OF that temporally interpolates the depth for this area. Results: *Bright* fore- and *dark* background. Error: *Black* small and *white* large error. Original video from [27].

ing segmentation $\mathcal{R}$ (Figure 5.29 *d)*) and (tracked) scribble positions (Figure 5.29 *c)*) are shown. This example illustrates three sources of error that are also linked to illumination effects. First, as a result of a missed scribble pair (Figure 5.29 *a)*, scribbles on small dragon) due to color differences between the marked pixels in the first and the last frame, the temporal depth interpolation for the small dragon was not performed. Second, the obtained result contains erroneous assignments (e.g., Figure 5.29 *b)*, *yellow arrows*) that were caused by color ambiguities between objects (e.g., Figure 5.29 *a)*, *yellow arrow*). While in many cases such errors are prevented by the components STC and CON, this is not the case for the scribbles on the large dragon's wing (Figure 5.29 *c)*, *violet* scribbles in last frame). Since these scribbles are not present in the first frames, the components

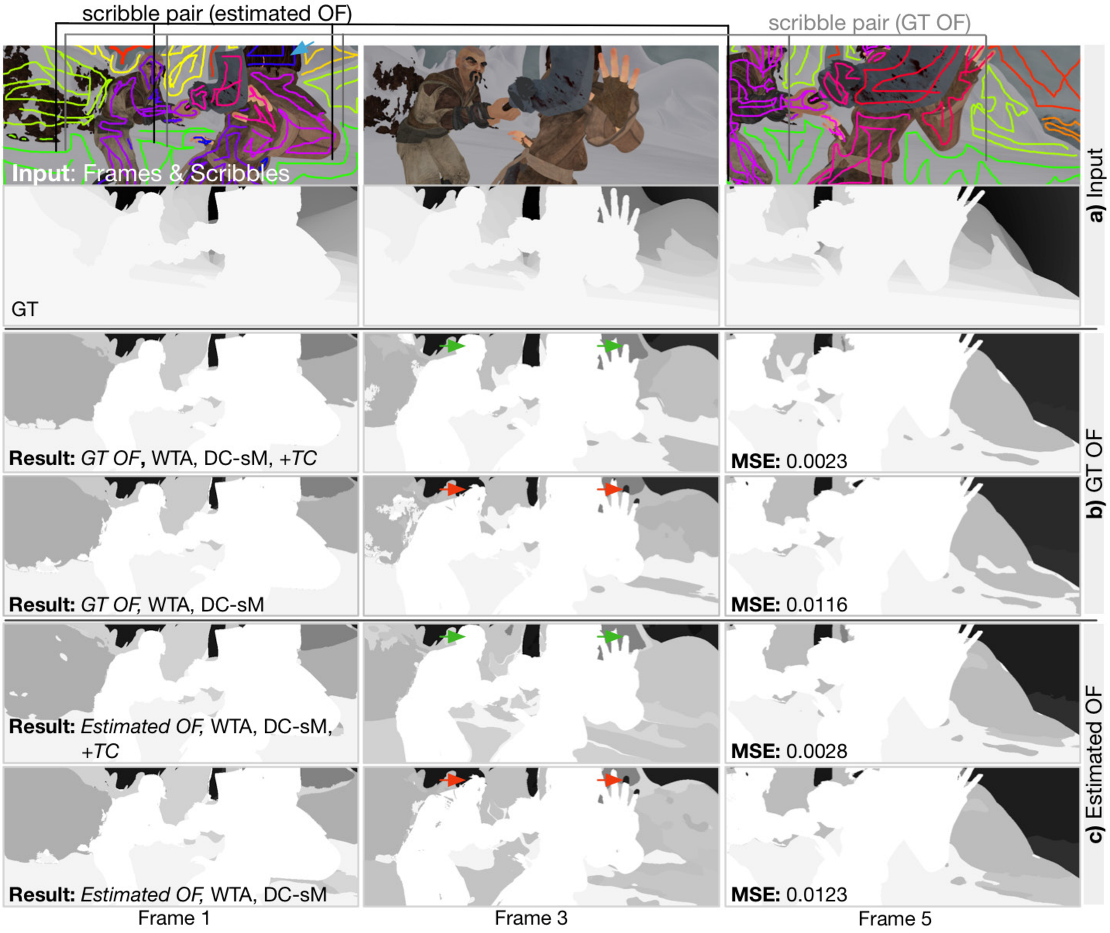


Figure 5.27: Comparison of results with different motion information. *a)* Input video (*Ambush2*) with scribbles (*top*) and depth GT (*bottom*). Our conversion results (i.e., DC -sM (+TC) WTA) *b)* with GT OF and *c)* with estimated OF. Foreground: *bright*, background: *dark*. *Red arrows* indicate areas that are improved (*green arrows*) by enabling +TC. *Blue arrow* points to a scribble that could not be tracked throughout the video for both OFs. Original video from [27].

STC and CON cannot be applied for them. Third, the example contains an erroneous depth interpolation (Figure 5.29, *blue arrow*) which is caused by an erroneously extracted depth order. As discussed in Section 5.2.2, the depth order is determined from occlusions between segments in $\mathcal{R}$. Thus, $\mathcal{R}$s that are severely affected by color ambiguities, such as in Figure 5.29 *d)*, can lead to erroneous temporal depth interpolations.

Comparison to related 2D-to-3D conversion algorithms. We compare our CVF-based algorithm to our segmentation-based conversion (GS+P) from Chapter 4, our implementation of Guttmann et al.'s algorithm [56] and a video version of Phan et al.'s algorithm [118][15]

---

[15] Phan et al.'s algorithm [118] was provided by the authors. The main difference to their originally presented

Table 5.4: Evaluation of sensitivity to illumination effects (WTA, GT OF). The final renderings of the test videos are used. The table lists the mean squared error (MSE) of our results averaged over all pixels and multiplied by 100. Our algorithm is tested in different versions (see text).

MSE	WTA, using GT OF									
	MS		**STC**		**CON**		**DC -tM**		**DC -sM**	
		+TC		**+TC**		**+TC**		**+TC**		**+TC**
Alley1	0.77	0.78	0.78	0.79	0.77	0.77	0.81	0.48	0.79	0.81
Ambush2	5.34	4.58	4.33	4.44	4.29	4.68	4.15	4.64	4.15	4.64
Ambush5	1.28	1.34	0.69	0.69	0.68	0.68	0.74	0.73	0.77	0.76
Ambush7	0.83	0.82	0.60	0.60	0.50	0.50	0.54	0.61	0.56	0.68
Shaman2	7.94	7.96	3.92	3.09	4.07	4.11	4.86	4.75	4.91	4.77
Shaman3	3.97	3.97	3.78	3.81	3.92	3.95	1.89	1.71	2.22	1.99
Sleeping1	3.44	3.45	3.56	3.67	3.66	3.68	0.97	1.02	1.02	1.08
Temple3	9.87	8.83	7.61	6.69	7.41	6.47	7.44	6.47	7.42	6.51

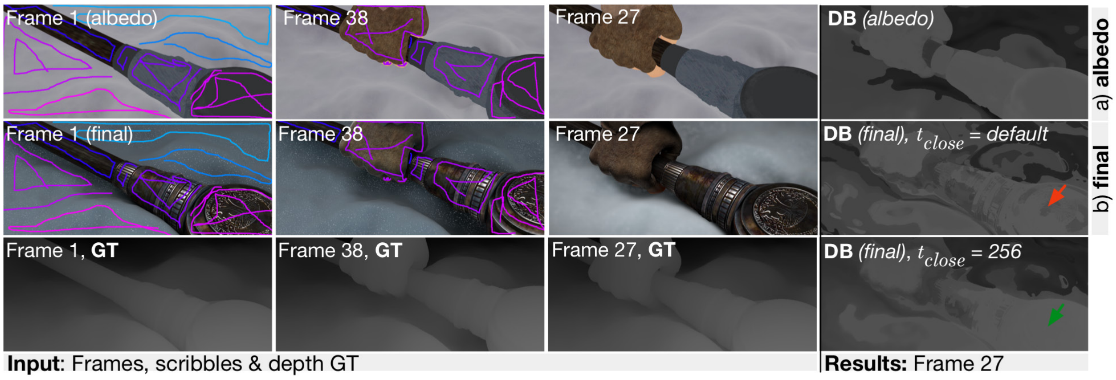


Figure 5.28: Visual evaluation of sensitivity to illumination effects. *Left:* Input frames with scribbles and depth GT. *a)* albedo rendering and *b)* final rendering of *Ambush7*. Scribble positions and GT remain the same. *Right:* Our results (with DB, CON, +TC, GT OF). Newly introduced ambiguities (*red arrow*) between objects and colors that were not covered by scribbles in *b)* decrease the quality of our results. Adjusting t_{close} can improve (*green arrow*) the results obtained for *b)*. Foreground: *bright*, background: *dark*. Original videos from [27].

that is, like ours, based on an interactive segmentation approach [19]. For both algorithms parameters that were suggested by the authors in [56] and [118] were used. The evaluation is performed on a dataset that contains five recorded videos with stereo generated (using [12]) reference solutions (i.e., *City*, *Parade*, *Palace*, *Stairs* and *Football*), videos with up to 101 frames that are provided with disparity and depth GT (i.e., *Child*, *Head*, *Interview*) and three computer-generated videos with GT disparity maps from the new Tsukuba dataset [102] (i.e., *Tsukuba1*, *Tsukuba50*, *Tsukuba380*). In Appendix A, we provide visual examples for all 11 videos including their reference solutions. To evaluate the

---

image version is that their video version applies their conversion algorithm volumetrically on the entire video.

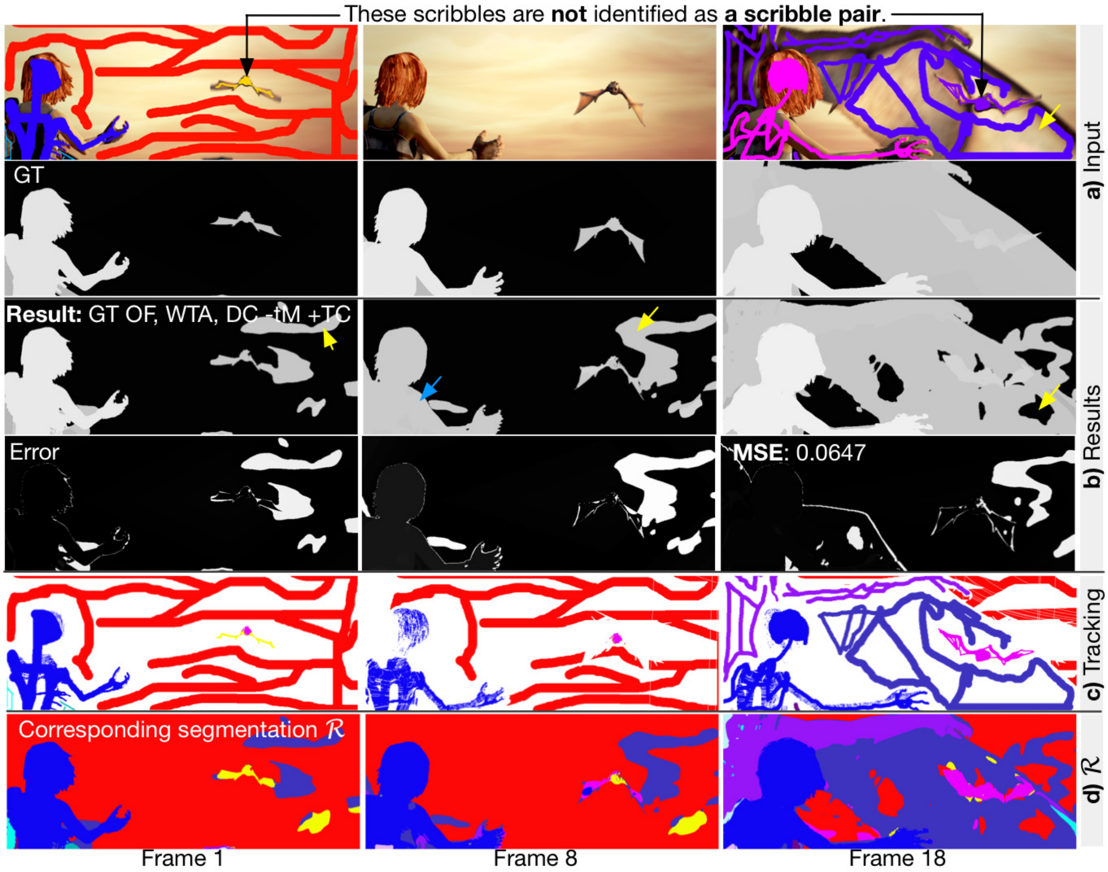

These scribbles are **not** identified as **a scribble pair.**
GT
Result: GT OF, WTA, DC -tM +TC
Error
MSE: 0.0647
Corresponding segmentation $\mathcal{R}$
Frame 1
Frame 8
Frame 18
a) Input
b) Results
c) Tracking
d) $\mathcal{R}$

Figure 5.29: Failure case due to illumination effects. *a)* Input video (i.e., *Temple3*) with scribbles (*top*) and depth GT (*bottom*). *b)* Our result (with DC -tM, +TC, WTA, GT OF) that was generated for the final rendering, i.e., conversion result (*top*) and its error to the GT (*bottom*). Result and GT: *bright* foreground, *dark* background. Error: *Black* small error, *white* large error. *c)* (Tracked) scribbles. Paired and grouped scribbles have the same color. *d)* Multi-label segmentation $\mathcal{R}$: Segment colors correspond to scribble colors in *c)*. *Yellow arrows:* Errors due to color ambiguities. *Blue arrow:* Erroneous depth interpolation. Original video from [27].

quality of the algorithms' results, we compare them to their respective reference solutions. We employ a similar strategy as in previous 2D-to-3D conversion evaluations, i.e., the algorithms propagate the reference solution at the scribble positions. Since our CVF-based algorithm and [118] are based on scribbles that indicate a single disparity, we use the mean reference solution of each scribble for all tested algorithms for a fair comparison.

Table 5.5 lists the measured errors for each tested algorithm on this dataset. For our CVF-based algorithm, we list the best version (using estimated OF [96]) for each label assignment scheme, i.e., WTA and DB. The shown MSEs indicate that our CVF-based algorithm achieves competitive 2D-to-3D conversion results. It outperforms the previous work of Guttmann et al. [56] (that uses estimated OF from [110]) on ten test videos and Phan et al.'s [118] (without OF) on nine videos. For five of the test videos, i.e., *Palace*, *Child*, *Head*, *Tsukuba50*, *Tsukuba1*, we also achieve better results than GS+P (Chapter 4). It is important to note that these videos contain (slight) motion in depth (e.g., due to camera movement). Figure 5.33 exemplarily shows the results for *Tsukuba1*, which exhibits the largest temporal disparity change, Figure 5.30 for *Parade*, Figure 5.31 for *Stairs* and Figure 5.32 *e)* for *Palace*. It can be seen (Figure 5.31 *e)*, Figure 5.33 *e)* and Table 5.5) that our CVF-based algorithm produces plausible conversions that also capture the change in disparity in a video. This is not the case for all tested algorithms, e.g., Phan et al.'s [118] algorithm (Figure 5.33 *c)*, Figure 5.31 *d)*) does not address the problem of temporal disparity changes due to object motion or of perceptual coherence. Instead disparities from the first and the last frame are propagated independently from each other. However, it is fair noting that further developments of [118] in [117] (naively) address temporal disparity changes when converting 2D videos to 3D. GS+P captures temporal disparity changes, however, in the shown example (Figure 5.33 *d)*) the results contain artifacts. These artifacts are caused by a temporal disparity interpolation that is performed within multiple small segments with different temporal extent. Our 2D-to-3D conversions contain hard disparity edges near object borders (e.g., Figure 5.30 *e)*, Figure 5.31 *e)*, Figure 5.32 *d)-e)* and Figure 5.33 *e)*). Contrary to previous work where the disparity assignment also depends on a global optimization, i.e., [56, 118], our algorithm does not suffer from over-smoothed disparities at object borders. While in some cases this effect might be desired to add volume to objects (e.g., Figure 5.30 *d)*, buildings), in other cases it unrealistically bends foreground objects towards the background (e.g., Figure 5.30 *d)*, person in the foreground). When examining our results in detail, we notice that most videos that contain temporal disparity changes, e.g., *City*, *Palace*, *Stairs*, *Head*, *Tsukuba1*, *Tsukuba50* and *Tsukuba380*, perform best when disparity change models are enabled. This observation highlights the importance of temporal disparity interpolations in context of 2D-to-3D conversion. Furthermore, test videos that exhibit large motion, e.g., *Child*, *Tsukuba1* and *Tsukuba380*, achieve the lowest MSEs when using a version with motion guided filtering (+TC), which confirms its usefulness for such scenes. We observe limitations for videos that contain (close-by) objects with similar colors and scribble annotations that result in overlapping color models (e.g., Figure 5.31 *e)* and Figure 5.32 *d)*, *red arrows*). Concerning these limitations, STC and CON improved our results compared to MS. In fact, in Table 5.5 MS is not listed, i.e., MS was never the version that exhibited the smallest MSE.

5.4 Summary

In this chapter we have first proposed an interactive video object segmentation and matting framework for videos that obtains spatio-temporally coherent segmentations at interactive rates (i.e., 250 fps for a video with a resolution of 620×360 pixels per frame). Its user interface allows performing the segmentation based on sparse user input (i.e., few foreground scribbles). Our main contribution was to extend an interactive single image segmentation approach that is based on spatial cost volume filtering to the temporal domain. We have qualitatively and quantitatively

Table 5.5: Comparison to related 2D-to-3D conversion algorithms. Our segmentation-based conversion (GS+P, Chapter 4), our implementations of [56] and [118] are compared to reference solutions. For our CVF-based algorithm (WTA and DB) we list the best versions for each video. The table lists the mean squared error (MSE) averaged over all pixels and multiplied by 100.

MSE	**WTA**	**DB**	**WTA**	**DB**	**GS+P**	**[56]**	**[118]**
City	DC -tM +TC	DC -sM	1.08	1.06	0.47	1.24	1.08
Parade	STC	STC	0.74	0.63	0.28	0.99	0.85
Palace	DC -tM +TC	CON +TC	1.16	1.31	1.20	1.56	1.14
Stairs	DC -tM +TC	DC -tM +TC	0.86	0.96	0.51	0.72	0.60
Football	STC	STC	0.51	0.52	0.40	0.57	0.64
Child	CON +TC	CON +TC	0.57	0.55	0.58	1.09	1.13
Head	DC -tM +TC	DC -tM +TC	0.49	0.44	0.65	4.68	1.45
Interview	CON +TC	CON +TC	0.80	1.10	0.56	12.76	15.57
Tsukuba50	DC -tM +TC	DC -tM +TC	0.15	0.17	0.15	2.61	1.92
Tsukuba380	DC -tM +TC	CON +TC	0.44	0.54	0.21	2.22	0.69
Tsukuba1	DC -tM +TC	DC -tM +TC	0.10	0.09	0.15	2.22	0.79

shown that additional aggregation of costs across frames significantly reduces flickering in the segmentation results. We have further demonstrated that our proposed approach outperforms previous video segmentation algorithms that have similar runtime capabilities as ours.

The semi-automatic 2D-to-3D conversion algorithm that was presented in the main part of this chapter has taken a step towards the generation of perceptually coherent disparity maps. With our object segmentation algorithm as foundation, we were able to use spatio-temporal segmentation information to capture hard edges in disparity maps and perform smooth disparity interpolations over time. These disparity interpolations were performed in accordance with motion-caused occlusions. Evaluations demonstrated that our proposed algorithm generates plausible disparity maps that capture the disparity change of dynamic objects in a 2D video shot. Enabling different components of our algorithm decreased the error rates of our results, e.g., additionally using a motion guided filtering instead of a common filter operation decreased the error rates by approximately 16 percent. Further evaluations of our algorithm's sensitivity to motion information revealed that our scribble matching and grouping is often influenced by the motion information used. This suggests it would be beneficial to support manual adjustments of the scribble matching and grouping results during processing. In presence of perceptual conflicts in conversion results that were generated using a naive disparity interpolation, our proposed disparity interpolation has demonstrated its ability to improve the 2D-to-3D conversion results. Our systematic evaluation not only revealed the mentioned strengths, but also some weaknesses, e.g., sensitivity to illumination effects and color ambiguities, of our algorithm. In comparison to related semi-automatic 2D-to-3D conversion algorithms, our CVF-based algorithm generated highly competitive results on a set of recorded and computer-generated 2D videos and outperformed a well-established 2D-to-3D conversion algorithm on nearly all tested videos. In this context, key advantages of our approach were the reduction of over-smoothing and the additional capture of temporally changing disparities in a spatio-temporal and perceptual coherent manner.

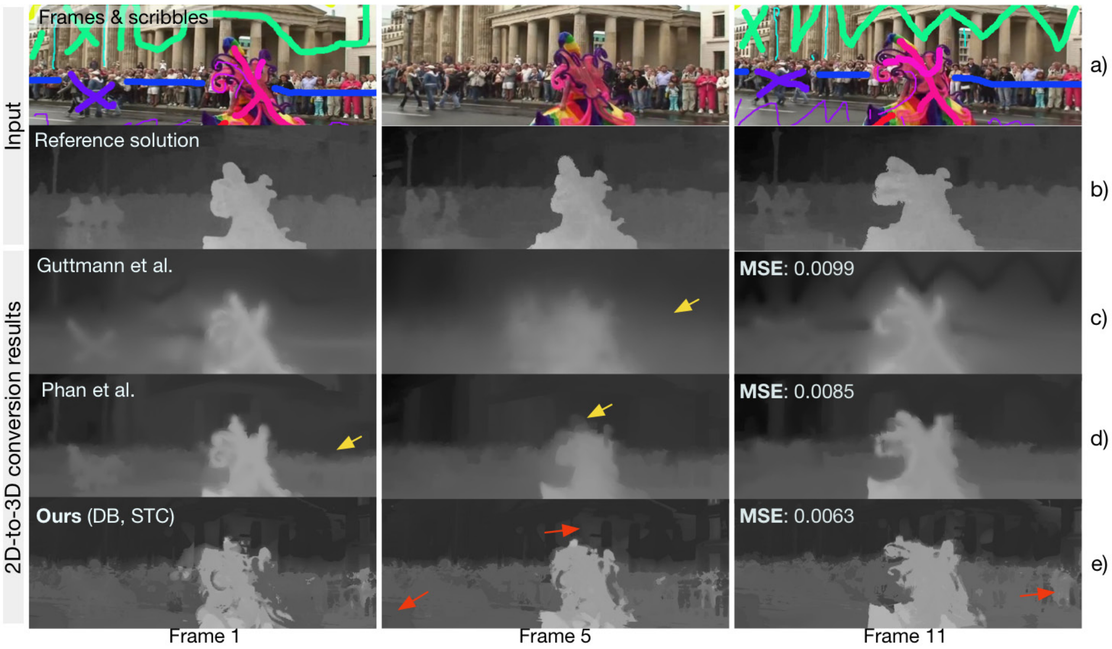


Figure 5.30: Visual evaluation and comparison to [56] and [118] (*Parade*). *a)* Frames and scribbles. *b)* Reference solution. Foreground: *bright*, background: *dark*. *c)* Result obtained with [56] and *d)* with [118]. *Yellow arrows* highlight over-smoothing. *e)* Our CVF-based results: Hard disparity edges at object borders. *Red arrows*: Errors due to overlapping color models.

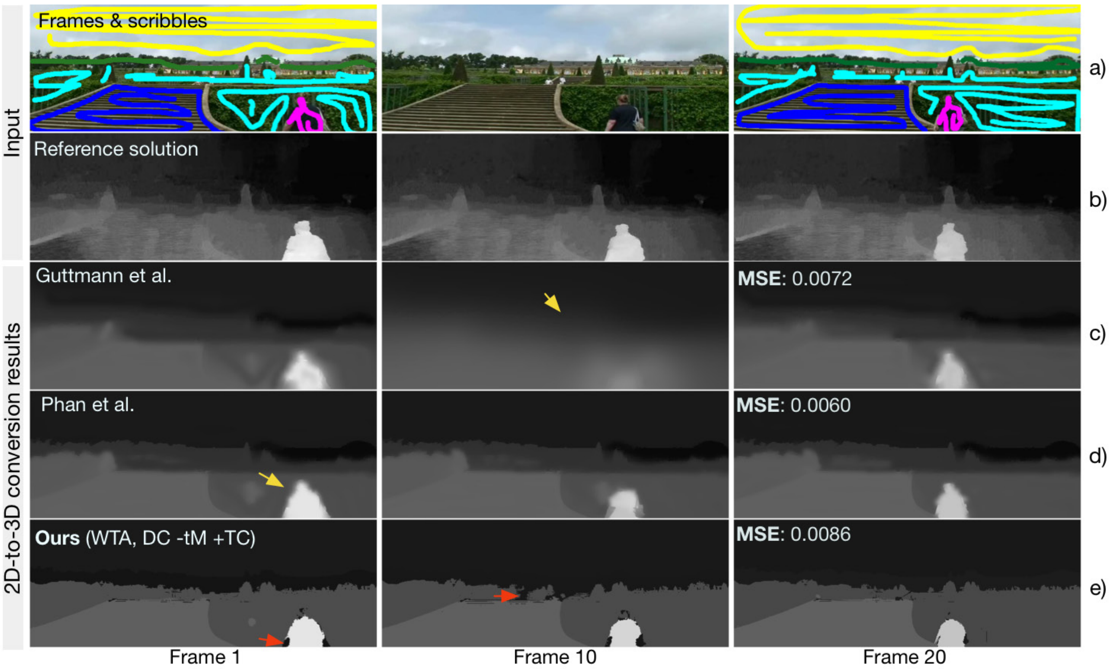


Figure 5.31: Visual evaluation and comparison to [56] and [118] (*Stairs*). *a)* Frames and scribbles. *b)* Reference solution. Foreground: *bright*, background: *dark*. Result obtained with *c)* [56] and *d)* [118]. *e)* Our CVF-based WTA results: Contrary to *c)* and *d)*, in *e)* the disparity change (person approaching stairs) is captured. Compared to *c)*, *e)* has hard disparity edges at object borders, but no smooth transitions between different disparities (e.g., on stairs) to indicate slanted surfaces. *Yellow arrows*: Over-smoothing. *Red arrows*: Errors due to overlapping color models.

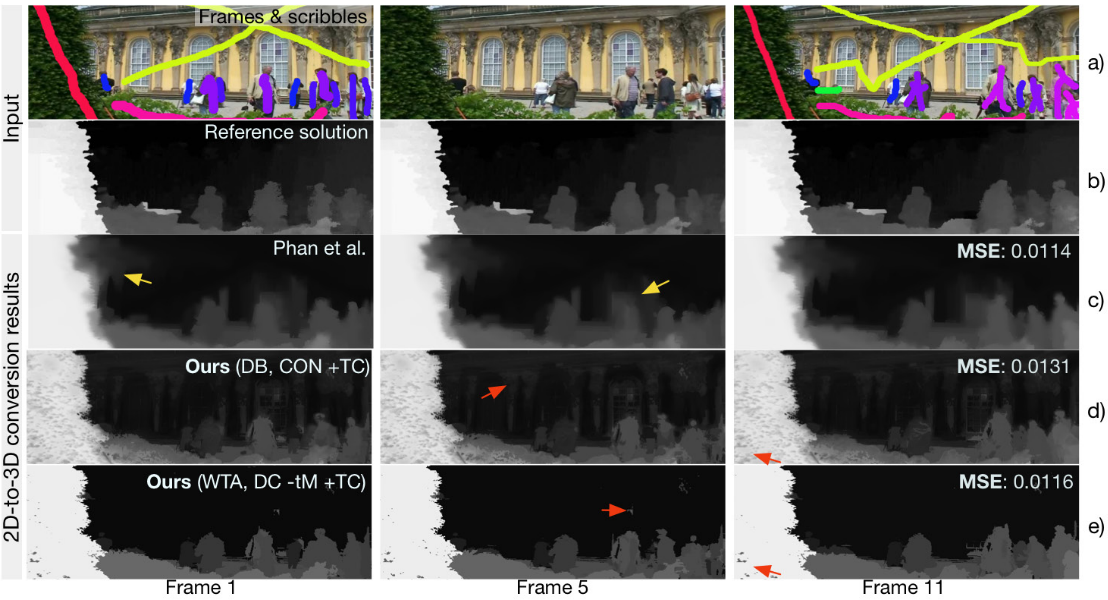


Figure 5.32: Visual evaluation and comparison to [118]. *a)* Frames and scribbles from *Palace*. *b)* Reference solution. Foreground: *bright*, background: *dark*. *c)* Result obtained with [118]: *Yellow arrows*: Over-smoothing. *d)* and *e)* Our CVF-based results. *d)* and *e)* have hard disparity edges at object borders compared to *c)*. *Red arrows*: Errors due to overlapping color models.

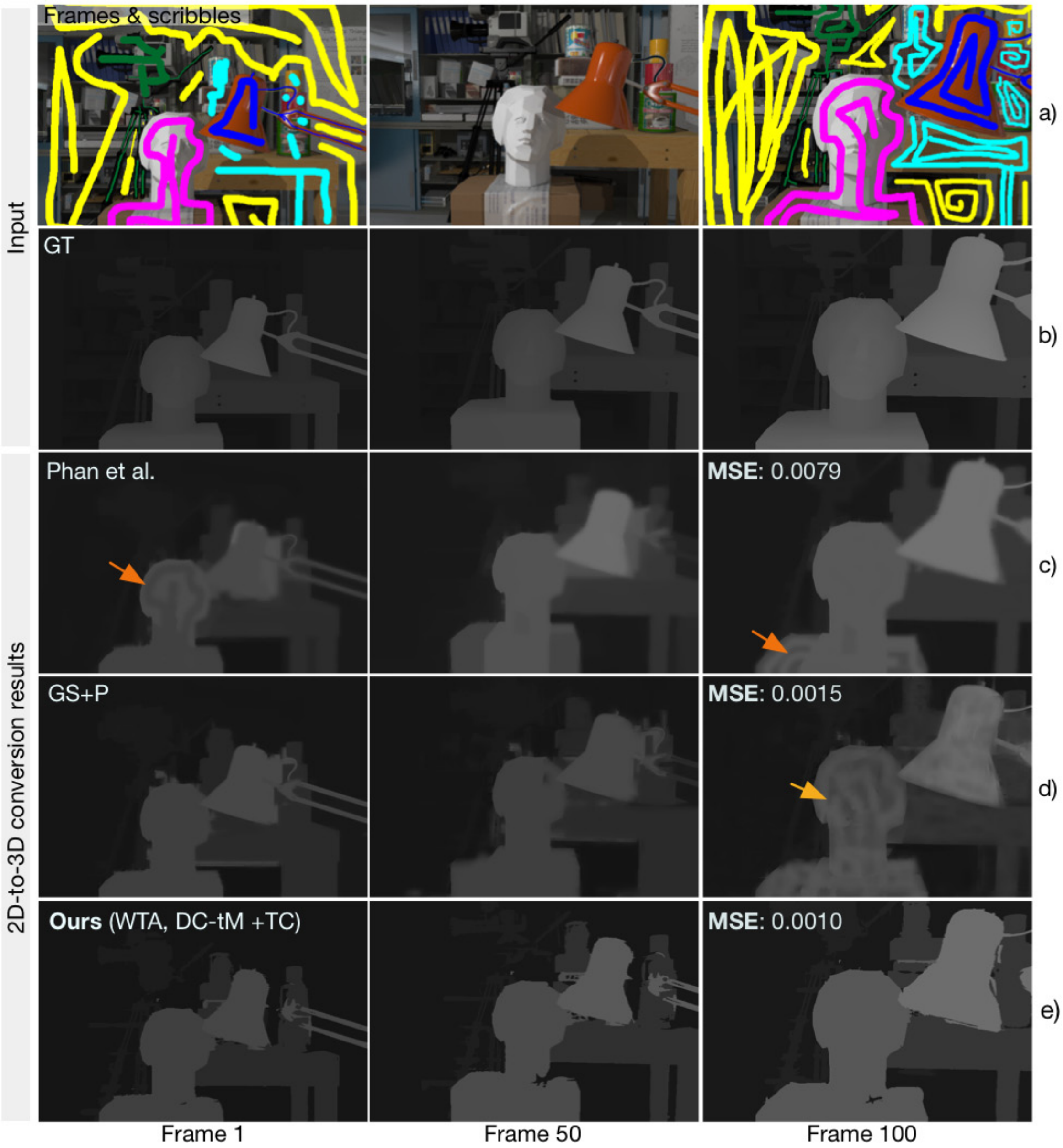


Figure 5.33: Visual evaluation and comparison to [118]. *a)* Frames and scribbles from *Tsukuba1*. *b)* GT disparity maps: *Bright* foreground, *dark* background. *c)* Result from Phan et al. [118]. *d)* Our joint segmentation and propagation results (GS+P, Chapter 4). *e)* Our CVF-based results: Contrary to *c)* and *d)*, in *e)* the disparity change due to a camera zoom is captured evenly. *Orange arrows* highlight errors in *c)* and *d)*. Original video from [102].

CHAPTER 6

Evaluation of 2D-to-3D Conversion Systems in Conjunction with User Input

6.1 Introduction

Semi-automatic 2D-to-3D conversion systems are strongly based on user input (e.g., scribble-based user annotations) to initialize their conversions. This dependency of conversion systems on user input, e.g., on scribble placement or on a user preferred scribbling strategy, can influence the conversion results. Not only a user, but also a 2D-to-3D conversion system might prefer one scribbling strategy over another. For instance, a small scribble in the center of an object can lead to different conversion results than a large scribble of the same disparity that roughly traces the object border. We already observed similar effects. In Chapter 4, we have compared different 2D-to-3D conversion results that were initialized with the same disparities and generated with the same system, but using different scribbling strategies. In some similarity, in Chapter 5, we have kept the scribbles constant, but used different renderings (that considered different illuminations) of the same 2D scenes. Thus, the scribbles covered the same but differently colored pixels as in the initial rendering. Likewise, the 2D-to-3D conversion results that were obtained for the different renderings of the same scene were quite different. Although a given user input matter-of-factly constitutes an important factor when generating 3D content, it is typically neglected when evaluating 2D-to-3D conversions, e.g., [56, 82, 156]. As the mentioned examples from the literature, the evaluations that were performed in the previous chapters mainly focus on the comparison of 2D-to-3D conversion systems when given the same fixed scribbles without considering their placement or the user's preferred scribbling strategy.

The main contribution of this chapter is a systematic evaluation of semi-automatic 2D-to-3D conversion systems under consideration of the user input. Our evaluations focus on systems that work with scribble-based user annotations, as and including those that were presented in this thesis. We automatically generate various sets of scribble inputs that simulate users who take on different scribbling strategies, including minimalistic and more labor-intensive ones (Section 6.2). The 3D content, generated based on these scribbles, is then compared with reference solutions that encode ground truth (GT) depth information of the processed 2D data. In the first experiment (Section 6.3.1), we compare the accuracy of 2D-to-3D conversion systems and scribbling strategies. We then investigate the effect of small scribble perturbations on the 2D-to-3D conversion results (Section 6.3.2). Since scribble-based user annotations are not necessarily accurate, e.g., a scribble with a foreground disparity might accidentally cover some pixels that should be assigned to a background disparity, our third experiment addresses the robustness to such errors (Section 6.3.3). Our fourth series of experiments (Section 6.3.4) is an in-depth evaluation of the tested 2D-to-3D conversion systems and scribbling strategies under consideration of the 2D content. Specifically, we investigate which scribbling strategy is ideal for which region in a 2D image (e.g., homogenous regions or multicolored regions). With these four experiments, we reveal strengths and weaknesses of the tested 2D-to-3D conversion systems in conjunction with different scribbling strategies and provide practical insights concerning the scribble-based annotation process. We believe that both aspects are useful for future developments in the field of 2D-to-3D conversion and the efficient usage of such systems.

In the 2D-to-3D conversion literature, evaluations focus on quantitative comparisons of conversion results with GT (disparities or depths) or other reference solutions (e.g., [28, 31, 56, 58, 94, 156, 159, 175]), user studies in which participants subjectively assess the results' quality (e.g., [28, 56, 92, 94, 117, 163, 175]) or sole visual comparisons of shown results (e.g., [29, 41, 44, 82, 91, 116, 118, 130, 165, 169]). In these evaluations the user annotations are initially performed and kept fixed. In [92], the authors include the usability of the given 2D-to-3D conversion system in their evaluations by measuring the annotation times required by users to convert 2D content to 3D. Contrary to our field of 2D-to-3D conversion, in the related field of interactive segmentation, the user-centric component of segmentation systems is more often considered during evaluation. In [76], Kohli et al. propose a framework which simulates an interactive user who evaluates interactive segmentation algorithms. Starting with an initial set of fixed scribbles, their simulated user places scribbles in the largest wrongly segmented region (comparison with segmentation GT) until a maximum number of allowed scribbles is reached. In some similarity to our work, an experiment in [76] compares simulated users with different strategies for automated scribble placement. Related works also include evaluation studies concerning the error tolerance of interactive segmentation algorithms when inaccurate scribble annotations are given (e.g., [3]), objective and subjective evaluations concerning the effect and user preference of different interaction methods in the field of interactive segmentation (e.g., [60]) and discussions concerning the minimum number and placement of scribbles in the context of image colorization algorithms (e.g. [37, 128]). Since all of the previously mentioned evaluation studies are performed in different fields, there is no strong overlap with our evaluation study. However, the previously mentioned evaluation studies motivated our work. Inspired by [60] and [76] we simulate users who take on different scribbling strategies and study their effect on the obtained 2D-to-3D conversion results. Like [3], we investigate the effect of inaccurate scribbles on generated results and in some similarity to [37, 128], we are interested in finding guidelines for an ideal scribble placement to further facilitate and speed-up the annotation process for users.

6.2 Evaluation Strategy

We perform our evaluation study on a dataset that contains 19 still images and reference solutions from [134] and [27]. The benchmark setup, including the test data, is discussed in Section 6.2.1. The tested 2D-to-3D conversion systems are discussed in Section 6.2.3. Our study evaluates each conversion system by annotating each image with 41 different, automatically generated sets of scribble input. The generation of these scribbles is discussed in Section 6.2.2. Due the large number of different scribble-sets per image and conversion system, we perform an objective evaluation study in which the 2D-to-3D conversion results are compared with corresponding reference solutions. The large amount of generated 3D content, i.e., $5 \times 41 = 205$ conversions per image, is the main factor for our decision to perform an objective evaluation study on images instead of videos or subjective assessments of the result's quality.

6.2.1 Benchmark Setup

Test data and reference solutions. Our evaluation study is performed using a dataset (Figure 6.1) that contains 19 still images and corresponding reference solutions; 15 images are taken from the Middlebury stereo dataset [134] and four images are taken from the MPI Sintel dataset [27]. The Middlebury stereo dataset provides GT disparity maps for 2D images. The authors of the MPI Sintel dataset[1] provided us with GT scene depths (in meters) for their 2D content. To derive reference solutions that can act as disparity maps, i.e., small values in the background and large values in the foreground, the depth data from [27] is inverted. In both cases, we normalize the GT by the maximal (disparity) value per image, resulting in normalized GTs in the range of $[0, 1]$ that act as reference solutions in our evaluations. Although the reference solutions were derived from two different, but (inversely proportional) related units (see Chapter 2 for details), i.e., disparity (in pixels) and depth (in meters), the inversion of the depth data and the normalization to the range of $[0, 1]$ align them and make it possible to use them together in a single evaluation study. Since scribble-based annotations only initialize a few distinct disparities[2] and typically do not cover the entire range of disparities given in a GT, we further modify the range of the reference solutions. The reference solution of an image contains the same set of disparities as the scribbles that are used to annotate the image. In a nutshell, our test data is accompanied by reference solutions that encode the depth of each pixel as normalized "disparity", which is large in the foreground and small in the background. The data used (Figure 6.1) contains computer-generated (e.g., *Ambush2*) and recorded (e.g., *Aloe*) images. Scenes with multicolored objects (e.g., *Aloe*), low contrast scenes (e.g., *Lampshade1*), cluttered scenes (e.g., *Dolls*) or color similarities between objects at different disparities (e.g., *Temple3*) pose challenges to 2D-to-3D conversion systems. Generally, the computer-generated 2D

---

[1] The same authors later released training data with depth GT, stereo videos with disparity GT and segmentation GT [27, 166]. Since disparity GT was not available at the time our evaluations were performed, we use depth GT that was provided earlier for us by the authors of [27].

[2] Scribbles that initialize 2D-to-3D conversions typically result in a similar, but simpler depth impression than provided by a depth or disparity GT (reduced amount of input scribbles). For example, less salient slanted or rounded surfaces are often annotated with a single scribble and, hence, are approximated to fronto-parallel ones.

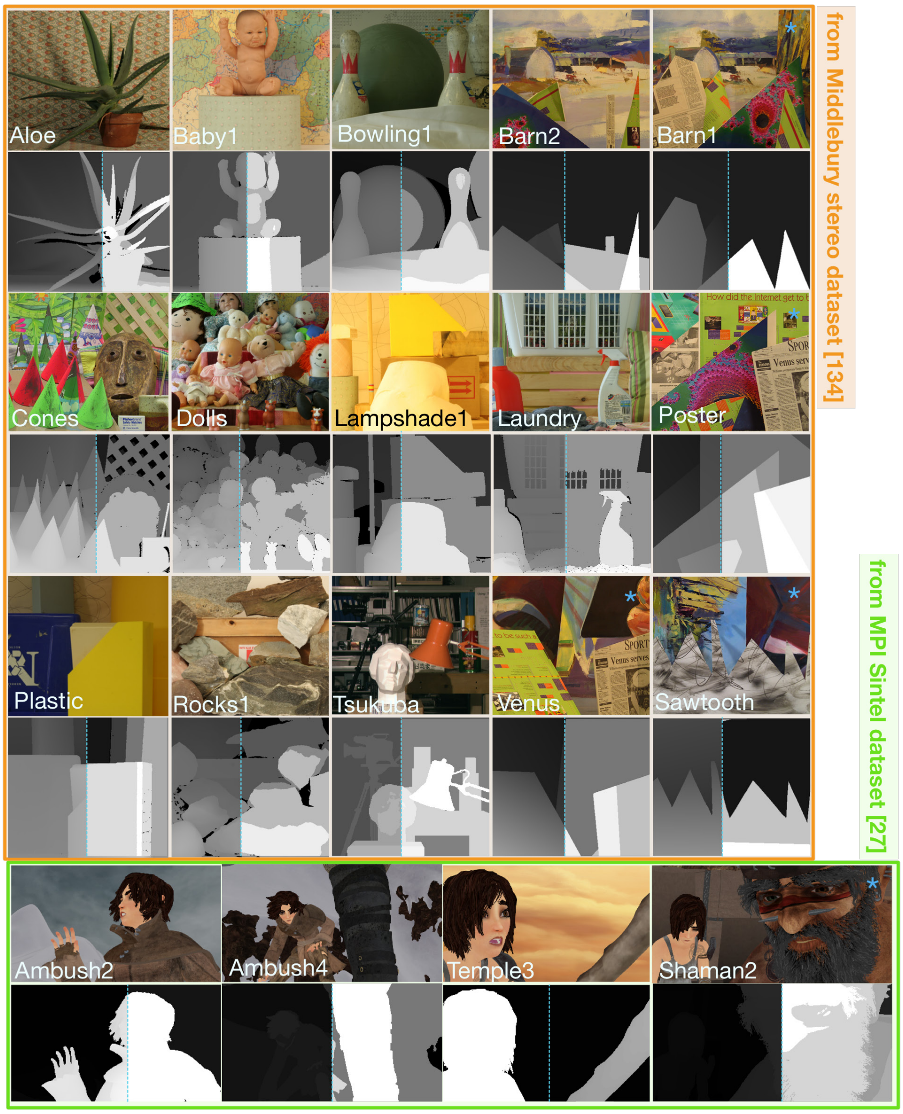


Figure 6.1: Overview of test data. 2D images (*top*) and reference solutions (*below*), i.e., disparities (*orange*) and inverse depths (*green*) from [134] and [27]. The GT that was originally provided with the 2D images (*left of dotted line*) and our derived, normalized reference solutions (*right of dotted line*) are shown (background: *dark*, foreground: *bright*, missing GT pixels in [134]: *black*). Marked (*blue asterisk*) images are removed for a smaller version of the dataset.

images from [27] have sharper object borders (color) than those from [134], but also exhibit large color overlaps between objects at different disparities and multicolored objects that are located at a single disparity.

GT scribbles. During this evaluation study, we use the same strategy as in our previously performed evaluation of 2D-to-3D conversion results, i.e., we propagate the reference solution at the scribble positions instead of directly using a disparity that is encoded by a scribble color. Thus, our scribbles do not introduce inaccurate choices of scribble disparities. As will be further detailed in Section 6.2.2 and Section 6.2.3, each scribble encodes exactly one disparity. All investigated 2D-to-3D conversion systems are tested with the same scribbles that encode the same disparities and the same 2D images.

Quality metric. We use a quantitative error metric that estimates the quality of the obtained 2D-to-3D conversion results. We compare the obtained results, i.e., the generated disparity maps, with the reference disparity maps that were derived from an image's GT. More precisely, we follow [56] and compute the *mean squared error* (MSE) (Equation (4.8)) between the generated disparity maps and the reference disparity maps and average the error over all pixels in our dataset that have a reference value. Since this averaged MSE can be distorted due to outliers, e.g., large MSE only for one particular image and low MSEs for the remaining images, we additionally provide the *average ranks with respect to the four investigated scribbling strategies* ($\in [1,4]$) and the *average ranks with respect to the five tested 2D-to-3D conversion systems* ($\in [1,5]$). In case of the former, a rank is computed for each 2D image, where 1 refers to the scribbling strategy with the smallest and 4 to the scribbling strategy with the largest MSE for a particular image. These ranks are averaged over the entire dataset to obtain the corresponding average rank. The ranks with respect to the 2D-to-3D conversion systems are computed analogously.

6.2.2 Scribble Generation

In our evaluation study scribble-based annotations are performed automatically instead of manually by a user. Inspired by [76], we simulate different users that take on different scribbling strategies, i.e., placing scribbles of different sizes at different positions in the 2D images. The basic idea behind our scribble generation algorithm is to generate them based on the GT of a 2D image. Similar to users who annotate a 2D image with scribbles of constant disparity, we first abstract its GT by reducing it to several layers (fusing several disparities to one). Likewise, in our experience[3], a user who is faced with the task of annotating 2D content with disparity scribbles often follows a similar pattern: Typically, a user focuses on drawing a few rough scribbles to indicate the main disparity layers in the scene. The starting point of this process is either the annotation of the object(s) farthest away or the annotation of the object(s) closest to the camera, as a base layer. This base layer often acts as a reference for selecting the disparities for further scribbles. This means, users often draw scribbles following a (descending or ascending) disparity order, using the previously drawn scribbles as a reference for the choice of the next disparity. On a related note, it is worth mentioning that this pattern conforms with observations that were made in cognitive studies [75, 155]. These studies indicate that assessing one point as closer or farther from the camera than another point can be considered a simple and natural task. The range of the GT (Figure 6.2 *b)*) is reduced to, on the average, seven distinct values per image

---

[3] Our experience is based on performing scribble-based annotation for 2D-to-3D conversion on multiple occasions, including to generate the results shown in this thesis, and discussing the process of annotating 2D content with students who were faced with the same task. Unfortunately, no user study could be found in the literature to further validate our experiences. We believe that such a study would be an interesting topic for future work.

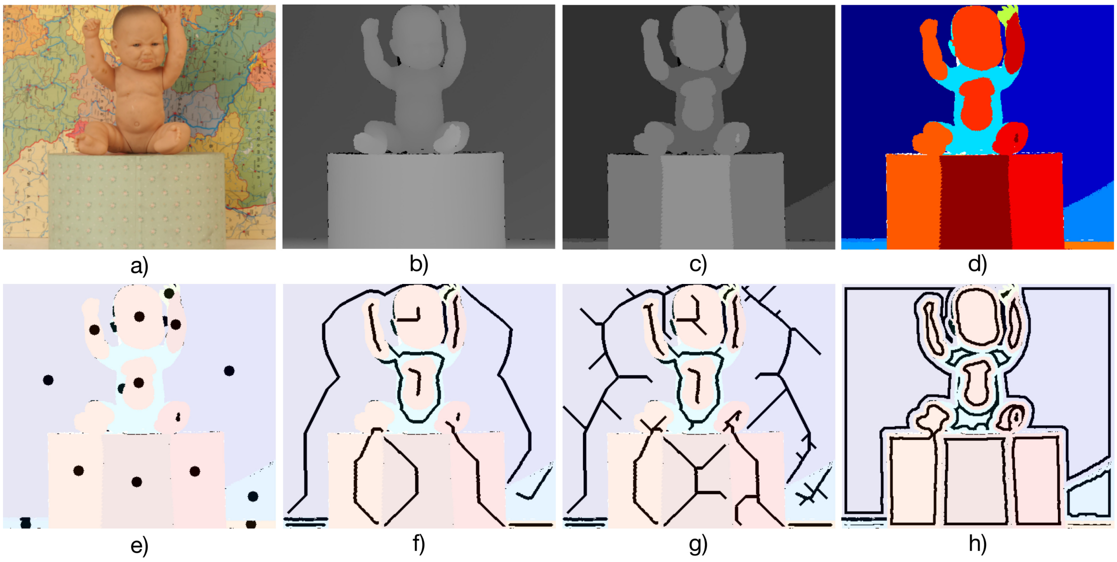


Figure 6.2: Scribble generation and scribbling strategies. *a)* 2D image (*Baby1* [134]), *b)* its GT and *c)* GT with reduced range. *Bright* fore- and *dark* background, missing GT: *black*. Color-coded disparity regions in *d)* have a single disparity in *c)*. Regions have different colors in *d)*. In *e)-h)*, we automatically place a scribble (*black*) in each region. Scribbles follow different scribbling strategies: *e)* point scribbles, *f)* line scribbles, *g)* expert scribbles and *h)* border scribbles.

(Figure 6.2 *c)*). This results in spatially connected *disparity regions* that each exhibit a single disparity in our reference solutions (Figure 6.2 *d)*) and mimic the main layers that a user would annotate. These regions are the foundation of our scribble generation process. Using different scribbling strategies, we automatically place a scribble in each of these regions if GT is provided for them. Figure 6.2 *e)-h)* gives an overview of the considered scribbling strategies, i.e., *point scribbles* (PS), *line scribbles* (LS), *expert scribbles* (ES) and *border scribbles* (BS). Below we discuss their characteristics and their generation.

Point scribbles (PS) refer to the most minimalistic scribbling strategy in our evaluation study. Similar to [76], our simulated novice user places a brush in the form of a dot with a maximum size (radius of eight pixels) in the middle (i.e., centroid or closest pixel to it within the region) of each disparity region (e.g., regions in Figure 6.2 *d)* and PS in *e)*). Thus, this scribbling strategy would be effortless when performed by an actual user considering two aspects: (1) the low number of marked pixels and (2) the distance of each scribble from object borders. Concerning the latter, Kohli et al. [76] observed that aiming for precise annotations close to object borders is perceived as a hard task. However, the small number of marked pixel colors might not be enough to properly define a scribble in terms of the underlying pixel's color distribution.

Line scribbles (LS) follow a scribbling strategy that we often observed in the literature of both fields, 2D-to-3D conversion and interactive segmentation. Many researchers, e.g., [41, 56, 76, 92, 117, 118, 125, 163], draw scribbles in form of lines that spread over a large extent

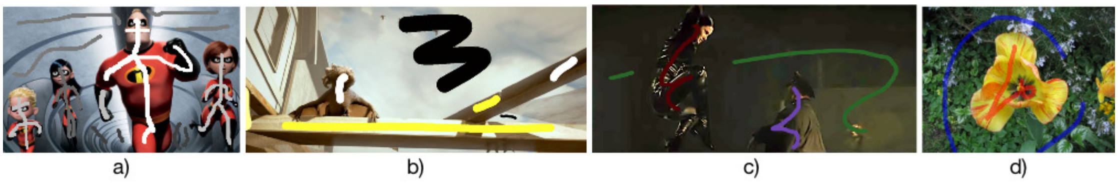


Figure 6.3: Frequently employed scribbling strategy. Independent of the scribbles' color-codings and meanings, user-centric systems, such as *a)* [163], *b)* [118], *c)* [56] and *d)* [125], are often initialized by scribbles in form of lines that are placed in the middle of objects.

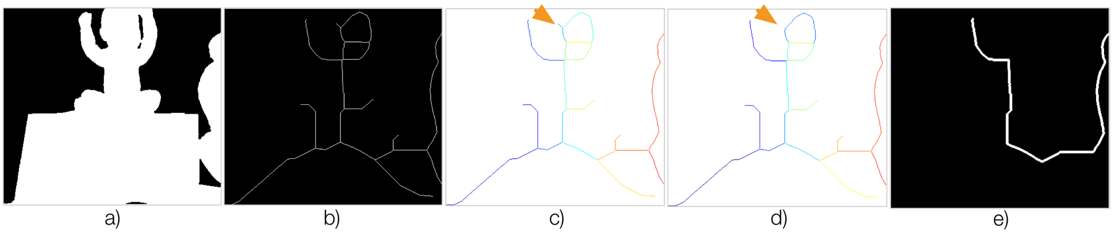


Figure 6.4: Illustration of automatic generation of line scribbles. *a)* Disparity region for which we want to generate a line scribble (*white*). *b)* Minimally connected stroke (*white*) obtained by thinning. *c)* Segmentation of this stroke into line segments, i.e., branches (*color-coded*), which are subsequently removed in ascending order of their length. *d)* The shortest branch was removed (*orange arrow*). This process is repeated until the minimally connected stroke is branch-free. *e)* Final line scribble (*white*), i.e., dilated branch-free minimally connected stroke.

of an object and are placed approximately in its middle (Figure 6.3). While, to the best of our knowledge, the reasons for the frequent choice of this scribbling strategy are not discussed in the literature, we suppose it is also motivated by its simplicity. As discussed above, placing scribbles in the middle of objects is less labor-intensive than near their borders. Contrary to PS, LS cover more pixels that contain potentially more colors and provide more spatial support. Regardless of our supposed reasons, perhaps this scribbling strategy simply feels more natural to users than other strategies. The frequent choice of this scribbling strategy makes it worth to integrate it in our evaluation study. To this end, we first apply a thinning algorithm [80] (Figure 6.4 *a)* and *b)*) and then iteratively remove branches of the obtained minimally connected stroke in ascending order of their length (Figure 6.4 *c)* and *d)*). Finally, the branch-free stroke is dilated. As a result (Figure 6.4 *e)* and Figure 6.2 *f)*), we obtain similar scribbles as observed in the literature (Figure 6.3).

Expert scribbles (ES) are closely related to LS. While LS cover more pixels than PS, they still not necessarily cover differently colored pixels within disparity regions. More experienced users, i.e., expert users, might draw scribbles deliberately across differently colored pixels within a region. In fact, we observed this strategy in the literature. For instance, in Figure 6.3 *a)*, the cartoon characters are annotated with "LS" that have additional branches to areas with different colors (e.g., scribble on red costume with branch to black glove). Based on this observation, we extend LS to ES by adding similar branches. Specifically, for each LS (Figure 6.5 *a)*), we perform a rudimentary color segmentation by comparing

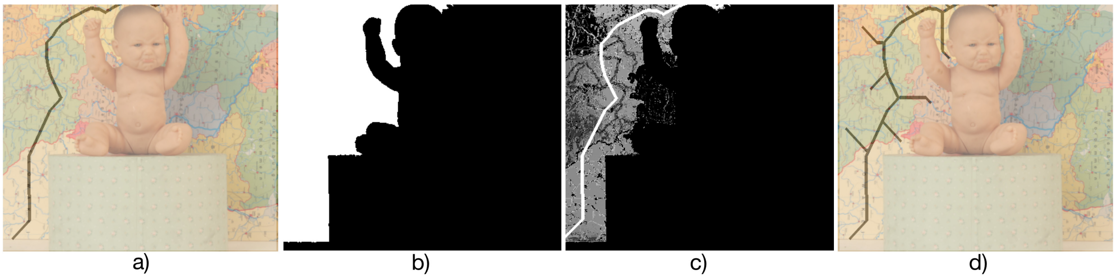


Figure 6.5: Illustration of automatic generation of expert scribbles. *a)* Input image with a single line scribble (*black*). *b)* Disparity region (*white*) for which the scribble was generated. *c)* Rudimentary color segmentation based on the line scribble. Pixels within the disparity region with colors covered by the line scribble are *white*, remaining pixels are *black*. *d)* The expert scribble (*black*) covers more colors than the line scribble in *a)*.

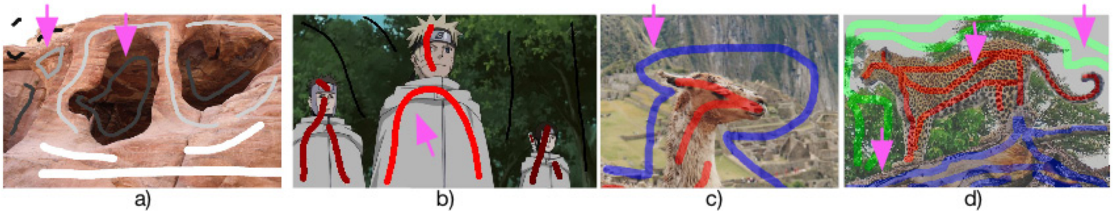


Figure 6.6: Border scribbles in the literature. Independent of the color-coding and meaning of the scribbles, user-centric systems, such as *a)* [163], *b)* [117], *c)* [125] and *d)* [79], are often initialized with scribbles that are placed near object borders (*pink arrows*).

the color model obtained from pixels covered by the scribble with the color model obtained from the remaining pixels in the same disparity region (Figure 6.5 *b)*). This comparison seperates pixels with colors covered by the scribble from pixels with colors not covered by the scribble (Figure 6.5 *c)*). To obtain our final ES, we add branches that start in areas of the latter pixels and follow the shortest path to the scribble. When comparing LS (Figure 6.5 *a)* and Figure 6.2 *f)*) with corresponding ES (Figure 6.5 *d)* and Figure 6.2 *g)*), it is evident that ES cover a larger variety of colors than LS.

Border scribbles (BS) are used in our (for users) most labor-intensive scribbling strategy. With BS, users roughly trace object borders using scribbles, which often resemble (closed) circular lines. This strategy can be observed in the literature (e.g., Figure 6.6, *pink arrows*). BS are more labor-intensive than the previous discussed scribbling strategies concerning two aspects: (1) the larger number of marked pixels and (2) the closeness of the scribbles to object borders. However, no additional efforts to cover different colors are made. A unique advantage of BS over PS, LS and ES is that BS further constrain the solution of conversion systems close to object borders and, thus, potentially increase the accuracy of the results near them. We simulate this scribbling strategy by shifting the borders of the disparity regions closer to their center using morphological operators. As shown in Figure 6.2 *h)*, the resulting BS are closed circular lines that follow region borders.

6.2.3 Investigated Semi-automatic 2D-to-3D Conversion Systems

Our evaluations focus on semi-automatic 2D-to-3D conversion systems that are based on scribble-based user annotations. In particular, we investigate five systems: (1) our segmentation-based conversion (*GS+P*, Chapter 4), our CVF-based conversion with the assignment schemes (2) DB (*CVF (DB)*, Chapter 5) and (3) WTA (*CVF (WTA)*, Chapter 5), (4) our implementation of Guttmann et al.'s algorithm [56] (*LSE*) and (5) the conversion system by Phan et al. [118] (*GC+RW*). They were discussed in detail in Chapter 4, Chapter 5 and Section 2.2. It is clear from previous chapters, that these systems specifically, and semi-automatic 2D-to-3D conversion systems in general, follow two basic steps: (i) the annotation of 2D content with sparse disparity (or depth) information and (ii) the propagation of this information to the remaining pixels. Below, we briefly recap the tested systems and discuss their most important similarities and differences, as well as aspects that should be noted in the context of our evaluation study.

The first step in semi-automatic 2D-to-3D conversion is the annotation of 2D content. While the tested conversion systems are all based on scribble-based user annotations that directly encode the disparities of the marked pixels, the encoding of their scribbles varies. Specifically, while GS+P and LSE support scribbles that contain multiple disparities, this is not the case for CVF (WTA), CVF (DB) and GC+RW. In our evaluations, we equal the former systems to the latter systems by also using a single disparity per scribble. As mentioned above, all tested systems are initialized with GT-derived disparities at scribble positions instead of relying on the respective encoding of the scribbles. Hence, in our study, the tested systems are all based on the exact same user input. A mutuality concerning the scribble-based annotations for all five semi-automatic 2D-to-3D conversion systems is that annotated 2D content is processed after all scribbles are drawn[4]. Thus, our evaluation study abstains from simulating an interactive user who follows a progressive annotation strategy (such as in [76]). Contrary, we focus on the effect of different scribble strategies in the final 2D-to-3D conversion results.

The second step and principal part of semi-automatic 2D-to-3D conversion is the propagation of scribble disparities to the remaining pixels. Since our evaluations are performed on a dataset of images (as opposed to videos), our study uses image versions of the 2D-to-3D conversion systems mentioned above. For GS+P[5], this means that the propagation is performed jointly with a graph-based segmentation of a single image [43, 53] instead of a video segmentation that additionally connects pixels across frames. Note that the image version also includes the region merging step [53] that uses richer similarity measures than local per-pixel similarities as in [43]. During this segmentation spatially connected and similar (color) pixels are merged to segments following local, greedy decisions (that satisfy global properties [43,53]) and disparities are propagated. The subsequent interpolation and refinement steps are performed on a single image and not across multiple frames of a video. Thus, in the image version their main task is not temporal disparity interpolation, but the interpolation of disparity within segments. For GS+P we use the following parameters: segmentation parameters $\{\tau = 0.3, T_{minsize} = 5,$ 7 iterations$\}$, segment-wise filter parameters $\{r_s = 9, \epsilon = 0.0001\}$ and filter parameters for refinement $\{r_s = 3, \epsilon = 0.0001\}$. The color histograms have 20 bins per color channel (LAB).

---

[4] Users still have the option to change their annotations and update the previously obtained conversion results. This, however, involves the recomputation of the entire solution.

[5] We use the un-optimized C++ implementation of the segmentation-based propagation algorithm in Chapter 4.

While GS+P is based on an automatic segmentation algorithm, CVF (WTA) and CVF (DB) are an extension of an interactive object segmentation algorithm (Chapter 5). GS+P directly propagates disparities between immediately connected (groups of) pixels with similar appearance (i.e., color). It neither explicitly models the appearance of specific pre-defined labels (i.e., disparities) nor considers the similarity of pixels to them. Contrarily, our CVF-based algorithms view propagation as a label-based optimization problem, in which the appearance of a scribble's pixels (i.e., set of all pixels marked by a specific scribble) explicitly models its given disparity. They then compute (color) similarities, between individual pixels and these color models, to obtain probabilities for the models' associated disparities. A local filtering technique aggregates probabilities to approximately solve the label-based optimization problem [125]. This approach can bypass propagations across large distances. The image version computes probabilities based on color similarities to multiple user-drawn scribbles (i.e., MS), spatial closeness to these scribbles (i.e., STC) and employs a 3D connectivity constraint (i.e., CON). Since our evaluation study focuses on 2D images, motion related components of the algorithm, such as temporal disparity interpolations (i.e., DC -tM and -sM), motion guided filtering (i.e., +TC) and motion segmentation-based scribble matching and grouping are disabled. Throughout this evaluation study, we use the following parameter set: CVF parameters $\{r_s = 10, \epsilon = 0.0016\}$, DB parameter $n = 2$, STC parameter t_{close} is automatically set to the maximal possible spatial distance in each image. The color histograms have 32 bins per color channel (RGB).

Phan et al. [118] provided us with an image version of their 2D-to-3D conversion system (GC+RW). Like CVF (WTA) and CVF (DB), it is based on an interactive object segmentation algorithm, i.e., [19]. In [118], the segmentation result is, as an additional constraint, incorporated into a global edge-aware interpolation [51] to avoid over-smoothing of edges in the conversion results. We use the constants and parameters that were suggested by the authors of [118].

Our implementation of Guttmann et al.'s 2D-to-3D conversion system [56] (LSE) expresses the propagation in terms of a continuous global function that is solved by a standard minimization technique [84]. In some similarity to GS+P, disparities are propagated between neighboring pixels without explicitly considering global color models as in CVF (WTA), CVF (DB) or GC+RW. Unlike the system's original video version, its image version only employs spatial smoothness constraints that are based on color similarity of neighboring pixels and disables all motion related components. In our evaluation study, we use the parameters suggested in [56].

In summary, all tested systems implement the same key assumption, i.e., that pixels with similar colors are likely to have similar disparities as opposed to pixels with strong color contrast. While CVF (WTA), CVF (DB) and LSE compute these color similarities in the RGB color space, GS+P and GC+RW use the LAB color space. CVF (WTA) and CVF (DB) determine color similarities using color histograms (i.e., color models). GC+RW, GS+P and LSE express color similarly (additionally) in terms of absolute color differences between individual pixels. Furthermore, all tested systems assume that spatially close pixels are more likely to have a similar disparity than spatially distant pixels, whereas spatial closeness is differently realized by the tested systems. GS+P, LSE and GC+RW propagate disparities only between spatially connected immediate neighbor pixels. Thus, these systems indirectly enforce that only pixels that are spatially connected to a specific scribble are influenced by its disparity. Contrary, CVF (WTA) and CVF (DB) directly reduce the probability of assigning spatially distant pixels (defined by

Table 6.1: Quantitative evaluation of conversion accuracy. The tested 2D-to-3D conversion system are used with different scribbling strategies. The MSEs are computed by comparing the conversion results with their reference solutions and are multiplied by 100. *Red* superscripts are average ranks with respect to the conversion systems. *Blue* subscripts are average ranks with respect to the scribbling strategies. Low ranks and MSEs indicate more accurate results than large ranks and MSEs. Errors for images from [134] and from [27] are listed separately.

$\mathbf{MSE}^{\text{Rank (systems)}}_{\text{Rank (scribbles)}}$	images from [134]				images from [27]			
	BS	**ES**	**LS**	**PS**	**BS**	**ES**	**LS**	**PS**
GS+P	$0.01^{1.0}_{1.1}$	$0.02^{1.5}_{2.0}$	$0.04^{2.2}_{2.9}$	$0.12^{2.9}_{4.0}$	$0.06^{1.5}_{1.0}$	$0.15^{1.5}_{2.0}$	$0.40^{2.3}_{2.8}$	$10.31^{3.8}_{4.0}$
GC+RW [118]	$0.01^{2.8}_{1.1}$	$0.04^{2.9}_{2.0}$	$0.05^{2.8}_{2.9}$	$0.15^{3.2}_{4.0}$	$0.09^{2.8}_{1.0}$	$0.23^{2.8}_{2.0}$	$0.25^{2.8}_{3.0}$	$8.05^{2.8}_{4.0}$
CVF (WTA)	$0.02^{3.4}_{1.2}$	$0.03^{2.9}_{1.8}$	$0.05^{2.8}_{3.0}$	$0.18^{3.5}_{4.0}$	$0.20^{2.5}_{1.3}$	$0.48^{3.0}_{1.8}$	$0.53^{2.8}_{3.0}$	$4.22^{2.0}_{4.0}$
CVF (DB)	$0.04^{4.7}_{1.6}$	$0.05^{3.9}_{1.7}$	$0.06^{3.3}_{2.7}$	$0.18^{2.9}_{4.0}$	$1.15^{3.8}_{1.5}$	$1.17^{3.3}_{2.0}$	$1.14^{2.8}_{2.5}$	$4.76^{2.0}_{4.0}$
LSE [56]	$0.01^{3.1}_{1.0}$	$0.04^{3.9}_{2.0}$	$0.06^{3.9}_{3.0}$	$0.13^{2.5}_{4.0}$	$0.39^{4.5}_{1.0}$	$2.14^{4.5}_{2.0}$	$3.22^{4.5}_{3.0}$	$10.70^{4.5}_{4.0}$

a spatial distance transform [17] using Chessboard distance) to a disparity of a scribble by employing a spatial closeness (i.e., STC) and a connectivity (i.e., CON) constraint. Compared to GS+P, LSE and GC+RW, their connectivity constraint is more general, i.e., it allows spatial connections in the background (taking occlusions into account). As mentioned above, GS+P and LSE do not model explicitly the appearance of each scribble, whereas in CVF (WTA), CVF (DB) and GC+RW related color models are used. The systems further differ in their performed smoothing. While CVF (WTA) refrains from smoothing, CVF (DB) blends the two most probable disparities for each pixel. LSE and GC+RW globally smooth neighboring disparities with similar colors, whereas GC+RW additionally relies on segmentation to reduce over-smoothing. GS+P takes on a similar approach by locally smoothing disparities within segments.

6.3 Experimental Results and Evaluation

We compare the 2D-to-3D conversion results for 2D images of our dataset (Section 6.2.1) with their reference solutions. The results were obtained by the tested 2D-to-3D conversion systems (Section 6.2.3), using different scribbling strategies (Section 6.2.2). In the first experiment (Section 6.3.1), we compare the accuracy of results from different systems and scribbling strategies. The second experiment (Section 6.3.2) focuses on their robustness to small scribble perturbations. In our third experiment (Section 6.3.3) the tolerance to inaccuracies in scribble annotations is investigated. The fourth experiment (Section 6.3.4) is an in-depth series of evaluations that considers the content of the 2D images. More precisely, for each system, the ideal scribbling strategy for different types of regions (e.g., multicolored or homogenous) is investigated.

6.3.1 Conversion Accuracy

In our first experiment, we compare the results obtained by the tested 2D-to-3D conversion systems using different scribbling strategies. Specifically, each 2D image from our dataset is

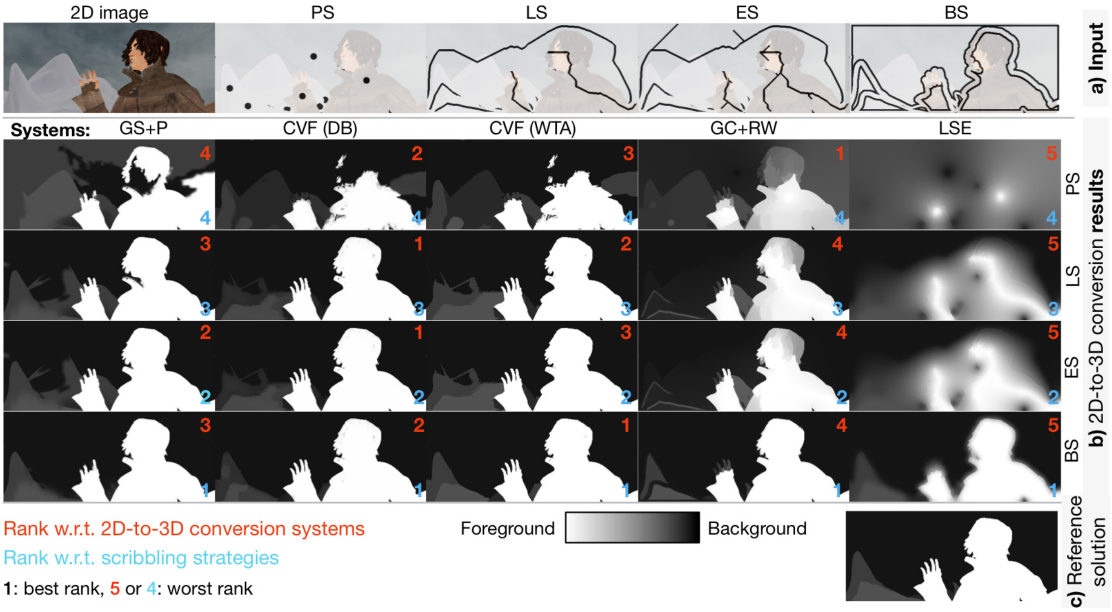


Figure 6.7: Visual evaluation of conversion results. *a)* Input image (*Ambush2* from [27]) with scribbles (*black*) and *c)* reference solution. *b)* Results of the tested 2D-to-3D conversion systems when using different scribbling strategies (BS, ES, LS, PS). *Red* numbers are the ranks of the tested 2D-to-3D conversion systems. *Blue* numbers are the ranks of the scribbling strategies.

automatically annotated exclusively using the scribbling strategies[6] PS, LS, ES and BS and processed by five tested conversion systems, i.e., GS+P, GC+RW [118], CVF (WTA), CVF (DB) and LSE [56]. We compare their results with their reference solutions and show the resulting MSEs in Table 6.1 and Figure 6.7. Table 6.1 further lists the average ranks. The *red* superscripts are average ranks ($\in [1, 5]$) with respect to the conversion systems (per scribbling strategy). The *blue* subscripts are average ranks ($\in [1, 4]$) of the scribbling strategies (per system).

Let us first focus on the scribbling strategies. When comparing the errors in the rows of Table 6.1, BS is preferred by the tested systems. Two factors explain BS' good performances. BS have a larger amount of marked pixels and are especially favorable to conversion systems that employ strict connectivity constraints. In particular, BS can for GS+P, CG+RW and LSE act as a "barrier" that avoids erroneous assignments within areas that are enclosed by BS. In our CVF-based systems, which support connectivity paths that continue in the background, scribbles have a potentially larger area of influence. For them, BS only reduces errors within areas enclosed by BS. When averaging the scribbles' ranks in Table 6.1 (*blue*), BS are followed by ES, LS and PS. ES cover differently colored pixels and are spatially larger than LS and PS. ES typically yield plausible conversions, which, however, might exhibit errors near borders between regions with similar colors (Figure 6.7 *b)*, snow). In case of LS and PS, errors are often related to object colors that were not covered by scribbles (Figure 6.7 *b)*, PS, head) or their color ambiguities.

---

[6] First versions of PS and LS (i.e. DS and TS) were used in Section 4.4 to annotate and convert 2D images to 3D using GS+P. Then, the influence of GS+P's region merging step on the conversion quality was evaluated.

Table 6.1 also compares the 2D-to-3D conversion systems per scribbling strategy (i.e., comparison of errors in columns). The errors of a system vary when using different scribbling strategies. Furthermore, the systems' rankings differ between images from [134] and from [27] (e.g., PS). Nonetheless, overall our evaluations indicate that the tested conversion systems that use segmentation algorithms (GS+P, GC+RW, CVF (WTA) and CVF (DB)) perform better than a system that solely relies on edge-aware interpolation (LSE). Our segmentation-based conversion from Chapter 4, i.e., GS+P, performs best for most scribbling strategies (BS, ES and LS). When averaging the systems' ranks in Table 6.1 (*red*), GS+P is followed by GC+RW, CVF (WTA), CVF (DB) and finally LSE. In this context, the second and third best performing conversion systems, i.e., GC+RW and CVF (WTA), build upon interactive segmentation algorithms. While CVF (WTA) focuses on direct assignments of scribble disparities to pixels, GC+RW additionally smoothes them using edge-aware interpolation. Thus, contrary to CVF (WTA), GC+RW's results contain disparity falloffs which are desired for rounded, but undesired for fronto-parallel objects (e.g., Figure 6.7 *b)*, head). The two worst performing conversion systems are CVF (DB) and LSE. CVF (DB)'s disparity blending approach (i.e., DB) averages the two most probable disparities per pixel. This results in (small) quantitative errors at each pixel (which add up to large MSEs and leads to large average ranks) and a sensitivity to overlapping color models of scribbles. Being based on a segmentation algorithm, CVF (DB) captures disparity edges at object borders (e.g., Figure 6.7 *b)*). This is not the case for LSE, which generally and especially for sparser scribble inputs (PS) over-smoothes its results (e.g., Figure 6.7 *b)*). In this context, it should be noted that images from [27] (as opposed to images from [134]) contain scenes with large foreground objects on distant backgrounds and typically fewer scribbles.

Overall, this first experiment leads to three important observations concerning the conversion accuracy: **(1)** We have confirmed that scribbles varying in length and placement affect the results of the tested 2D-to-3D conversion systems and, thus, should be investigated further. Generally, it pays off to put more effort into the annotation process, i.e., by covering different colors or drawing larger scribbles (using ES or BS). Especially systems that employ a stricter spatial connectivity constraint (GS+P, LSE and GC+RW) as opposed to a more general 3D connectivity constraint (CVF (DB) and CVF (WTA)) have shown to exploit the additional spatial support of our most laborious scribbling strategy (BS). **(2)** The performance of the tested systems not only depends on the placement of the used scribbles, but also on the 2D content. Different images (areas) might call for different scribbling strategies, while for other images (areas) the scribbling strategy might not matter. In the following experiments, this aspect will be further investigated. **(3)** Our segmentation-based algorithm (GS+P) clearly outperforms the 2D-to-3D conversion by solely global edge-aware interpolation (LSE). Furthermore, we have often observed better performances for systems that integrate segmentation algorithms in their conversion process (GS+P, GC+RW, CVF (WTA) and CVF (DB)) as opposed to the system that solely relies on edge-aware interpolation and does not incorporate a segmentation algorithm in its conversion process (LSE).

6.3.2 Robustness to Scribble Perturbations

In our second experiment, we address the robustness of the tested systems (Section 6.2.3) to small scribble perturbations for each scribbling strategy (Section 6.2.2). To this end, we automatically generate two additional sets of annotations per scribbling strategy. To obtain the first set of

Table 6.2: Quantitative evaluation of robustness to scribble perturbations (scribble comparison). Performance changes between systems' results from scribbles' original positions (i.e., 0) and each of their shifted positions (i.e., by 10 and 20 pixels) are compared. The table ranks scribbling strategies with respect to these changes per 2D-to-3D conversion system (comparing scribbles column-wise and separately within $0-10$ and within $0-20$). Low ranks indicate robuster scribbles than large ranks. Ranks for images from [134] and from [27] are listed separately.

Avg. Ranks	images from [134]					images from [27]				
(scribble comparison)	**CVF (DB)**	**GS +P**	**CVF (WTA)**	**LSE**	**GC+ RW**	**CVF (DB)**	**GS +P**	**CVF (WTA)**	**LSE**	**GC+ RW**
BS $0-10$	2.7	1.9	1.9	2.7	2.7	3.5	2.0	2.5	2.8	1.0
ES $0-10$	1.7	2.6	2.1	2.3	1.9	2.3	2.5	2.3	2.3	2.8
LS $0-10$	2.1	3.1	2.4	2.6	2.5	2.0	2.8	1.5	3.0	2.8
PS $0-10$	3.5	2.4	3.5	2.4	2.9	2.3	2.5	3.8	2.0	3.5
BS $0-20$	2.7	2.1	2.3	3.6	3.1	2.5	2.3	2.0	3.5	1.3
ES $0-20$	2.2	2.9	2.9	2.5	1.7	2.8	2.3	2.0	2.3	3.0
LS $0-20$	2.2	2.1	1.9	2.1	2.3	1.3	2.5	2.0	2.0	2.5
PS $0-20$	2.9	2.8	2.9	1.8	2.9	3.5	3.0	4.0	2.3	3.3

Table 6.3: Quantitative evaluation of robustness to scribble perturbations (system comparison). Performance changes between systems' results from scribbles' original positions (i.e., 0) and each of their shifted positions (i.e., by 10 and 20 pixels) are compared. The table ranks 2D-to-3D conversion systems with respect to these changes per scribbling strategy (comparing the systems row-wise and separately within $0-10$ and within $0-20$). Low ranks indicate robuster systems than large ranks. The ranks for images from [134] and from [27] are listed separately.

Avg. Ranks	images from [134]					images from [27]				
(system comparison)	**CVF (DB)**	**GS +P**	**CVF (WTA)**	**LSE**	**GC+ RW**	**CVF (DB)**	**GS +P**	**CVF (WTA)**	**LSE**	**GC+ RW**
BS $0-10$	4.4	3.2	2.5	2.3	2.5	4.0	3.0	3.0	3.8	1.3
ES $0-10$	3.5	4.1	3.1	2.2	2.1	4.0	3.0	2.8	3.3	2.0
LS $0-10$	3.2	4.0	3.3	2.3	2.2	3.5	3.8	2.5	3.5	1.8
PS $0-10$	3.9	2.7	3.9	2.1	2.5	2.5	2.5	3.0	3.0	4.0
BS $0-20$	3.0	3.3	3.1	3.3	2.3	3.8	2.8	2.5	4.3	1.8
ES $0-20$	3.0	3.9	3.7	2.8	1.6	3.3	2.5	2.3	3.8	3.3
LS $0-20$	3.1	3.9	3.2	2.6	2.2	3.0	3.8	2.5	3.0	2.8
PS $0-20$	3.2	3.4	3.8	2.1	2.5	2.8	3.8	2.8	2.5	3.3

annotations, we shift each scribble within a region (Section 6.2.2) by maximal 10 pixels (i.e., five to the right and five to the bottom) from their original positions. For the second set a shift by maximal 20 pixels (i.e., 10 to the right and 10 to the bottom) from their original positions is performed (e.g., Figure 6.8 *a)*). When performing these position shifts, we maintain the disparity, connectivity and the number of pixels of each scribble. The two additional sets of annotations are subsequently used to convert the images in our dataset (Section 6.2.1) with the tested 2D-to-3D

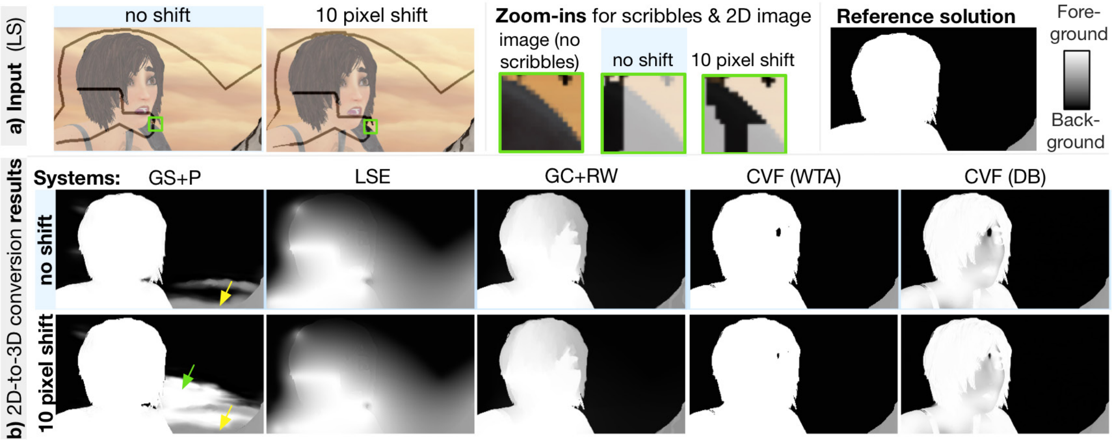


Figure 6.8: Example results after scribble perturbations (*Temple3* from [27]). *a)* Input image with LS shifted by 0 and 10 pixels, zoom-ins (*green*) that show *black* LS on transparent image, and reference solution. *b)* 2D-to-3D conversion results from GS+P, LSE [56], GC+RW [118], CVF (WTA) and CVF (DB). GS+P's results exhibit the visually largest change. *Green arrow*: A perturbation-caused annotation of mixed pixels (*zoom-ins*) leads to propagation of foreground disparity into the background. *Yellow arrows*: Similar error with initial scribbles (*no shift*).

conversion systems. Table 6.2 and Table 6.3 focus on the change that was observed between these results. Specifically, Table 6.2 lists average ranks that compare the scribbling strategies and were computed from the MSE changes (i.e., absolute differences) between the results generated from original and shifted scribble inputs. These ranks ($\in [1, 4]$) are computed within each set of shifted annotations (comparing the scribbling strategies separately within $0 - 10$ and within $0 - 20$) and for each system. Table 6.3 gives an analogous ranking ($\in [1, 5]$) with respect to the 2D-to-3D conversion systems. Below, we first discuss the ranking of the scribbling strategies and then the ranking of the conversion systems.

In Table 6.2, PS has large average ranks (i.e., is not robust to perturbations) for most systems (CVF (DB), GS+P, CVF (WTA) and GC+RW). However, for LSE, PS' severely over-smoothed results are very similar to each other, leading to PS' low average ranks with LSE. In the case of GS+P, BS obtain the lowest average ranks with images from ([134] and [27]), which indicates that GS+P is most robust to scribble perturbations when using BS. In most cases, the performances of the scribbling strategies in Table 6.2 vary across conversion systems and datasets. Furthermore, in some cases the scribbles' average ranks within a dataset are very similar to each other (e.g., LSE, images from [134], PS $0 - 10$: $\in [2.3, 2.7]$). For these reasons, we could not identify a single scribbling strategy that is throughout most robust to scribble perturbations.

Table 6.3 compares the robustness of the tested 2D-to-3D conversion systems to scribble perturbations (i.e., comparison of ranks in rows). When averaging these ranks over all scribbling strategies, CVF (DB) is most sensible to scribble perturbations. It is followed by GS+P, CVF (WTA), LSE and finally GC+RW, the on average most robust system. Thus, in this evaluation conversion systems that perform global interpolations (LSE and GC+RW) are more robust to scribble perturbations than systems that are based on local decisions or smooth locally (GS+P, CVF (WTA) and CVF (DB)). As shown in Figure 6.8 *b)*, for LSE and GC+RW scribble perturbations visually manifest as subtle changes of disparity falloffs in their results. Note that in the case of LSE, this is related to its (severely) over-smoothed conversion results. In this context, it is worth mentioning that LSE is less robust for scenes (from [27]) that contain large foreground objects on distant backgrounds than for cluttered scenes (from [134]) with lower disparity differences between objects. In the former scenes, differences between LSE's over-smoothed results carry quantitatively more weight. GS+P, CVF (WTA) and CVF (DB) are, in most cases (note the mentioned exception of LSE and images from [27]), less robust than LSE and GC+RW, and can be considered more local than those systems in certain aspects. Shifting scribble positions involves the additional coverage or exclusion of object colors by scribbles. For our CVF-based systems such changes affect the scribbles' color models and might alter local disparity assignments in their results. Since CVF (DB)'s local DB approach is sensitive to changes in these models, CVF (DB)'s conversion results typically change after shifting the scribbles. GS+P's sensitivity to scribble perturbations comes from the greedy nature of its segmentation algorithm [53], which is initially based on color similarities between individual pixels. For GS+P shifted scribbles at object borders and colors that are (locally) not covered by scribbles after their perturbation might lead to erroneous assignments of large areas (Figure 6.8 *b)*, *green arrow*). This can be avoided by using BS, which increase the robustness of GS+P (Table 6.2).

Overall, in this second experiment two main observations concerning the robustness to small perturbations of the scribble input were made: **(1)** The robustness to scribble perturbations varied across 2D-to-3D conversion systems and datasets. This indicates the choice of scribbling strategy alone is not a major dependency for a system's robustness. Analogously, no single conversion system was throughout most robust. **(2)** The amount and type of smoothing emerged as a crucial factor in the quantitative robustness-evaluation. For systems that perform smooth global interpolations (LSE and GC+RW), scribble perturbations manifested in changes of disparity falloffs. For systems that are based on local decisions or smooth locally (GS+P, CVF (WTA) and CVF (DB)), large areas might be (abruptly) assigned to different disparities than before the scribble perturbation. The latter changes were visually and quantitatively more noticeable.

6.3.3 Error Tolerance

In our third experiment, we focus on the tolerance against inaccurate scribble annotations. In practice user annotations are not necessarily accurate, e.g., foreground scribbles might accidentally cover some pixels that should be assigned to background disparities. We simulate inaccurate user annotations by adding 10, 20 and 50 pixels that actually should be assigned to a different disparity to each scribble. In order to maintain the connectivity of a distorted scribble, we first add a minimal connected branch (i.e., width of one pixel) that connects the original scribble in a region (Section 6.2.2) to the closest pixel on the region border (e.g., Figure 6.9 *a)*, *zoom-ins*). The set of altered scribbles, i.e., with the additional branch but without erroneous assignments, serves as baseline in this experiment. Subsequently, we generate three additional sets of annotations per scribbling strategy by adding 10, 20 and 50 pixels from neighboring regions (i.e., erroneous assignments) to these branches. By removing the same amount of correctly assigned pixels as were added, the number of pixels per scribble is maintained. In summary, we obtain four sets of annotations per scribbling strategy, i.e., the set of baseline scribbles and three sets of

Table 6.4: Quantitative evaluation of error tolerance. This table compares performance changes between the systems' results obtained with undistorted (i.e., 0) scribbles and their distorted versions with 10, 20 and 50 erroneous pixels. Systems and scribbles are ranked according to these changes (within each set of distorted annotations). Average (*avg.*) ranks of the scribbling strategies (*left*, comparison of ranks in columns) and of the systems (*right*, comparison of ranks in rows) are shown. Low avg. ranks indicate more error-tolerant results than large avg. ranks.

	Avg. Ranks					
		GS+P	**CVF (DB)**	**GC+ RW**	**CVF WTA**	**LSE**
scribble comparison	**BS** 0 − 10	1.4	1.8	1.4	1.4	1.3
	ES 0 − 10	2.5	2.4	2.2	2.2	2.5
	LS 0 − 10	2.8	2.2	2.4	2.6	2.5
	PS 0 − 10	3.3	3.6	4.0	3.7	3.7
	BS 0 − 20	1.4	1.5	1.4	1.4	1.3
	ES 0 − 20	2.1	2.4	2.2	2.1	2.5
	LS 0 − 20	3.1	2.2	2.4	2.7	2.7
	PS 0 − 20	3.4	3.9	4.0	3.9	3.5
	BS 0 − 50	1.2	1.6	1.2	1.3	1.2
	ES 0 − 50	2.1	2.1	2.2	2.2	2.1
	LS 0 − 50	3.1	2.5	2.6	2.6	2.6
	PS 0 − 50	3.4	3.7	4.0	3.9	4.0

Avg. Ranks	system comparison				
	GS+P	**CVF (DB)**	**GC+ RW**	**CVF WTA**	**LSE**
BS 0 − 10	3.4	3.4	2.7	2.9	2.6
ES 0 − 10	4.2	3.1	3.0	2.2	2.4
LS 0 − 10	4.2	2.8	3.0	2.7	2.3
PS 0 − 10	3.6	3.3	3.2	2.9	2.0
BS 0 − 20	3.2	3.9	2.9	2.9	2.1
ES 0 − 20	4.2	3.5	3.0	2.1	2.2
LS 0 − 20	4.3	2.9	2.7	2.9	2.2
PS 0 − 20	3.6	3.3	3.6	3.1	1.4
BS 0 − 50	3.1	3.6	3.6	2.6	2.1
ES 0 − 50	4.1	3.0	3.1	2.4	2.4
LS 0 − 50	4.3	2.9	3.3	2.6	1.9
PS 0 − 50	3.3	2.9	3.6	3.1	2.2

distorted scribbles with 10, 20 and 50 erroneously annotated pixels per scribble. These 16 sets of input scribbles per image are processed by the five tested 2D-to-3D conversion systems. To reduce the amount of performed conversions (i.e., 80 per image), we use a smaller version of our dataset which contains only 14 images (i.e., 11 images from [134] and three from [27]) instead of originally 19 images (Figure 6.1).[7] Table 6.4 lists average ranks that were computed from the MSE changes (i.e., absolut differences) that are observed when comparing results obtained from the baseline scribbles with results obtained from the distorted scribbles.

In Table 6.4 (*left*, comparison of ranks in columns per distortion), BS exhibit the lowest ranks and are followed by ES, LS and finally PS. While BS clearly emerge as the most error-tolerant scribbling strategy, PS is the most sensitive one. This observation can be explained by PS' small amount of annotated pixels and, thus, their smaller ratio of correctly to erroneously annotated pixels per scribble. Besides the larger amount of annotated pixels, BS have an additional advantage over the remaining scribbling strategies. As mentioned in Section 6.3.1, the systems' connectivity constraints avoid (GS+P, GC+RW and LSE) or reduce (CVF (WTA) and CVF (DB)) erroneous propagations into BS' enclosed areas. Thus, for BS errors are more or less constrained to pixels between the areas enclosed by BS, while scribbling strategies that are closer to regions' centers (e.g., PS) leave larger areas in which erroneous propagations might spread.

---

[7] The five images that were removed from our full dataset only had a small contribution in terms of content diversity to the full version of the dataset (Figure 6.1). For instance, in case of the removed images *Barn2*, *Poster* and *Venus*, a very similar image, i.e., *Barn1*, is still contained in the smaller version of our dataset.

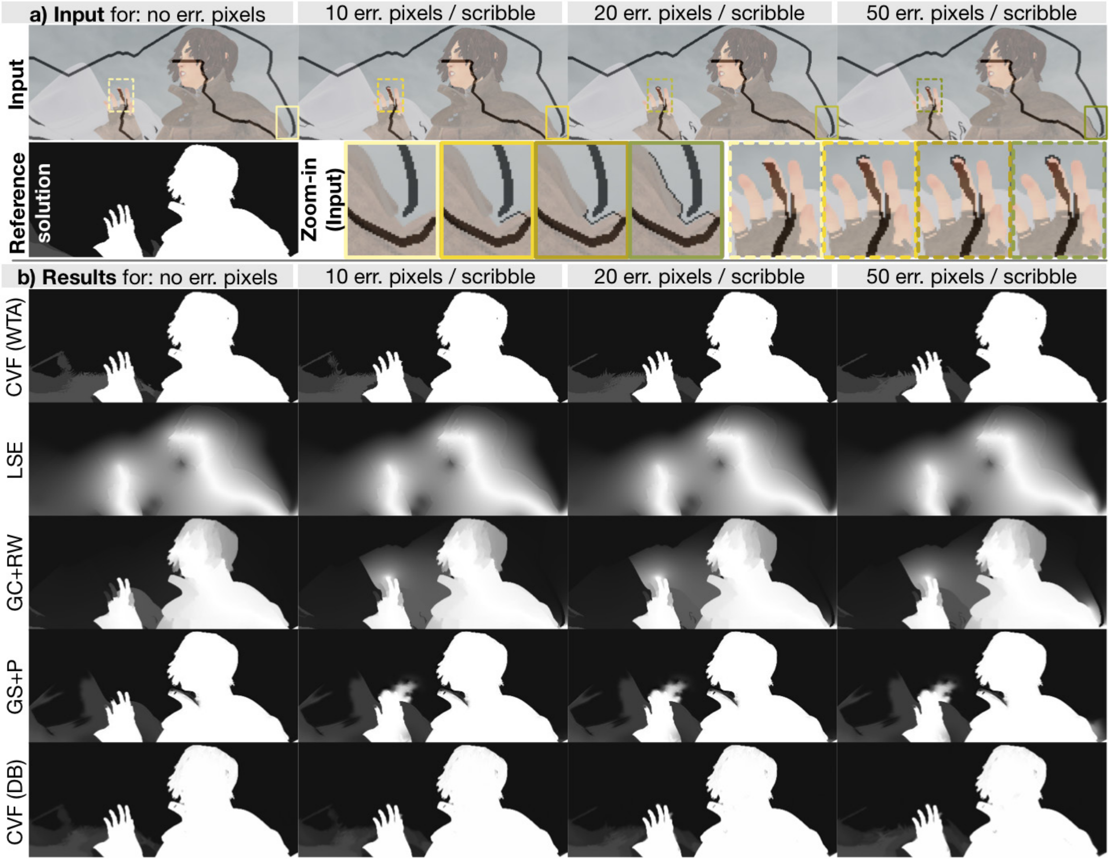

a) Input for: no err. pixels
10 err. pixels / scribble
20 err. pixels / scribble
50 err. pixels / scribble
Input
Reference solution
Zoom-in (Input)
b) Results for: no err. pixels
10 err. pixels / scribble
20 err. pixels / scribble
50 err. pixels / scribble
CVF (WTA)
LSE
GC+RW
GS+P
CVF (DB)

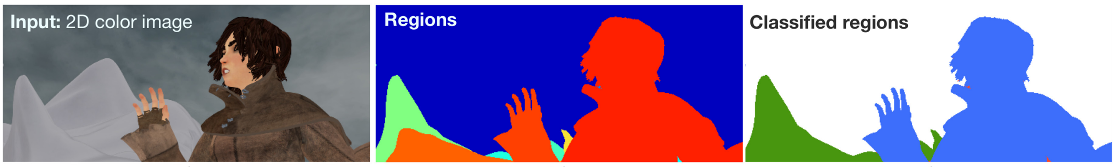
Figure 6.9: Visual evaluation of error tolerance (*Ambush2* from [27]). *a)* Input image with LS and its reference solution. Foreground: *bright*, background: *dark*. LS are distorted by 0, 10, 20 and 50 erroneously (*err.*) annotated pixels per scribble. *Zoom-ins* on the input provide a detailed view on the (distorted) scribbles for two image areas, which are marked with *solid* and *dotted lines*, respectively. *b)* 2D-to-3D conversion results from CVF (WTA), LSE [56], GC+RW [118], GS+P and CVF (DB). Note the distortion-caused errors close to the person's hand.

Table 6.4 (*right*) ranks the tested 2D-to-3D conversion systems according to their distortion-caused changes (comparison of ranks in rows). When averaging Table 6.4 (*right*) over scribbling strategies, LSE is the most error-tolerant system among the tested ones. As in Section 6.3.2, LSE's low ranks in Table 6.4 (*right*) are related to its generally over-smoothed conversion results. Specifically, the introduced distortions disappear in LSE's already inaccurate results (Figure 6.9). LSE is followed by CVF (WTA), GC+RW, CVF (DB) and finally GS+P. Thus, among the remaining tested conversion systems, systems that use color models (CVF (WTA), GC+RW and CVF (DB)) are more error-tolerant than a system that propagates the disparity from individual annotated pixels (GS+P). The mentioned color models explicitly model the disparity of each scribble by the appearance (color) of all its scribble's pixels (i.e., set of all pixels marked by a specific scribble). With the support of a set of pixels, color models provide additional robustness against inaccurate annotations compared to propagations from single pixels (as in GS+P). The results are further influenced by the systems' smoothing components. In some similarity to Section 6.3.2, CVF (DB)'s DB approach is sensitive to inaccurate annotations. With overlapping color models due to inaccurate scribble annotations, the DB approach is typically less robust than the WTA approach (Table 6.4, *right*). Analogue to Section 6.3.2, for GC+RW distorted scribbles often cause different disparity falloffs in the conversion results compared to the results obtained from the undistorted scribbles (e.g., Figure 6.9 *b)*). In the case of distorted scribbles, these falloffs also include smooth, erroneous disparity propagations (e.g., Figure 6.9 *b)*).

Overall, this third experiment leads to two main findings concerning the tolerance to inaccurate annotations: **(1)** We quantitatively confirmed that the error tolerance of the tested 2D-to-3D conversion systems can be improved by using scribbling strategies that cover larger, differently colored areas in 2D images and place scribbles closer to each other (e.g., BS). Especially, scribbling strategies that are located close to object borders (BS) were able to avoid (GS+P, GC+RW and LSE) or reduce (CVF (WTA) and CVF (DB)) erroneous annotations to spread across large image areas. **(2)** We observed that systems that use color models (CVF (WTA), GC+RW and CVF (DB)) are more error-tolerant than a system that propagates the disparity from each single annotated pixel (GS+P). The seemingly best results in terms of error-tolerance were obtained with the system that was generally less accurate (LSE). For LSE, the introduced erroneous annotations disappeared in its already inaccurate (i.e., over-smoothed) results.

6.3.4 Content-aware Scribble Placement

It is clear from our previous evaluations that the used scribbling strategies in conjunction with the 2D content influences the 2D-to-3D conversion result. Thus, this section presents a more detailed accuracy evaluation that focuses on the ideal scribble placement in conjunction with different regions in the 2D images. In particular, we investigate four types of image regions: (1) *homogenous* regions that exhibit no or only few color variations, (2) *multicolored* regions that contain different colors or color variations (e.g., texture), (3) regions with *weak* and, contrary, (4) regions with *strong region borders*, that have similar and dissimilar colors as neighboring regions, respectively. While homogenous regions are expected to be ideal for the 2D-to-3D conversion based on color constancy assumptions, multicolored regions are more challenging in this context. If a region is additionally bounded by weak borders, an erroneous propagation of disparities across these borders might lead to errors in the 2D-to-3D conversion results. Contrary, regions with strong borders are less likely to "leak" into neighboring regions. To introduce these four different region classes into our evaluation, we relate each region (from Section 6.2.2) in the smaller version of our dataset (Section 6.2.1) to the region classes that were mentioned above. Specifically, each region is classified into either a (1) homogenous or (2) multicolored region and into either a (3) weak or (4) strong border region (e.g., Figure 6.10). The classifications are based on low-level image features. Essentially, the distinction between (1) and (2) is based on squared color gradients [6] and a region's variance in 2D images. The distinction between (3) and (4) depends on color gradients near their region borders. It should be noted that (4) strong border regions are defined with respect to all neighboring regions. In summary, 86 percent of the pixels in our dataset belong to regions with (3) weak borders and 14 percent of the pixels belong to regions with (4) strong borders. When considering the division into (1) homogenous or (2) multicolored



a) b) c)

Figure 6.10: Region classification example. *a)* 2D image (*Ambush2* from [27]). *b)* Its disparity regions (Section 6.2.2): Each region has a unique color. *c)* Classified regions from *b)*: Multicolored regions with strong borders (*blue*), homogenous regions with strong borders (*red*), multicolored regions with weak borders (*white*), homogenous regions with weak borders (*green*).

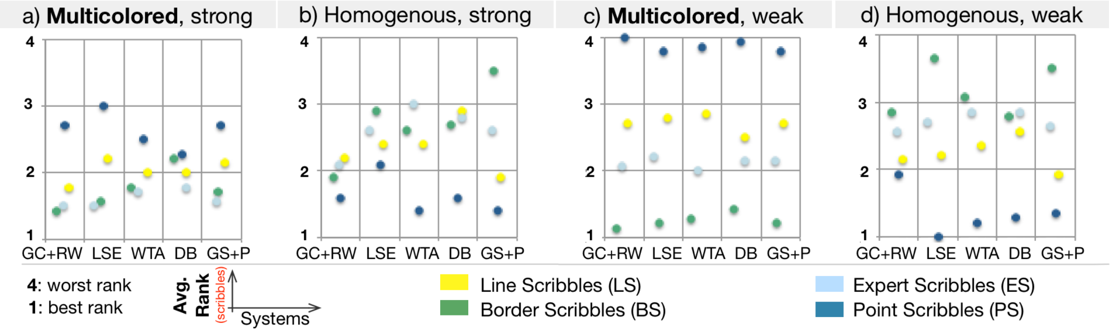


Figure 6.11: Evaluation of content-aware scribble placement (multicolored regions). The evaluated conversion results were generated by changing the scribbling strategy in multicolored regions while keeping it constant (i.e., LS) in homogenous regions. Each plot compares the conversion accuracy achieved with different scribbling strategies placed in multicolored regions. These scribble rankings (*color coded points*) are computed per region (*a)-d)*) and per system.

regions, 9 percent and 91 percent of the pixels in the dataset belong to them, respectively. Given the classification of the regions, we vary the scribbling strategy (Section 6.2.2) in a specific region class (e.g., in all multicolored regions) while keeping the scribbles in the inverse region class (e.g., in all homogenous regions) fixed (i.e., using LS as default scribbling strategy). The resulting 13 sets of annotations per 2D image are used to convert the images with each tested 2D-to-3D conversion system (Section 6.2.3). The conversion results are compared to the images' reference solutions to measure their conversion accuracy in Figure 6.11, Figure 6.12, Figure 6.14 and Figure 6.15. These figures show the average ranks of the scribbling strategies (color-coded *points*) that were measured within each region class. They indicate the ideal scribbling strategy for a specific (*bold* marked) region class (e.g., in Figure 6.11 for multicolored regions) and the scribbling strategies' effect on the inverse region class (e.g., in Figure 6.11 on homogenous regions). In this manner, the evaluation results are discussed below.

Different scribbling strategies in multicolored regions. Figure 6.11 focuses on the choice of scribbling strategy in multicolored regions, while all homogenous regions are marked with LS. Throughout the tested systems, BS are top-performing (low ranks) within multicolored regions with weak (Figure 6.11 *c)*) and ES and BS within multicolored regions with strong

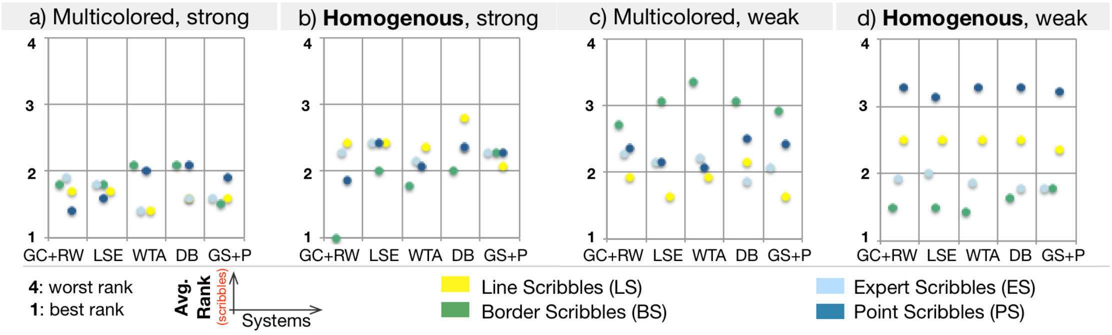


Figure 6.12: Evaluation of content-aware scribble placement (homogenous regions). The evaluated conversion results were generated by changing the scribbling strategy in homogenous regions while keeping it constant (i.e., LS) in multicolored regions. Each plot compares the conversion accuracy achieved with different scribbling strategies placed in homogenous regions. These scribble rankings (*color coded points*) are computed per region (*a)-d)*) and per system.

borders (Figure 6.11 *a)*). Thus, the importance of increased support at weak region borders (BS) and of capturing different colors within multicolored regions (ES or BS) is confirmed.

While using BS or ES in multicolored regions is preferable for the error in multicolored regions, this is not necessarily the case for the error in homogenous regions (i.e., BS and ES have large ranks in Figure 6.11 *b)* and *d)*). For example, the disparity from BS can still “leak“ into neighboring homogenous regions across a weak joint region border.

Different scribbling strategies in homogenous regions. When investigating the choice of scribbling strategy for homogenous regions, it is important to note that only a small amount of pixels (i.e., 9 percent) in our dataset belong to this region class. The key idea of 2D-to-3D conversion, i.e., conversion-generated disparity maps should be piecewise smooth with disparity edges at color edges, implies that homogenous regions are ideal for conversion. The fundamental role of this region class conflicts with its small amount of pixels in our dataset. This suggests an extension of the key assumption of 2D-to-3D conversion to include further constancy assumptions (e.g., consider repetitive color patterns as in the segmentation approaches [5, 77, 79] or motion information as in GS+P’s video version).

In homogenous regions with weak borders all tested systems prefer BS (Figure 6.12 *d)*, low ranks for BS) due to BS’ support at weak region borders. In homogenous regions with strong borders (Figure 6.12 *b)*) the scribbles’ ranks are similar to each other, indicating that all scribbling strategies pose good choices. However, since only 0.02 percent of the pixels in the dataset belong to this region class and since their small region sizes lead to a strong resemblance between scribbling strategies, their evaluation results are not reliable.

The scribble choice for homogenous regions may affect the results in multicolored regions. Since homogenous regions contain approximately a single color, the pixel colors covered by all scribbling strategies are similar to each other when placed in homogenous regions. The difference between the scribbling strategies boils down to the number of their annotated pixels and their spatial positions. This means (i) over-smoothing, (ii) multicolored regions’

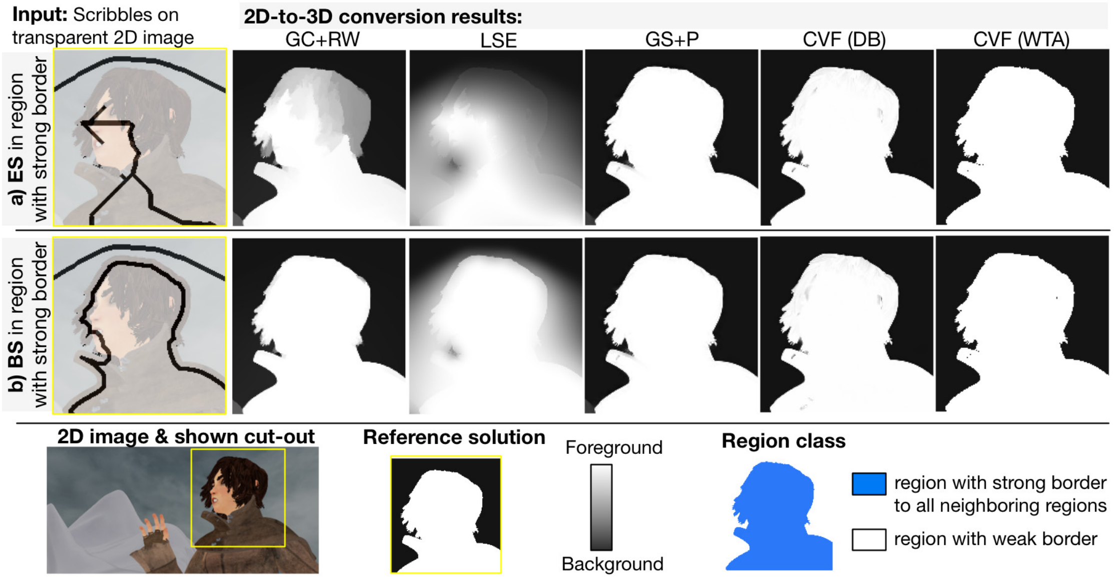

Input: Scribbles on transparent 2D image
2D-to-3D conversion results:
GC+RW
LSE
GS+P
CVF (DB)
CVF (WTA)
a) ES in region with strong border
b) BS in region with strong border
2D image & shown cut-out
Reference solution
Foreground
Background
Region class
region with strong border to all neighboring regions
region with weak border

Figure 6.13: Example of multicolored region with strong borders. *Left:* Scribbles (*black*), i.e., *a)* ES or *b)* BS in strong and LS in weak border region. *Right:* Results of different conversion systems. Generally the systems' connectivity constraints avoid (GC+RW, LSE, GS+P) or reduce (CVF (DB), CVF (WTA)) erroneous propagations into BS' enclosed areas. Specifically in the shown example, for GC+RW and LSE: Compared to ES, BS avoids the propagation of disparities from surrounding scribbles into the strong border region, but causes such errors in surrounding regions. GS+P, CVF (DB) and CVF (WTA): Results with ES and BS are similar to each other. *Bottom:* Cut-out of input image (*Ambush2* from [27]), reference solution and region classes.

insufficient annotations (i.e., LS) and (iii) color similarities between (neighboring) regions can lead to errors within multicolored regions. Multicolored regions with weak borders (Figure 6.12 *c)*) have large ranks when using BS in homogenous regions. This indicates that erroneous propagations into multicolored regions, due to (i)-(iii), are more likely if scribbles in neighboring regions are placed spatially closer to them. Multicolored regions with strong borders were practically not affected by the scribble choice for homogenous regions (Figure 6.12 *a)*, rankings have a narrow range, i.e., below one).

Different scribbling strategies in strong border regions. In strong border regions (Figure 6.14 *a)* and *b)*) we can confirm the observations made above. The preferred (low ranks) scribbling strategies in multicolored regions with strong borders (Figure 6.14 *a)*) are ES and BS. Figure 6.13 illustrates GC+RW's and LSE's preference for BS over ES. For GC+RW and LSE, BS avoid blended disparities in regions' interior, which is not the case with ES. For these two systems the spatial placement of a scribble can regulate a disparity falloff.

The choice of scribbling strategy for strong border regions may affect the results in weak border regions. Related errors in weak border regions can be caused by (i) over-smoothing strong borders and (ii) insufficiently annotated pixel colors (e.g., LS) in either region.

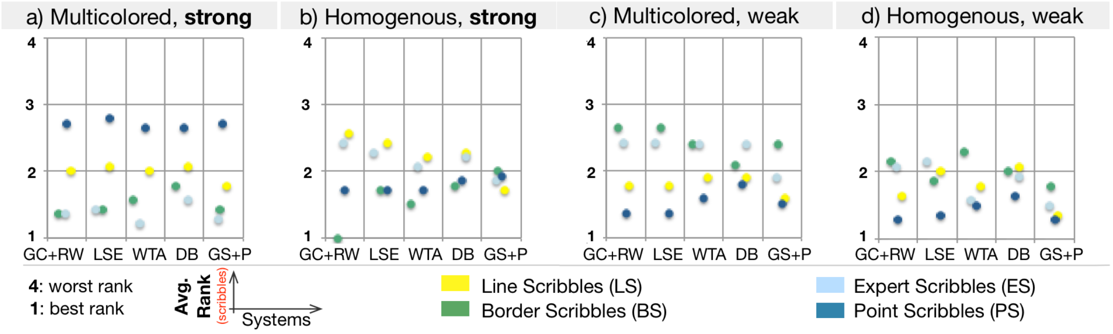


Figure 6.14: Evaluation of content-aware scribble placement (strong border regions). The evaluated conversion results were generated by changing the scribbling strategy in strong border regions while keeping it constant (i.e., LS) in weak border regions. Each plot compares the conversion accuracy achieved with different scribbling strategies placed in strong border regions. These scribble rankings (*color coded points*) are computed per region (*a)-d)*) and per system.

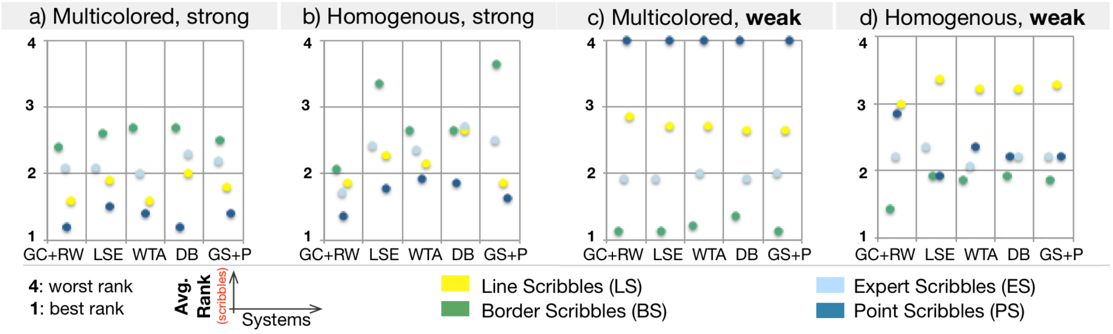


Figure 6.15: Evaluation of content-aware scribble placement (weak border regions). The evaluated conversion results were generated by changing the scribbling strategy in weak border regions while keeping it constant (i.e., LS) in strong border regions. Each plot compares the conversion accuracy achieved with different scribbling strategies placed in weak border regions. These scribble rankings (*color coded points*) are computed per region (*a)-d)*) and per system.

Due to the definition of strong borders with respect to all neighboring regions, (iii) color similarities between a weak and a strong border region do not arise at their joint region border. Furthermore, scribbles that are placed (vi) spatially close to neighboring regions are prone to erroneous propagations across strong borders. In most cases, the scribbles' ranks in weak border regions (Figure 6.14 *c)* and *d)*) are similar to each other, which indicates little influence from the scribble choice in strong border regions. GC+RW and LSE are the exceptions to this observation. For these systems, BS and ES have a larger effect on multicolored regions with weak borders (Figure 6.14 *c)*), i.e., might cause errors (e.g., due to (i)) in multicolored regions with weak borders (e.g., Figure 6.13 *b)*, LSE)

Different scribbling strategies in weak border regions. Throughout the tested systems BS are top-performing (low ranks) within regions with weak borders (Figure 6.15 *c)* and *d)*). This observation further confirms that weak border regions are preferably annotated using BS.

Analogue to previous discussions, the scribble choice for weak border regions is not necessarily ideal for strong border regions. Reasons for related errors in strong border regions include the points (i)-(iv) from above. For example, with BS propagations into strong border regions, due to (i)-(iv), are more likely than with scribbles that are placed closer to region centers (Figure 6.15 *a)* and *b)*, larger ranks for BS than ES, LS and PS). In multicolored regions with strong borders (Figure 6.15 *a)*) a similar effect can be observed for ES. When using ES (or BS) in weak border regions, (ii) insufficient scribble annotations (i.e., LS) are only present in multicolored regions with strong borders. Thus, in these regions, colors that are not captured by LS might be assigned to a disparity from a scribble (BS or ES) that is associated with a similar (or the same) color as the missing one.

Overall, this in-depth evaluation revealed four main insights concerning the scribble placement: **(1)** While the key assumption of 2D-to-3D conversion focuses on homogenous regions, ideally with high-contrast color edges, this region class turned out to be uncommon in our dataset. We suspect similar behavior for other (recorded) 2D content and, hence, believe that extensions of the common assumption of color constancy would be beneficial for 2D-to-3D conversion systems. **(2)** Our evaluation results highlighted the importance of capturing different colors within objects (ES or BS) and providing additional spatial constraints close to low-contrast object borders (BS) for all tested conversion systems. **(3)** While in many aspects the investigated conversion systems behaved similarly, they differed in some. For example, the smoothing effect of systems that employ global edge-aware interpolations (LSE and GC+RW) is (also) regulated by the spatial positions of the scribbles and, hence, ideally taken into account when annotating the 2D content. For these systems the annotation of fronto-parallel objects might require a scribbling strategy that provides more (spatial) constraints (BS). Vice versa, systems that perform less or a different kind of smoothing (GS+P, CVF (WTA) and CVF (DB)) might benefit from additional scribbles when indicating disparity falloffs. **(4)** We highlighted that the choice of scribbling strategy not only influences the object that is annotated with it, but may also influence nearby objects. While this was expected, we showed that this aspect should ideally be considered when choosing a scribbling strategy. The prime example in this context were objects with low-contrast borders. Our evaluations suggested that ideally both sides of a low-contrast border are annotated with scribbles that are placed close to them (BS). In general, using a more adequate annotation strategy (BS or ES) in one region often caused erroneous propagations into regions that were insufficiently annotated (e.g., LS).

6.4 Summary

This chapter has evaluated the impact of the user input, i.e., scribble-based user annotations, on five different semi-automatic 2D-to-3D conversion systems. In the course of this evaluation, the tested conversion systems were compared as well. We have automatically generated sets of scribble inputs that follow four different scribbling strategies. They simulate minimalistic and effortless scribbling strategies as might be drawn from a system-unaware user as well as more advanced and labor-intensive scribbling strategies that take the content of the 2D images into account. Specifically, we use minimalistic point scribbles (PS), similar effortless line scribbles (LS) that cover larger areas than PS, expert scribbles (ES) that deliberately annotate different image colors and even more labor-intensive border scribbles (BS), which trace object borders. We have quantitatively compared 2D-to-3D conversion results of different 2D-to-3D conversion systems and different scribbling strategies with corresponding reference solutions.

The first experiment compared the accuracy of results from different systems and scribbling strategies. The evaluations confirmed that the choice of scribbling strategy affects the accuracy of the conversion results and the benefit of putting more effort into the annotation process (ES or BS). The top-performing conversion system was our segmentation-based algorithm from Chapter 4 (GS+P). Tested systems that use segmentation information (GS+P, GC+RW, CVF (WTA) and CVF (DB)) in most cases outperformed the system that does not rely on segmentation (LSE).

The second experiment focused on the robustness against scribble perturbations. We could neither identify a single scribbling strategy nor 2D-to-3D conversion system as the most robust throughout. In systems that rely on global interpolations (GC+RW and LSE), perturbations manifested themselves in changes of disparity falloffs in their results. Systems that are based on local decisions or smooth locally (GS+P, CVF (WTA) and CVF (DB)), often assigned large areas to a different disparity after a perturbation. In the latter systems changes were visually and quantitatively more noticeable.

The third experiment addressed inaccurate user annotations by performing the 2D-to-3D conversions on sets of distorted scribbles that contained erroneous disparity assignments to pixels. We have confirmed that the tested systems' error tolerance can be improved by using scribbling strategies that cover larger, differently colored areas in 2D images and place scribbles spatially closer to each other (e.g., BS). Systems that use color models (CVF (WTA), GC+RW and CVF (DB)) were more error-tolerant than a system that propagates disparities from individual pixels (GS+P). The seemingly most error-tolerant system was generally less accurate (LSE). The introduced erroneous annotations were hardly noticeable in its already inaccurate results.

The fourth experiment focused on the systems' ideal scribbling strategies with respect to different types of image regions (i.e., multicolored or homogenous, strong or weak border regions). Generally, the tested systems preferred scribbling strategies that capture different colors for multicolored regions and scribbles close to weak object borders (with low color-contrast). Ideally, the characteristics of the used systems, such as their respective smoothing effects, are taken into account when annotating 2D content. For example, when annotating fronto-paralell objects, systems hat employ global edge-aware interpolations (LSE and GC+RW) might require a scribbling strategy that provides more (spatial) constraints (BS). When analyzing interactions between different region types that are annotated using different scribbling strategies, we noticed that the usage of a more adequate scribbling strategy in one region (e.g., BS in a weak border region) often caused erroneous propagations into nearby regions that were not sufficiently annotated.

Overall, our evaluation study provided practical insights into the scribble-based annotation process in the context of the tested semi-automatic 2D-to-3D conversion systems. Our evaluations have shown that none of the tested conversion systems excels in all four experiments with each scribbling strategy. However, two segmentation-based systems, i.e., our 2D-to-3D conversion from Chapter 4 (GS+P) and a system that combines segmentation and edge-aware interpolation (GC+RW), achieved high conversion quality. Ideally, these systems are used with a scribbling strategy that places scribbles close to object borders (BS), which improves their conversion results, robustness to scribble perturbations and tolerance to inaccurate annotations.

CHAPTER 7

Conclusions and Future Work

7.1 Conclusions

This thesis has focused on the problem of cost-efficiently converting monoscopic (2D) videos to stereoscopic (3D) videos based on sparse user-given disparity information. This semi-automatic 2D-to-3D conversion process ideally (i) requires only minimal user input and (ii) efficiently generates disparity maps of high conversion quality. In this thesis we have addressed both aspects. We have proposed two semi-automatic 2D-to-3D conversion algorithms that rely on spatio-temporal video analysis. In particular, we have exploited the similarities between the problems of 2D-to-3D conversion and video segmentation by integrating segmentation techniques into the conversion process. While 2D-to-3D conversion propagates sparse disparities between pixels that are similar (in terms of color) to each other, segmentation techniques group pixels into segments according to some homogeneity criterion (e.g., color). This suggests that pixels within the same segment are likely to be assigned the same or a similar disparity. In our work, the segmentation information has been used to improve the quality of the resulting disparity maps near object borders and to enable coherent disparity interpolations over time.

In our first algorithm, the conversion is performed jointly with an automatic graph-based video segmentation. The scribbles' purpose is only to initialize some pixels with disparities. The algorithm propagates available disparities between neighboring pixels while assigning them to the same segment. In this manner, we have generated 2D-to-3D conversion results with sharp disparity edges close to object borders in the 2D video. A selective filtering step further enables disparity changes within segments and over time. For this algorithm, we have provided an optimized implementation that achieves interactive runtimes (one fps for a resolution of 634×480 pixels).

The second semi-automatic 2D-to-3D conversion algorithm builds upon an interactive video object segmentation algorithm that was developed in this thesis. The segmentation algorithm achieves fast-processing speeds (250 fps for a video with a resolution of 620×360 pixels) by using a framework that solves label-based optimization problems within a frame-wise filtering scheme. In order to improve temporal coherency in the context of object segmentation, we have expanded the filtering scheme to the temporal domain. In the proposed 2D-to-3D conversion algorithm, pixels covered by one and the same scribble are marked as belonging to the same segment and their disparity is defined by globally describing their appearance (using color models) in the 2D video. According to these color models, we perform a multi-label segmentation, in which pixels within a segment are assigned to the same or similar disparities. The segmentation result further enables us to analyze arising occlusions between segments. By performing temporal disparity interpolations that respect the observed occlusions, we are able to obtain perceptually more coherent disparity maps and, hence, improve the quality of our conversion results.

We have experimentally shown that our algorithms achieve high conversion quality on computer-generated and recorded 2D videos and are competitive when being compared to related algorithms. In a final evaluation study, we have compared our algorithms with established semi-automatic 2D-to-3D conversion algorithms over a dataset with ground truth reference solutions. These evaluations are performed under consideration of different scribbling strategies, hence, we have generated 205 different conversions per image in the dataset. As expected, the results of the tested algorithms were affected by the chosen scribbling strategy and all tested algorithms preferred labor-intensive scribbling strategies over minimalistic ones. In terms of accuracy, our first algorithm, which is based on a graph-based segmentation (Chapter 4), was the top-performer for all investigated scribbling strategies. The tested conversion algorithms that incorporate segmentation algorithms, including our first (Chapter 4) and second (Chapter 5) algorithm, outperformed a well-established 2D-to-3D conversion algorithm [56] that does not incorporate segmentation algorithms in its conversion process. In this context, both our algorithms' sharp disparity edges close to object borders were a key advantage over the mentioned competing algorithm. Further evaluations revealed that our first algorithm (Chapter 4) is more sensitive to scribble perturbations and inaccurate annotations than its competitors. In these aspects, our second conversion algorithm (Chapter 5) was more stable, even though its quantitative results were outperformed by a competitor [56] that over-smoothed its conversion results. Concerning the robustness to scribble perturbations, we observed quantitatively better results for algorithms that employ global interpolations [56, 118]. Concerning inaccurate user annotations, algorithms that use color models, such as our second conversion algorithm (Chapter 5), were found to be more error-tolerant than algorithms that propagate the disparity from each single pixel, as our first conversion algorithm (Chapter 4). A final evaluation sought to shed more light on the ideal scribbling strategy in conjunction with different image areas to provide practical insights into the annotation process. It highlighted, e.g., the importance of drawing scribbles close to object borders that separate similarly colored objects and annotating differently colored areas within objects with one and the same scribble.

7.2 Future Research Topics

We have demonstrated that semi-automatic 2D-to-3D conversion is a promising alternative to laborious manual 2D-to-3D conversion. The integration of segmentation algorithms into the conversion process enabled us to improve the quality of generated 3D content. However, there are various open topics that can be pursued in future research. Some of them are listed below:

- The semi-automatic 2D-to-3D conversion algorithms that were presented in this thesis (Chapter 4 and Chapter 5) are based on the same common key assumption that pixels with similar colors are likely to have similar disparities. However, this assumption becomes unreliable for low-contrast scenes and scenes that contain objects with similar colors. In addition, objects that should obtain similar disparities might contain quite different colors. As shown in Chapter 6, both cases require additional user involvement. While our algorithms have incorporated motion information to guide the propagation process, further research might seek to extend the usage of motion information to long-term trajectories as in [86], to consider hints such as repetitive color patterns and image gradients as in [5, 8, 79, 122] or to further constrain the generated disparity maps by simultaneously generating novel views similarly to [163].

- While our interactive video object segmentation algorithm (in Chapter 5) adopted a progressive annotation strategy, our semi-automatic 2D-to-3D conversion algorithms trigger their respective conversion procedures after all scribbles have been drawn. Consequently, adjusting the scribble input to improve a conversion result also requires its recomputation. When using a progressive annotation strategy, this recomputation might only concern image regions that are connected to the adjusted scribble and, thus, could be performed faster. Furthermore, visual previews of the conversion results might facilitate the annotation process. We believe that addressing this point in future research could lead to the development of even more practical and intuitive 2D-to-3D conversion systems.
- In Chapter 4, we showed that semi-automatic 2D-to-3D conversion naturally fits with a graph-based (automatic) video segmentation technique. While it led to excellent conversion results, the concept of performing disparity propagation during video segmentation still might be improved upon. For example, disparity scribbles could be treated as soft instead of hard constraints to increase the algorithm's error tolerance. Extending the region-graph by sparse edges that respect a 3D connectivity constraint might further reduce the amount of required scribbles. Further research could also focus on increasing the perceptual coherence of the first algorithm, for example, by exploring long-range temporal interactions among objects [86] and adapting the propagation rule set accordingly.
- In Chapter 5, we took a step towards the generation of perceptually coherent disparity maps. A promising direction of research would be refining and extending this basic idea by focusing on the generation of 3D content with high chances of visual comfort in general.
- In this thesis (e.g., Chapter 6), evaluations have focused on comparisons of conversion-generated disparity maps with reference solutions. However, our publication [106] indicates that in stereoscopic viewing conditions, objective evaluations only weakly correlate with subjective ones. Distorted 3D content might still create plausible 3D effects [73, 106]. Thus, subjective evaluation studies of conversion-generated 3D content are an interesting topic to pursue. Setting apart subjectively distracting and insignificant errors in disparity maps would further enable future 2D-to-3D conversion algorithms to focus on the former.
- Related to the previous point, future work could address the usability of semi-automatic 2D-to-3D conversion systems. These studies might consider the annotation time and amount of user interaction required to generate subjectively satisfactory conversion results. In this context, the scribble-based annotation process itself could be investigated further. In this thesis, we have performed scribble-based annotations and, based on this subjective experience, have automatically generated them in Chapter 6. However, no user study was performed to further validate our experiences. We believe that such studies could lead to a more thorough understanding of the scribble-based annotation process.

APPENDIX A

Descriptions of Used Datasets

Throughout this thesis evaluations were performed using images and videos from different datasets. This appendix provides an overview and more detailed descriptions of the used data.

Tsukuba dataset. Our Tsukuba dataset (Figure A.1) consists of three sub-videos (resolution of 640×480 pixels) that were taken from the new Tsukuba dataset [102]. This dataset [102] is a computer-generated stereoscopic video in which a moving camera captures a static scene. It is provided with GT disparity maps that contain 256 disparity levels. The video was rendered with four different illuminations (see [102] for details), from which we use the *daylight* rendering (smooth illumination except for a few over-exposed areas). The used sub-videos, i.e., *Tsukuba1, Tsukuba50* and *Tsukuba380*, have 17, 18 and 100 frames, respectively. The number in their names is the number of their first frame according to the frame numbering of the video in [102] (e.g., the first frame of *Tsukuba1* is the first frame in [102]). Although being computer-generated, the videos pose several challenges to 2D-to-3D conversion algorithms. They contain, for example, temporal disparity changes (e.g., *Tsukuba1* and *Tsukuba50*), rounded objects (e.g., cans in *Tsukuba380*) and color similarities between objects at different disparities (e.g., camera and shelf in *Tsukuba1*).

Sintel dataset. Our Sintel dataset (Figure A.2) consists of ten videos (resolution of 1024×436 pixels) taken from the MPI Sintel Flow dataset [27, 166]. Their publicly available training data[1] contains computer-animated videos and GT OF [27]. The dataset was later extended by disparity, depth and segmentation GT. Since this data was not available at the time of our evaluations, we used depth GT (in meters) that was kindly provided by the authors before its release. The videos are provided in different levels of difficulty, i.e., *final* with illumination effects, *clean* with shading only and *albedo* with none of the mentioned effects. If not explicitly mentioned otherwise, we use *albedo* renderings. It is worth to note that although *albedo* constitutes the lowest difficulty level for evaluating OF estimation algorithms, it still poses several challenges to 2D-to-3D conversion algorithms, e.g., temporal disparity

---

[1] http://sintel.is.tue.mpg.de

changes (e.g., *Sleeping1*), rounded and slanted objects (e.g., *Ambush7*), color ambiguities (e.g., *Ambush5*) and fast moving objects (up to 100 pixels per frame [27], e.g., *Ambush2*). Contrary to the Tsukuba videos, the Sintel videos show various scenes and contain camera and object motion (e.g., *Ambush2, Shaman3*). The *final renderings* of these scenes provide even more challenges than the *albedo renderings*, including larger color ambiguities and low-contrast scenes (e.g., *Ambush2*).

Long video dataset. This video dataset (Figure A.3) consists of three longer videos, i.e., *Child*, *Head* and *Interview*, with 41, 81 and 101 frames, respectively. The videos were taken from [20, 42], i.e., *Child* (600×338 pixels), *Head* (600×330 pixels) and *Interview* (600×480 pixels). Our long video dataset contains computer-generated videos and a recorded video, i.e., *Interview*. In case of *Interview*, the provided reference solution contains depths that was recorded with a special camera [42]. For all their videos, [20] only provides disparity or depth values for a small number of frames, i.e., three frames for *Child* and five frames for *Head* and *Interview*. In this dataset, the depth values or disparities in the reference solutions are quite stable with only slight changes over time. However, they exhibit depth or disparity variations within objects, i.e., rounded and slanted objects (e.g., lawn in *Child*, cap in *Head*, table in *Interview*). All three videos contain color similarities between objects at different distances from the camera (e.g., head and blind in *Interview*).

Middlebury image dataset. The well-known Middlebury stereo benchmark [134], that was originally intended for evaluating stereo matching algorithms, provides still images with corresponding GT disparity maps. Its publicly available training dataset contains 35 stereo pairs with GT disparity maps that were obtained using different methods, e.g., using structured light (see [134] and the Middlebury stereo evaluation website[2] for details). Figure A.4 shows the left images of the stereo pairs, that were used when evaluating 2D-to-3D conversion algorithms, and their disparity GT. On average, the images in this dataset have a resolution of about 432×365 pixels. They cover a great variety of scenes, including cluttered scenes with many small objects (e.g., *Dolls*) and scenes that contain only few large objects (e.g., *Monopoly*), scenes with large homogenous regions (e.g., *Wood1*) and highly-textured scenes (e.g., *Cloth1*), color similarities between objects at different disparities (e.g., *Books*) and low-contrast images (e.g., *Lampshade1*), slanted and rounded objects (e.g., *Bowling1*), but also predominantly fronto-parallel scenes (e.g., *Laundry*).

Recorded stereo dataset. We use a recorded stereo dataset (Figure A.5) which consists of the five videos, i.e., *Parade*, *Palace*, *City*, *Stairs* and *Football*, that were recorded with a stereoscopic camera. *Parade*, *Palace*, *City* and *Stairs* have a resolution of 689×282 pixels and consist of 11, 10, 18 and 20 frames, respectively. *Football* has a resolution of 669×576 pixels and 21 frames. When applying 2D-to-3D conversion algorithms to these videos only one view (i.e., left view) is used. The stereoscopic videos were recorded without capturing GT disparities. We generated reference solutions by applying a stereo matching algorithm [12] that computes disparities for each video. The videos introduce complexities such as object and camera motion (e.g., *Parade*), partial occlusions (e.g., walking person in

---

[2] http://vision.middlebury.edu/stereo

Palace), temporal disparity changes (e.g., person in *Stairs*), rounded or slanted surfaces (e.g., field in *Football*) and color ambiguities (e.g., bushes in *Stairs*). Since these videos are recorded, their image quality is lower than the image quality of computer-generated test data. Furthermore, their stereo matching-generated reference solutions are less accurate than the rendered ground truth reference solutions from the computer-generated data.

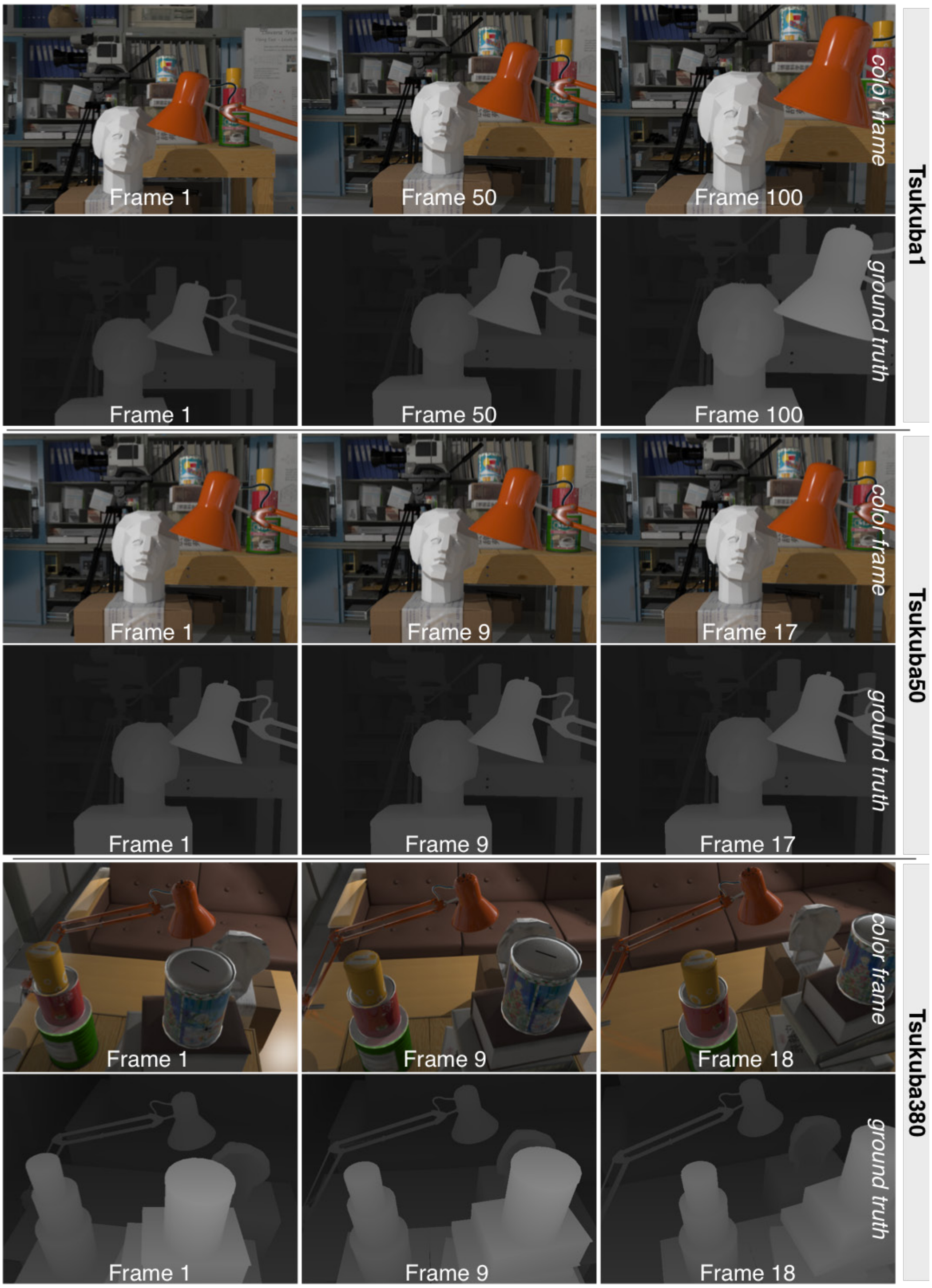


Figure A.1: Tsukuba dataset. Videos *Tsukuba1, Tsukuba50* and *Tsukuba380*: Frames (*top*) and their disparity GT (*bottom*) from [102]. Foreground: *bright*, background: *dark*.

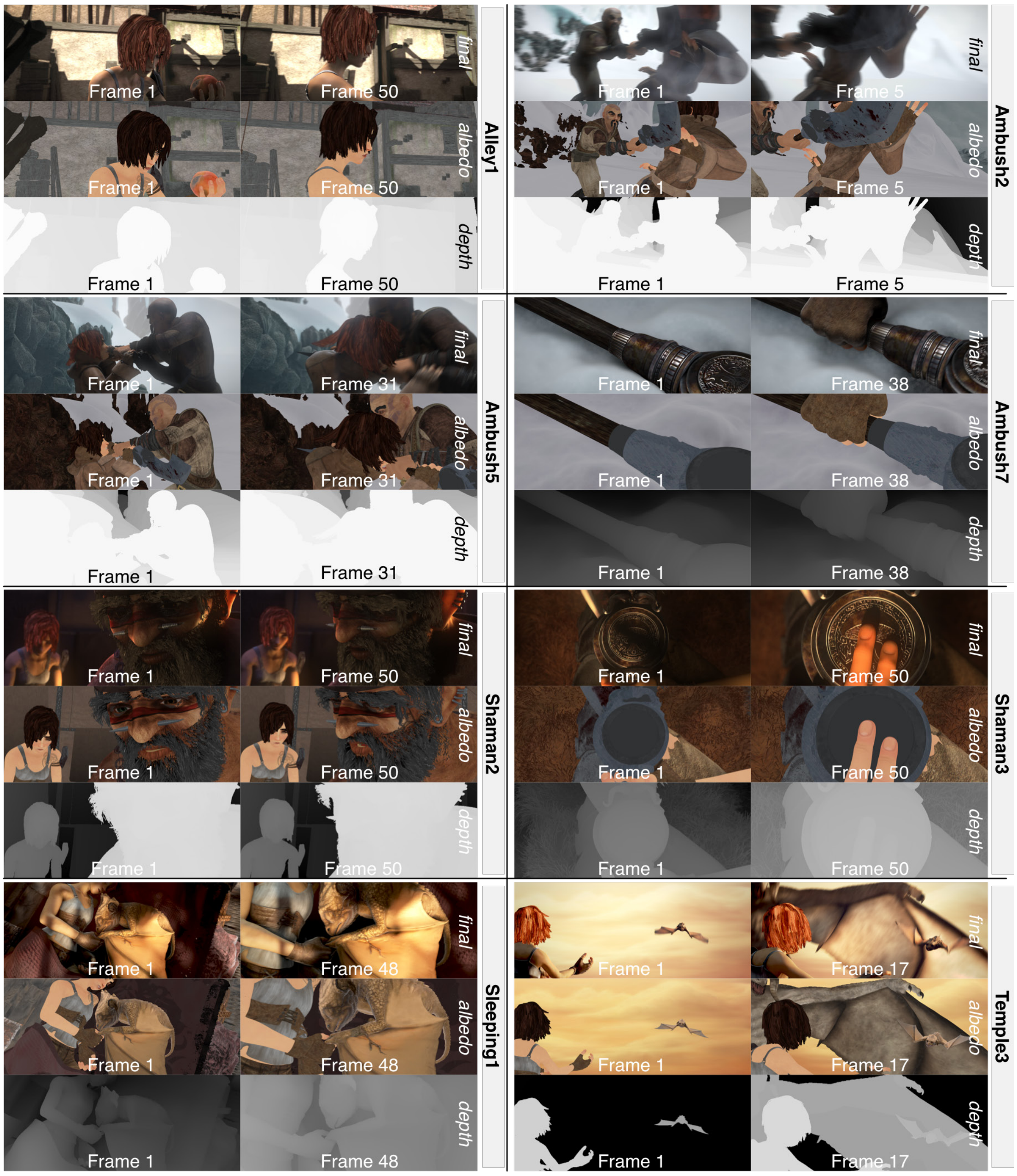


Figure A.2: Sintel dataset. Videos *Alley1, Ambush2, Ambush5, Ambush7, Shaman2, Shaman3, Sleeping1* and *Temple1*: *Final* and *albedo* renderings of first and last frame of each video (*top*) and corresponding inverted depth GT (*bottom*) that was provided by the authors from the MPI Sintel dataset [27, 166]. Foreground: *bright*, background: *dark*.

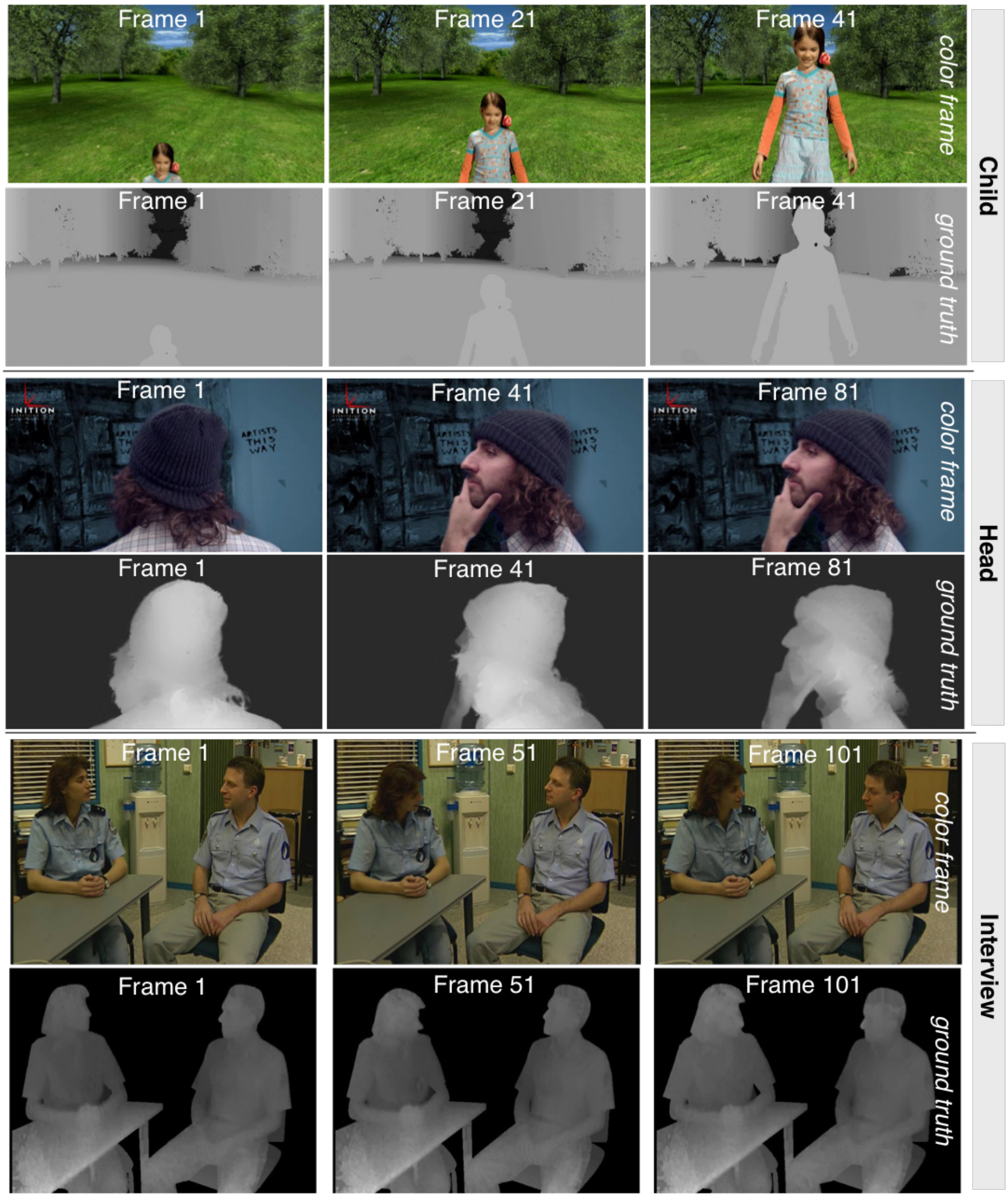


Figure A.3: Long video dataset. Videos *Child*, *Head* and *Interview*: Frames (*top*) and corresponding reference solutions (*bottom*) from [20, 42]. Foreground: *bright*, background: *dark*.

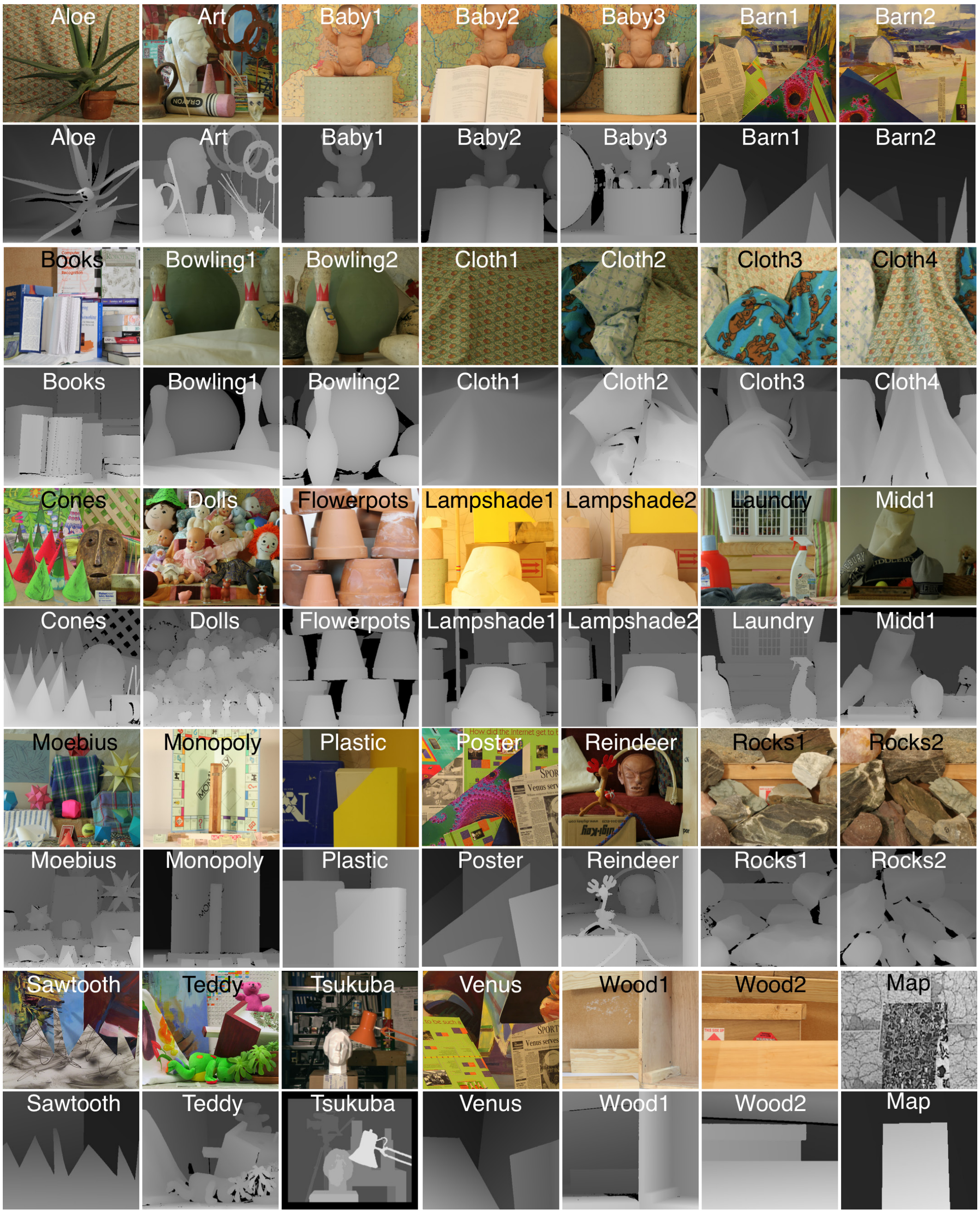


Figure A.4: Middlebury image dataset. 35 images (left views) (*top*) and corresponding disparity GT (*bottom*) from [134]. Foreground: *bright*, background: *dark*, no GT: *black*.

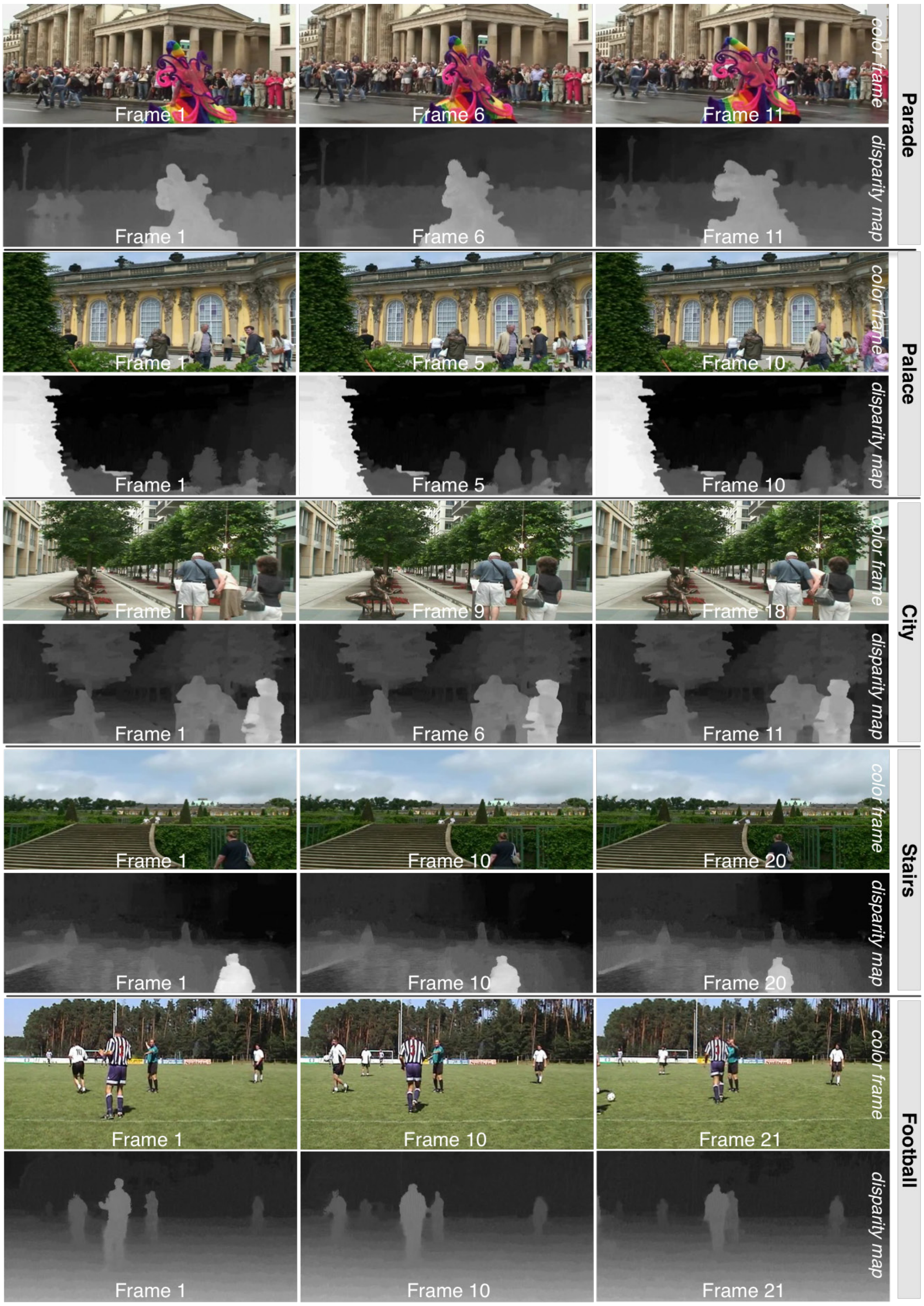


Figure A.5: Recorded stereo dataset. Color videos (left views) *Parade*, *Palace*, *City*, *Stairs* and *Football* (*top*) and reference solution (*bottom*). Foreground: *bright*, background: *dark*.

APPENDIX B

Supplements to Cost Volume Filtering for 2D-to-3D Conversion

This appendix provides a program flow and additional evaluation results that were not shown in Chapter 5 for our CVF-based 2D-to-3D conversion algorithm (Section 5.2.2).

B.1 Program Flow

Figures B.1 - B.6 provide the program flow for our CVF-based 2D-to-3D conversion algorithm that was described in Chapter 5 (in Section 5.2.2). As shown in Figure 5.5, this algorithm consists of six components: (1) The generation of cost volume P from multiple scribbles (*MS*, Figures B.1), (2) the spatio-temporal closeness constraint (*STC*, Figures B.2) and (3) cost volume filtering (*CVF*, Figures B.3) without or with motion guided filtering (*+TC*) to obtain a filtered cost volume P'. Subsequently, (4) temporal disparity change models (DC, Figures B.6) that correct naive interpolations (*-n*) by guided interpolations (*-g*) can be applied. They can either be applied with respect to time (*-tM*) or segment size $s(.)$ and motion (*-sM*). Finally, (5) a 3D connectivity constraint (*CON*, Figures B.5) and (6) the disparity assignment using a winner-takes-all (*WTA*) or a disparity blending (*DB*) scheme yields a final disparity video (Figures B.4). Components (1), (3) and (6) build the core algorithm, while the remaining components, i.e., (2), (4) and (5) are optional. All six components are discussed in Section 5.2.2 and their respective program flow is given in Figures B.1 - B.6. Figure B.6 focuses on depth order guided interpolations according DC -tM and DC -sM.

B.2 Additional Evaluation Results

B.2.1 Comparison of Different Algorithm Versions with Depth Blending

Table B.1 provides additional quantitative evaluation results for our CVF-based 2D-to-3D conversion algorithm (Section 5.3.2) in Chapter 5. It compares different versions of our algorithm

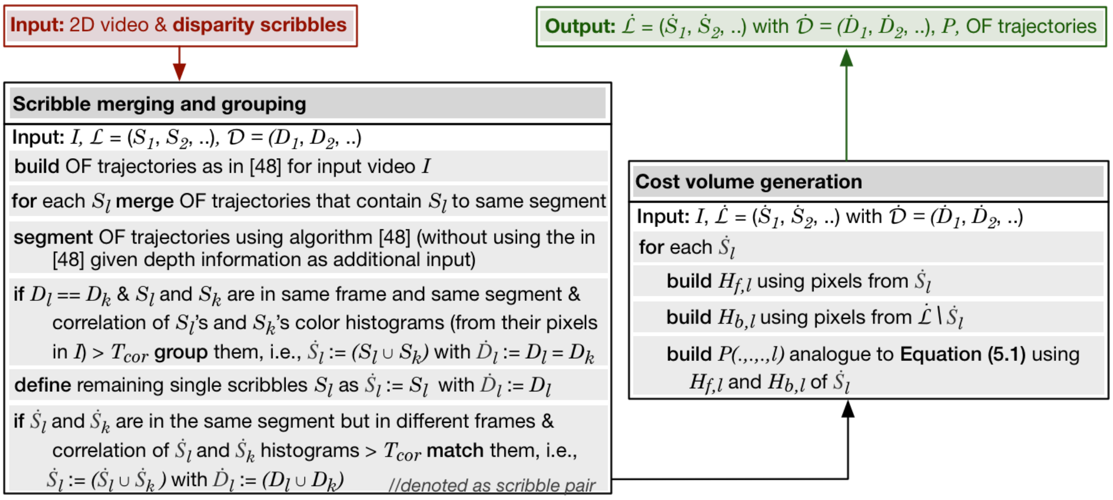

Input: 2D video & disparity scribbles
Scribble merging and grouping
Input: I, $\mathcal{L} = (S_1, S_2, ..)$, $\mathcal{D} = (D_1, D_2, ..)$
build OF trajectories as in [48] for input video I
for each S_l merge OF trajectories that contain S_l to same segment
segment OF trajectories using algorithm [48] (without using the in [48] given depth information as additional input)
if $D_l == D_k$ & S_l and S_k are in same frame and same segment & correlation of S_l's and S_k's color histograms (from their pixels in I) > T_{cor} group them, i.e., $\dot{S}_l := (S_l \cup S_k)$ with $\dot{D}_l := D_l = D_k$
define remaining single scribbles S_l as $\dot{S}_l := S_l$ with $\dot{D}_l := D_l$
if $\dot{S}_l$ and $\dot{S}_k$ are in the same segment but in different frames & correlation of $\dot{S}_l$ and $\dot{S}_k$ histograms > T_{cor} match them, i.e., $\dot{S}_l := (\dot{S}_l \cup \dot{S}_k)$ with $\dot{D}_l := (D_l \cup D_k)$ //denoted as scribble pair
Cost volume generation
Input: I, $\dot{\mathcal{L}} = (\dot{S}_1, \dot{S}_2, ..)$ with $\dot{\mathcal{D}} = (\dot{D}_1, \dot{D}_2, ..)$
for each $\dot{S}_l$
build $H_{f,l}$ using pixels from $\dot{S}_l$
build $H_{b,l}$ using pixels from $\dot{\mathcal{L}} \setminus \dot{S}_l$
build $P(.,.,.,l)$ analogue to Equation (5.1) using $H_{f,l}$ and $H_{b,l}$ of $\dot{S}_l$
Output: $\dot{\mathcal{L}} = (\dot{S}_1, \dot{S}_2, ..)$ with $\dot{\mathcal{D}} = (\dot{D}_1, \dot{D}_2, ..)$, P, OF trajectories

Figure B.1: Program flow for MS. User-provided scribbles $\mathcal{L} = (S_1, ..., S_L)$ with user-selected disparities $\mathcal{D} = (D_1, ..., D_L)$ are grouped and matched using a motion segmentation algorithm [48] and color comparisons (using a threshold T_{cor}). The color comparisons use the colors of pixels in the input video I that were marked by specific scribbles. This results in grouped and matched scribbles $\dot{S}_l$'s $\in \dot{\mathcal{L}}$ with $\dot{D}_l \in \dot{\mathcal{D}}$, OF vectors and their derived OF trajectories as a byproduct of [48]. The cost volume $P(x, y, t, l)$ contains probabilities that pixels $i = (x, y, t)$ belong to scribbles $\dot{S}_l$. P is generated from color histograms $H_{f,l}$ and $H_{b,l}$ built for each $\dot{S}_l$.

(i.e., MS, STC, CON, +TC, DC-tM and DC-sM) when using depth blending (DB) on a set of computer-generated videos with depth GT from [27]. The corresponding results that were generated with WTA (Table 5.3) and DB (Table B.1) behave similarly. Specifically, if enabling a component improves the WTA result, it typically also improves the DB result. Analogously, as in Table 5.3, when comparing the MSEs with GT OF and estimated OF in Table B.1, the average quality is hardly affected by the change of OF.

B.2.2 Quantitative Comparison of Naive and Depth Order Guided Interpolation

Table B.2 and Table B.3 provide additional quantitative evaluation results for our CVF-based disparity propagation (Section 5.3.2) in Chapter 5. In this context, Table B.2 and Table B.3 compare a naive temporal depth interpolation technique with our depth order guided temporal interpolation technique. The evaluation is performed on a set of computer-generated videos with corresponding depth GT from [27]. In Table B.2 our 2D-to-3D conversion algorithm uses GT motion information, i.e., GT OF, while in Table B.3 the comparison is conducted with estimated OF [96]. When we compare the quantitative evaluation results (i.e., in Table B.2 and Table B.3) of the different interpolation techniques, we observe that their MSEs are very similar. When applying our algorithm with estimated OF, our depth order guided interpolation quantitatively improves the results on average by 19 percent. For estimated OF, the major quantitative improvement

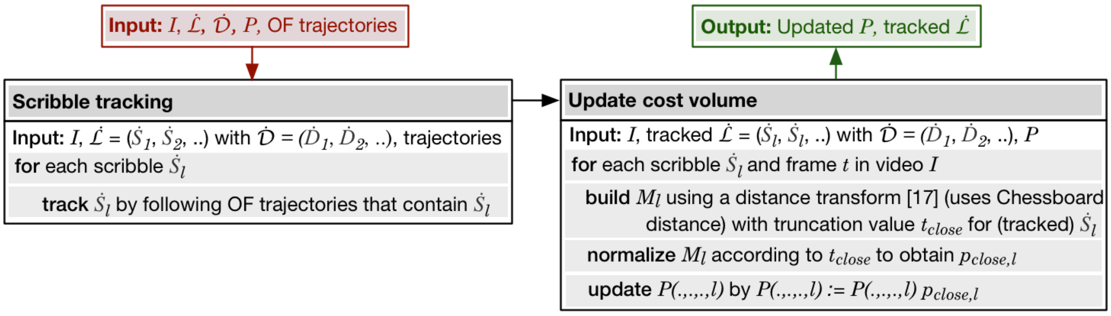

Input: I, $\dot{\mathcal{L}}$, $\dot{\mathcal{D}}$, P, OF trajectories
Output: Updated P, tracked $\dot{\mathcal{L}}$
Scribble tracking
Input: I, $\dot{\mathcal{L}} = (\dot{S}_1, \dot{S}_2, ..)$ with $\dot{\mathcal{D}} = (\dot{D}_1, \dot{D}_2, ..)$, trajectories
for each scribble $\dot{S}_l$
track $\dot{S}_l$ by following OF trajectories that contain $\dot{S}_l$
Update cost volume
Input: I, tracked $\dot{\mathcal{L}} = (\dot{S}_l, \dot{S}_l, ..)$ with $\dot{\mathcal{D}} = (\dot{D}_1, \dot{D}_2, ..)$, P
for each scribble $\dot{S}_l$ and frame t in video I
build M_l using a distance transform [17] (uses Chessboard distance) with truncation value t_{close} for (tracked) $\dot{S}_l$
normalize M_l according to t_{close} to obtain $p_{close,l}$
update $P(.,.,.,l)$ by $P(.,.,.,l) := P(.,.,.,l)\ p_{close,l}$

Figure B.2: Program flow for STC. Scribbles $\dot{S}_l \in \dot{\mathcal{L}}$ with disparities $\dot{D}_l \in \dot{\mathcal{D}}$ are tracked throughout the video I by following OF trajectories (from *MS*) that contain $\dot{S}_l$'s marked pixels. Each cost volume slice $P(.,.,t,l)$ is updated using a confidence weight $p_{close,l}$ which was computed from a (truncated) distance transform [17] M_l that measures the distance of each pixel to a (tracked) scribble $\dot{S}_l$ in a frame t. The truncation value t_{close} is typically set to the maximal possible value (i.e., frame diagonal) for all scribbles.

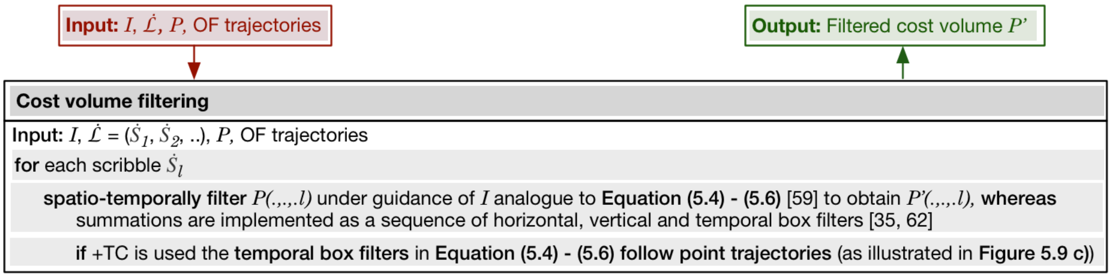
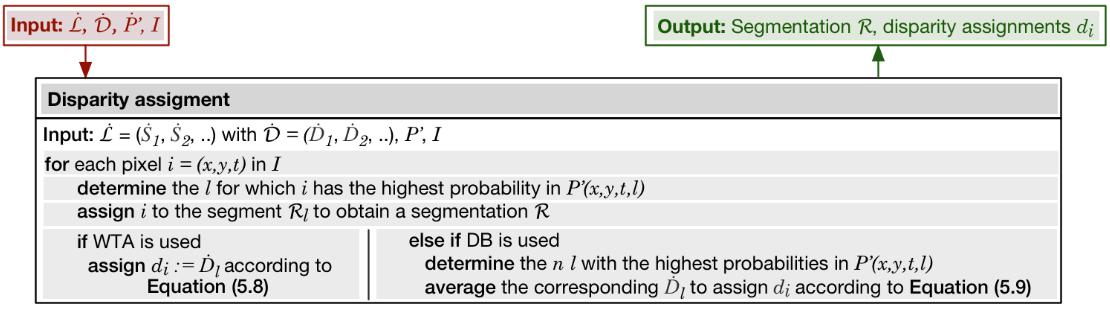

Input: I, $\dot{\mathcal{L}}$, P, OF trajectories
Output: Filtered cost volume P'
Cost volume filtering
Input: I, $\dot{\mathcal{L}} = (\dot{S}_1, \dot{S}_2, ..)$, P, OF trajectories
for each scribble $\dot{S}_l$
spatio-temporally filter $P(.,.,.l)$ under guidance of I analogue to Equation (5.4) - (5.6) [59] to obtain $P'(.,.,.l)$, whereas summations are implemented as a sequence of horizontal, vertical and temporal box filters [35, 62]
if +TC is used the temporal box filters in Equation (5.4) - (5.6) follow point trajectories (as illustrated in Figure 5.9 c))

Figure B.3: Program flow for CVF. Each cost volume slice $P(.,.,.,l)$ for a scribble $\dot{S}_l \in \dot{\mathcal{L}}$ is filtered under the guidance of the input video I to obtain its filtered version $P'(.,.,.,l)$. Either the common GF [59] or our motion guided GF (+*TC*) that filters along OF trajectories is used.

can be observed for *Ambush5*, which contained perceptual conflicts between large foreground objects (i.e., large depth differences). For test videos in which such conflicts occurred for smaller objects (e.g., *Temple3*) or between background objects, the observed qualitative improvements are smaller. In case of GT OF, the MSEs of the naive interpolation are practically not affected by our guided interpolation. In this context, it is important to note that the performed depth order guided interpolation corrects naive interpolations, if they result in a perceptual incoherency, to the closest valid disparity (or, for data from [27], depth value). Thus, they often lead to quantitatively similar results. Hence, in Section 5.3.2 we focus on visual comparisons of our algorithm between the different interpolation techniques mentioned above as opposed to quantitative comparisons.

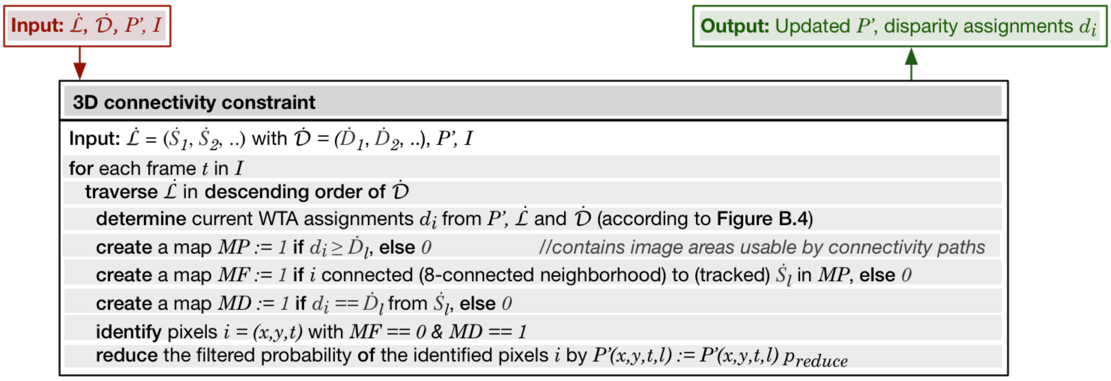


Input: $\dot{\mathcal{L}}$, $\dot{\mathcal{D}}$, $\dot{P}'$, I

Output: Segmentation $\mathcal{R}$, disparity assignments d_i

```
Disparity assigment
Input: $\dot{\mathcal{L}} = (\dot{S}_1, \dot{S}_2, ..)$ with $\dot{\mathcal{D}} = (\dot{D}_1, \dot{D}_2, ..)$, $P'$, $I$
for each pixel $i = (x,y,t)$ in $I$
    determine the $l$ for which $i$ has the highest probability in $P'(x,y,t,l)$
    assign $i$ to the segment $\mathcal{R}_l$ to obtain a segmentation $\mathcal{R}$
    if WTA is used
        assign $d_i := \dot{D}_l$ according to Equation (5.8)
    else if DB is used
        determine the $n$ $l$ with the highest probabilities in $P'(x,y,t,l)$
        average the corresponding $\dot{D}_l$ to assign $d_i$ according to Equation (5.9)
```

Figure B.4: Program flow for disparity assignment (WTA or DB). The disparity d_i of a pixel $i = (x, y, t)$ in the input video I can be assigned using one of two assignment schemes (*WTA* or *DB*). Both schemes determine the l with the largest probability in the filtered cost volume $P'(x, y, t, l)$ for i and accordingly assign i to segment $R_l \in \mathcal{R}$ corresponding to scribble $\dot{S}_l \in \dot{\mathcal{L}}$. WTA further assigns i's d_i to a disparity $\dot{D}_l \in \dot{\mathcal{D}}$ that corresponds to $\dot{S}_l$. DB averages the n $\dot{D}_l$ corresponding to the n l with the largest $P'(x, y, t, l)$ for i to obtain i's d_i.



Input: $\dot{\mathcal{L}}$, $\dot{\mathcal{D}}$, P', I

Output: Updated P', disparity assignments d_i

```
3D connectivity constraint
Input: $\dot{\mathcal{L}} = (\dot{S}_1, \dot{S}_2, ..)$ with $\dot{\mathcal{D}} = (\dot{D}_1, \dot{D}_2, ..)$, $P'$, $I$
for each frame $t$ in $I$
    traverse $\dot{\mathcal{L}}$ in descending order of $\dot{\mathcal{D}}$
        determine current WTA assignments $d_i$ from $P'$, $\dot{\mathcal{L}}$ and $\dot{\mathcal{D}}$ (according to Figure B.4)
        create a map $MP := 1$ if $d_i \geq \dot{D}_l$, else $0$          //contains image areas usable by connectivity paths
        create a map $MF := 1$ if $i$ connected (8-connected neighborhood) to (tracked) $\dot{S}_l$ in $MP$, else $0$
        create a map $MD := 1$ if $d_i == \dot{D}_l$ from $\dot{S}_l$, else $0$
        identify pixels $i = (x,y,t)$ with $MF == 0$ & $MD == 1$
        reduce the filtered probability of the identified pixels $i$ by $P'(x,y,t,l) := P'(x,y,t,l)\, p_{reduce}$
```

Figure B.5: Program flow for CON. If, in the current WTA disparity map for frame t of the input video I, a pixel $i = (x, y, t)$'s disparity d_i (e.g., $\dot{D}_l \in \dot{\mathcal{D}}$) is not spatially connected to its corresponding (tracked) scribble (e.g., $\dot{S}_l \in \dot{\mathcal{L}}$), p_{reduce} reduces the filtered probability $P'(x, y, t, l)$ of i. A connectivity path in the current WTA disparity map consists of neighboring pixels i that are assigned to the same or a larger disparity d_i than the disparity $\dot{D}_l$ of the currently observed scribble $\dot{S}_l$. p_{reduce} is an appropriate factor to reach a value close to zero.

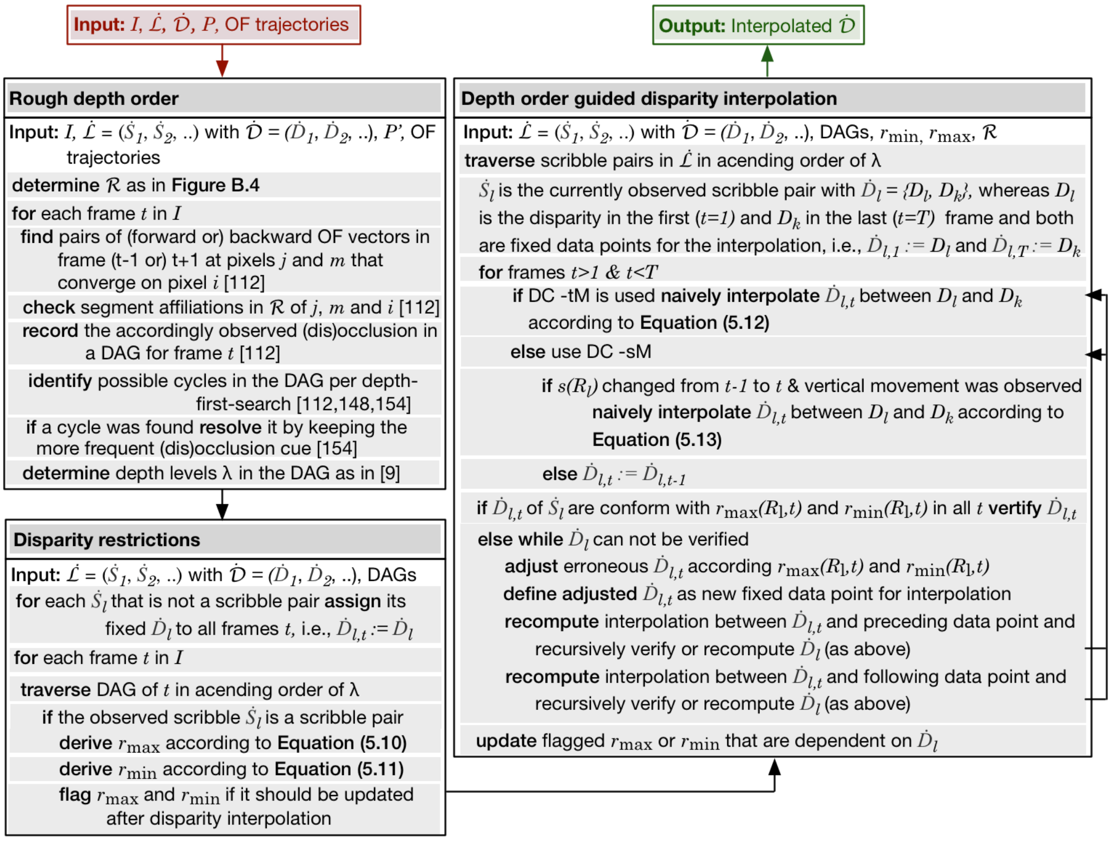

Input: I, $\dot{\mathcal{L}}$, $\dot{\mathcal{D}}$, P, OF trajectories
Output: Interpolated $\dot{\mathcal{D}}$
Rough depth order
Input: I, $\dot{\mathcal{L}} = (\dot{S}_1, \dot{S}_2, ..)$ with $\dot{\mathcal{D}} = (\dot{D}_1, \dot{D}_2, ..)$, P', OF trajectories
determine $\mathcal{R}$ as in **Figure B.4**
for each frame t in I
find pairs of (forward or) backward OF vectors in frame (t-1 or) t+1 at pixels j and m that converge on pixel i [112]
check segment affiliations in $\mathcal{R}$ of j, m and i [112]
record the accordingly observed (dis)occlusion in a DAG for frame t [112]
identify possible cycles in the DAG per depth-first-search [112,148,154]
if a cycle was found **resolve** it by keeping the more frequent (dis)occlusion cue [154]
determine depth levels λ in the DAG as in [9]
Disparity restrictions
Input: $\dot{\mathcal{L}} = (\dot{S}_1, \dot{S}_2, ..)$ with $\dot{\mathcal{D}} = (\dot{D}_1, \dot{D}_2, ..)$, DAGs
for each $\dot{S}_l$ that is not a scribble pair **assign** its fixed $\dot{D}_l$ to all frames t, i.e., $\dot{D}_{l,t} := \dot{D}_l$
for each frame t in I
traverse DAG of t in acending order of λ
if the observed scribble $\dot{S}_l$ is a scribble pair
derive $r_{\max}$ according to **Equation (5.10)**
derive $r_{\min}$ according to **Equation (5.11)**
flag $r_{\max}$ and $r_{\min}$ if it should be updated after disparity interpolation
Depth order guided disparity interpolation
Input: $\dot{\mathcal{L}} = (\dot{S}_1, \dot{S}_2, ..)$ with $\dot{\mathcal{D}} = (\dot{D}_1, \dot{D}_2, ..)$, DAGs, $r_{\min}$, $r_{\max}$, $\mathcal{R}$
traverse scribble pairs in $\dot{\mathcal{L}}$ in acending order of λ
$\dot{S}_l$ is the currently observed scribble pair with $\dot{D}_l = \{D_l, D_k\}$, whereas D_l is the disparity in the first ($t=1$) and D_k in the last ($t=T$) frame and both are fixed data points for the interpolation, i.e., $\dot{D}_{l,1} := D_l$ and $\dot{D}_{l,T} := D_k$
for frames $t>1$ & $t<T$
if DC -tM is used **naively interpolate** $\dot{D}_{l,t}$ between D_l and D_k according to **Equation (5.12)**
else use DC -sM
if $s(R_l)$ changed from $t\text{-}1$ to t & vertical movement was observed
naively interpolate $\dot{D}_{l,t}$ between D_l and D_k according to **Equation (5.13)**
else $\dot{D}_{l,t} := \dot{D}_{l,t-1}$
if $\dot{D}_{l,t}$ of $\dot{S}_l$ are conform with $r_{\max}(R_l,t)$ and $r_{\min}(R_l,t)$ in all t **verify** $\dot{D}_{l,t}$
else while $\dot{D}_l$ can not be verified
adjust erroneous $\dot{D}_{l,t}$ according $r_{\max}(R_l,t)$ and $r_{\min}(R_l,t)$
define adjusted $\dot{D}_{l,t}$ as new fixed data point for interpolation
recompute interpolation between $\dot{D}_{l,t}$ and preceding data point and recursively verify or recompute $\dot{D}_l$ (as above)
recompute interpolation between $\dot{D}_{l,t}$ and following data point and recursively verify or recompute $\dot{D}_l$ (as above)
update flagged $r_{\max}$ or $r_{\min}$ that are dependent on $\dot{D}_l$

Figure B.6: Program flow for DC -tM and DC -sM. Motion-caused (dis)occlusions are detected and investigated to determine a rough depth order between segments $R_l \in \mathcal{R}$ in each frame $t \in [1, T]$ of video I [112]. (Dis)occlusion cues from t are stored in a DAG for t. From a DAG the depth level $\lambda(R_l, t)$ of segments R_l and corresponding scribbles $\dot{S}_l \in \dot{\mathcal{L}}$ in t can be derived. DAGs are also used to derive disparity restrictions $r_{\min}(R_l, t)$ and $r_{\max}(R_l, t)$ for scribble pairs $\dot{S}_l$. These $r_{\min}$ and $r_{\max}$ are used in a depth order guided disparity interpolation (*DC -tM* or *DC -sM*) that is performed for scribble pairs $\dot{S}_l$ and between their different user-selected disparities $\dot{D}_l = \{D_l, D_k\}$ in the first and last frame and obtains the interpolated disparities $\dot{D}_{l,t}$ for each t.

Table B.1: Comparison of different algorithm versions (DB). The table lists the mean squared error (MSE) of results averaged over all pixels multiplied by 100. Our algorithm was applied with ground truth (GT) optical flow (OF) (*top*) and with estimated OF (*bottom*). The endpoint error (EE) measures the accuracy of the OF (*right*). Our algorithm is tested in different versions, in which different components are en- or disabled.

MSE	DB, using GT OF										**EE**
	MS		**STC**		**CON**		**DC -tM**		**DC -sM**		**OF**
		+TC		**+TC**		**+TC**		**+TC**		**+TC**	
Alley1	0.07	0.07	0.05	0.04	0.05	0.05	0.05	0.05	0.05	0.05	0.00
Ambush2	1.03	0.34	1.00	0.32	1.00	0.32	1.00	0.31	1.00	0.31	0.00
Ambush5	1.02	1.02	0.78	0.78	0.80	0.80	0.77	0.78	0.77	0.78	0.00
Ambush7	0.76	0.76	0.43	0.43	0.45	0.45	0.42	0.43	0.42	0.43	0.00
Shaman2	1.18	1.19	0.90	0.89	0.89	0.88	0.88	0.89	0.89	0.89	0.00
Shaman3	1.84	1.85	1.86	1.86	1.87	1.88	0.47	0.45	0.57	0.56	0.00
Sleeping1	3.68	3.68	3.61	3.61	3.60	3.60	0.68	0.68	0.79	0.78	0.00
Temple3	0.39	0.33	0.42	0.29	0.39	0.29	0.40	0.29	0.39	0.29	0.00

MSE	DB, using estimated OF										**EE**
	MS		**STC**		**CON**		**DC -tM**		**DC -sM**		**OF**
		+TC		**+TC**		**+TC**		**+TC**		**+TC**	
Alley1	0.06	0.06	0.04	0.04	0.04	0.04	0.05	0.04	0.05	0.04	1.73
Ambush2	1.06	0.31	1.15	0.31	1.15	0.31	1.04	0.37	1.04	0.37	73.68
Ambush5	1.04	1.05	0.74	0.74	0.76	0.76	0.74	0.74	0.73	0.74	7.12
Ambush7	0.74	0.74	0.41	0.41	0.40	0.40	0.42	0.42	0.42	0.42	2.20
Shaman2	1.20	1.21	0.92	0.92	0.89	0.89	0.89	0.89	0.89	0.89	0.60
Shaman3	1.81	1.81	1.81	1.81	1.83	1.83	0.21	0.25	0.30	0.33	1.20
Sleeping1	3.76	3.76	3.68	3.68	3.68	3.68	0.72	0.72	0.72	0.72	2.20
Temple3	0.40	0.30	0.42	0.27	0.41	0.27	0.37	0.26	0.37	0.26	13.29

Table B.2: Quantitative comparison of naive interpolation and proposed depth order guided interpolation (GT OF, WTA). The table lists the mean squared error (MSE) of our results averaged over all pixels when applying our algorithm with ground truth (GT) optical flow (OF) vectors. The given MSEs are multiplied by 100.

MSE	depth change with **naive interpolation**							
	Alley1	*Ambush2*	*Ambush5*	*Ambush7*	*Shaman2*	*Shaman3*	*Sleeping1*	*Temple3*
tM	0.04	1.16	0.64	0.49	0.39	0.18	0.57	0.27
sM	0.04	1.16	0.64	0.55	0.42	0.23	0.64	0.27
tM+TC	0.04	0.23	0.63	0.49	0.39	0.18	0.58	0.21
sM+TC	0.04	0.23	0.63	0.55	0.42	0.23	0.65	0.21

MSE	depth change with **depth order guided interpolation**							
	Alley1	*Ambush2*	*Ambush5*	*Ambush7*	*Shaman2*	*Shaman3*	*Sleeping1*	*Temple3*
tM	0.04	1.16	0.64	0.47	0.41	0.41	0.48	0.27
sM	0.04	1.16	0.64	0.47	0.42	0.51	0.56	0.27
tM+TC	0.04	0.23	0.64	0.48	0.41	0.40	0.48	0.21
sM+TC	0.04	0.23	0.64	0.48	0.42	0.49	0.57	0.21

Table B.3: Quantitative comparison of naive interpolation and proposed depth order guided interpolation (estimated OF, WTA). The table lists the mean squared error (MSE) of the depth values averaged over all pixels when applying our algorithm with estimated optical flow (OF) vectors. The given MSEs are multiplied by 100.

MSE	depth change with **naive interpolation**							
	Alley1	*Ambush2*	*Ambush5*	*Ambush7*	*Shaman2*	*Shaman3*	*Sleeping1*	*Temple3*
tM	0.05	1.23	1.43	0.49	0.40	0.30	0.57	0.27
sM	0.04	1.23	1.43	0.49	0.40	0.41	0.65	0.26
tM+TC	0.05	0.27	1.44	0.51	0.40	0.30	0.58	0.18
sM+TC	0.04	0.27	1.44	0.51	0.40	0.41	0.67	0.18

MSE	depth change with **depth order guided interpolation**							
	Alley1	*Ambush2*	*Ambush5*	*Ambush7*	*Shaman2*	*Shaman3*	*Sleeping1*	*Temple3*
tM	0.05	1.23	0.67	0.47	0.39	0.30	0.56	0.28
sM	0.05	1.23	0.66	0.47	0.40	0.40	0.60	0.28
tM+TC	0.05	0.28	0.66	0.47	0.38	0.34	0.56	0.15
sM+TC	0.05	0.28	0.66	0.47	0.39	0.42	0.61	0.15

Bibliography

[1] A. Agarwala, A. Hertzmann, D. H. Salesin, and S. M. Seitz. Keyframe-based tracking for rotoscoping and animation. In *SIGGRAPH'04*, pages 584–591, 2004.

[2] S. Aguirre and R. M. Rodriguez-Dagnino. Synthesizing stereo 3D views from focus cues in monoscopic 2D images. In *EI'03: Electronic Imaging, Stereoscopic Displays and Virtual Reality Systems X*, pages 377–388, 2003.

[3] J. Bai and X. Wu. Error-tolerant scribbles based interactive image segmentation. In *CVPR'14: Computer Vision and Pattern Recognition*, pages 392–399, 2014.

[4] X. Bai and G. Sapiro. Geodesic Matting: A framework for fast interactive image and video segmentation and matting. *International Journal of Computer Vision*, 82(2):113–132, 2009.

[5] X. Bai, J. Wang, D. Simons, and G. Sapiro. Video SnapCut: Robust video object cutout using localized classifiers. In *SIGGRAPH'09*, pages 70:1–70:11, 2009.

[6] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. In *ICCV'07: International Conference on Computer Vision*, pages 1–8, 2007.

[7] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011.

[8] D. Batra, A. Kowdle, D. Parikh, J. Luo, and T. Chen. Interactively co-segmentating topically related images with intelligent scribble guidance. *International Journal of Computer Vision*, 93(3):273–292, 2011.

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

[10] Blender. http://www.blender.org/. Accessed: 2014-05-28.

[11] M. Bleyer. *Segmentation-based Stereo and Motion with Occllusions*. PhD thesis, Vienna University of Technology, Institute of Software Technology and Interactive Systems, 2006.

[12] M. Bleyer and M. Gelautz. Simple but effective tree structures for dynamic programming-based stereo matching. In *VISAPP'08: International Conference on Computer Vision Theory and Applications*, pages 415–422, 2008.

[13] M. Bleyer, M. Gelautz, C. Rother, and C. Rhemann. A stereo approach that handles the matting problem via image warping. In *CVPR:'09: Conference on Computer Vision and Pattern Recognition*, pages 501–508, 2009.

[14] M. Bleyer, C. Rother, P. Kohli, D. Scharstein, and S. Sinha. Object Stereo - Joint stereo matching and object segmentation. In *CVPR'11: Conference on Computer Vision and Pattern Recognition*, pages 3081–3088, 2011.

[15] A. Bokov, D. Vatolin, A. Zachesov, A. Belous, and M. Erofeev. Automatic detection of artifacts in converted S3D video. In *EI'14: Electronic Imaging, Stereoscopic Displays and Applications XXV*, pages 1–14, 2014.

[16] J. S. Boreczky and L. A. Rowe. Comparison of video shot boundary detection techniques. *Journal of Electronic Imaging*, 5(2):122–128, 1996.

[17] G. Borgefors. Distance transformations in digital images. *Computer Vision, Graphics, and Image Processing*, 34(3):344–371, 1986.

[18] Y. Boykov and G. Funka-Lea. Graph cuts and efficient N-D image segmentation. *International Journal of Computer Vision*, 70(2):109–131, 2006.

[19] Y. Boykov and M. P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *ICCV'01: International Conference on Computer Vision*, pages 105–112, 2001.

[20] Broadband Network & Digital Media Lab. Test sequences. http://media.au.tsinghua.edu.cn/2Dto3D/testsequence.html. Accessed: 2013-10-22, 2013.

[21] N. Brosch, A. Hosni, G. Ramachandran, L. He, and M. Gelautz. Content generation for 3D video/TV. *e & i Elektrotechnik und Informationstechnik*, 128(10):359–365, 2011.

[22] N. Brosch, A. Hosni, C. Rhemann, and M. Gelautz. Spatio-temporally coherent interactive video object segmentation via efficient filtering. In *DAGM/ÖAGM'12: Joint Symposium of the German Association for Pattern Recognition and the Austrian Association for Pattern Recognition*, pages 418–427, 2012.

[23] N. Brosch, M. Nezveda, M. Gelautz, and F. Seitner. Efficient quality enhancement of disparity maps based on alpha matting. In *EI'14: Electronic Imaging, Stereoscopic Displays and Applications XXV*, pages 1–10, 2014.

[24] N. Brosch, C. Rhemann, and M. Gelautz. Segmentation-based depth propagation in videos. In *ÖAGM'11: Austrian Association for Pattern Recognition Workshop*, pages 1–8, 2011.

[25] N. Brosch, T. Schausberger, and M. Gelautz. Towards perceptually coherent depth maps in 2D-to-3D conversion. In *EI'16: Electronic Imaging, Stereoscopic Displays and Applications XXVII*, pages 1–11, 2016.

[26] T. Brox and J. Malik. Object segmentation by long term analysis of point trajectories. In *ECCV'10: European Conference on Computer Vision: Part V*, pages 282–295, 2010.

[27] D. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *ECCV'12: European Conference on Computer Vision*, pages 611–625, 2012.

[28] X. Cao, Z. Li, and Q. Dai. Semi-automatic 2D-to-3D conversion using disparity propagation. *Transactions on Broadcasting*, 57(2):491–499, 2011.

[29] J. Chen, J. Zhao, X. Wang, C. Huang, E. Dong, B. Chen, and Z. Yuan. A simple semi-automatic technique for 2D to 3D video conversion. In *AICI'09: Artificial Intelligence and Computational Intelligence*, pages 336–343, 2011.

[30] X. Chen, D. Zou, J. Li, X. Cao, Q. Zhao, and H. Zhang. Sparse dictionary learning for edit propagation of high-resolution images. In *CVPR'14: Conference on Computer Vision and Pattern Recognition*, pages 2854–2861, 2014.

[31] J. W. Choi and T. K. Whangbo. A key frame-based depth propagation for semi-automatic 2D-to-3D video conversion using a pair of bilateral filters. *International Journal of Digital Content Technology*, 7(17):94–103, 2013.

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

[33] C. Cotsaces, N. Nikolaidis, and I. Pitas. Video shot detection and condensed representation. A review. *Signal Processing Magazine*, 23(2):28–37, 2006.

[34] T. Cour, F. Benezit, and J. Shi. Spectral segmentation with multiscale graph decomposition. In *CVPR'05: Computer Vision and Pattern Recognition*, pages 1124–1131, 2005.

[35] F. C. Crow. Summed-area tables for texture mapping. In *SIGGRAPH'84*, pages 207–212, 1984.

[36] D. DeMenthon, X. Munoz, D. Raba, J. Marti, and X. Cufi. Spatio-temporal segmentation of video by hierarchical mean shift analysis. In *SMVP'02: Statistical Methods in Video Processing Workshop*, pages 142–151, 2002.

[37] X. Ding, Y. Xu, L. Deng, and X. Yang. Colorization using quaternion algebra with automatic scribble generation. In *MMM'12: Advances in Multimedia Modeling*, pages 103–114, 2012.

[38] D. Donatsch, N. Farber, and M. Zwicker. 3D conversion using vanishing points and image warping. In *3DTV-CON'13: 3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video*, pages 1–4, 2013.

[39] Z. Farbman, F. Fattal, D. Lischinski, and R. Szeliski. Edge-preserving decompositions for multi-scale tone and detail manipulation. In *SIGGRAPH'08*, pages 67:1–67:10, 2008.

[40] Z. Farbman, R. Fattal, and D. Lischinski. Diffusion maps for edge-aware image editing. In *SIGGRAPH Asia'10*, pages 145:1–145:10, 2010.

[41] M. Fawaz, R. Phan, R. Rzeszutek, and D. Androutsos. Adaptive 2D to 3D image conversion using a hybrid graph cuts and random walks approach. In *ICASSP'12: International Conference on Acoustics, Speech and Signal Processing*, pages 1441–1444, 2012.

[42] C. Fehn, K. Schüür, I. Feldmann, P. Kauff, and A. Smolic. Distribution of ATTEST test sequences for EE4 in MPEG 3DAV. MPEG Meeting, 2002.

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

[44] J. Feng, H. Ma, J. Hu, L. Cao, and H. Zhang. Superpixel based depth propagation for semi-automatic 2D-to-3D video conversion. In *ICNDC'12: International Conference on Networking and Distributed Computing*, pages 157–160, 2012.

[45] O. P. Gangwal and R. P. Berretty. Depth map post-processing for 3D-TV. In *ICCE'09: International Conference on Consumer Electronics*, pages 1–2, 2009.

[46] E. S. L. Gastal and M. Oliveira. Domain transform for edge-aware image and video processing. In *SIGGRAPH'11*, pages 69:1–69:12, 2011.

[47] S. Ghuffar, N. Brosch, N. Pfeifer, and M. Gelautz. Motion segmentation in videos from time of flight cameras. In *IWSSIP'12: Conference on International Systems, Signals and Image Processing*, pages 328–332, 2012.

[48] S. Ghuffar, N. Brosch, N. Pfeifer, and M. Gelautz. Motion estimation and segmentation in depth and intensity videos. *Integrated Computer-Aided Engineering*, 21(3):203–218, 2014.

[49] S. Goferman, L. Zelnik-Manor, and A. Tal. Context-aware saliency detection. *Transactions on Pattern Analysis and Machine Intelligence*, 34(10):1915–1926, 2012.

[50] G. H. Golub and C. F. Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

[51] L. Grady. Random walks for image segmentation. *Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1768–1783, 2006.

[52] D. M. Greig, B. T. Porteous, and A. H. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society*, 51(2):271–279, 1989.

[53] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph-based video segmentation. In *CVPR'10: Conference on Computer Vision and Pattern Recognition*, pages 1–14, 2010.

[54] G. Guo, N. Zhang, L. Huo, and W. Gao. 2D to 3D conversion based on edge defocus and segmentation. In *ICASSP'08: International Conference on Acoustics, Speech and Signal Processing*, pages 2181–2184, 2008.

[55] Z. Guo, H. Wang, K. Li, Y. Zhang, X. Wang, and Q. Dai. A novel edit propagation algorithm via L0 gradient minimization. In *PCM'15: Pacific-Rim Conference on Advances in Multimedia Information Processing*, pages 402–410, 2015.

[56] M. Guttmann, L. Wolf, and D. Cohen-Or. Semi-automatic stereo extraction from video footage. In *ICCV'09: International Conference on Computer Vision*, pages 136–142, 2009.

[57] K. Han and K. Hong. Geometric and texture cue based depth-map estimation for 2D to 3D image conversion. In *ICCE'11: International Conference on Consumer Electronics*, pages 651–652, 2011.

[58] P. V. Harman, J. Flack, S. Fox, and M. Dowley. Rapid 2D-to-3D conversion. In *EI'02: Electronic Imaging, Stereoscopic Displays and Virtual Reality Systems IX*, pages 78–86, 2002.

[59] K. He, J. Sun, and X. Tang. Guided image filtering. In *ECCV'10: European Conference on Computer Vision*, pages 1–14, 2010.

[60] R. Hebbalaguppe, K. McGuinness, J. Kuklyte, G. Healy, N. O'Connor, and A. Smeaton. How interaction methods affect image segmentation: User experience in the task. In *UCCV'13: Workshop on User-Centred Computer Vision*, pages 19–24, 2013.

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

[62] A. Hosni, C. Rhemann, M. Bleyer, and M. Gelautz. Temporally consistent disparity and optical flow via efficient spatio-temporal filtering. In *PSIVT'11: Pacific-Rim Symposium on Image and Video Technology*, pages 165–177, 2011.

[63] W. Hu, Z. Dong, and G. D. Yuan. Edit propagation via edge-aware filtering. *Journal of Computer Science and Technology*, 27(4):830–840, 2012.

[64] W. Huang, X. Cao, K. Lu, Q. Dai, and A. C. Bovik. Toward naturalistic 2D-to-3D conversion. *Transactions on Image Processing*, 24(2):724–733, 2015.

[65] I. Ideses, L. Yaroslavsky, and B. Fishbain. Real-time 2D to 3D video conversion. *Journal of Real-Time Image Processing*, 2(1):3–9, 2007.

[66] S. Iizuka, Y. Endo, Y. Kanamori, J. Mitani, and Y. Fukui. Efficient depth propagation for constructing a layered depth image from a single image. *Computer Graphics Forum*, 33(7):279–288, 2014.

[67] M. Ivancics, N. Brosch, and M. Gelautz. Efficient depth propagation in videos with GPU-acceleration. In *VCIP'14: Visual Communications and Image Processing*, pages 1–4, 2014.

[68] M. Ivancsics. Effiziente Tiefenpropagierung in Videos mit GPU-Unterstützung. Master's thesis, Vienna University of Technology, Institute of Software Technology and Interactive Systems, 2014.

[69] L. Jiangbo, S. Keyang, M. Dongbo, L. Liang, and M. N. Do. Cross-based local multipoint filtering. In *CVPR'12: Conference on Computer Vision and Pattern Recognition*, pages 430–437, 2012.

[70] Y. J. Jung, A. Baik, J. Kim, and D. Park. A novel 2D-to-3D conversion technique based on relative height-depth cue. In *EI'09: Electronic Imaging, Stereoscopic Displays and Applications XX*, pages 1–8, 2009.

[71] K. Karsch, C. Liu, and S. Kang. Depth extraction from video using non-parametric sampling. In *ECCV'12: European Conference on Computer Vision*, pages 775–788, 2012.

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

[73] P. Kellnhofer, T. Leimkühler, T. Ritschel, K. Myszkowski, and H. P. Seidel. What makes 2D-to-3D stereo conversion perceptually plausible? In *SAP'15: Symposium on Applied Perception*, pages 59–66, 2015.

[74] J. Kim, A. Baik, Y. J. Jung, and D. Park. 2D-to-3D conversion by using visual attention analysis. In *EI'10: Electronic Imaging, Stereoscopic Displays and Applications XXI*, pages 1–12, 2010.

[75] J. J. Koenderink, A. J. van Doorn, A. M. Kappers, and J. T. Todd. Ambiguity and the 'mental eye' in pictorial relief. *Perception*, 30(4):431–448, 2001.

[76] P. Kohli, H. Nickisch, C. Rother, and C. Rhemann. User-centric learning and evaluation of interactive segmentation systems. *International Journal of Computer Vision*, 100(3):261–274, 2012.

[77] V. Kolmogorov, A. Criminisi, A. Blake, G. Cross, and C. Rother. Bi-layer segmentation of binocular stereo video. In *CVPR'05: Computer Vision and Pattern Recognition*, pages 407–414, 2005.

[78] J. Kopf, M. F. Cohen, D. Lischinski, and M. Uyttendaele. Joint bilateral upsampling. In *SIGGRAPH'07*, 2007.

[79] V. Kramarev, O. Demetz, C. Schroers, and J. Weickert. Cross anisotropic cost volume filtering for segmentation. In *ACCV'12: Asian Conference on Computer Vision*, pages 803–814, 2013.

[80] L. Lam, S. W. Lee, and C. Y. Suen. Thinning methodologies - a comprehensive survey. *Transactions on Pattern Analysis and Machine Intelligence*, 14(9):869–885, 1992.

[81] M. Lang, A. Hornung, O. Wang, S. Poulakos, A. Smolic, and M. Gross. Nonlinear disparity mapping for stereoscopic 3D. In *SIGGRAPH'10*, pages 75:1–75:10, 2010.

[82] M. Lang, O. Wang, T. Aydin, A. Smolic, and M. Gross. Practical temporal consistency for image-based graphics applications. *Transactions on Graphics*, 31(4):34:1–34:8, 2012.

[83] S. Y. Lee, J. C. Yoon, and I. K. Lee. Temporally coherent video matting. *Graphical Models*, 72(3):25–33, 2010.

[84] A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. In *SIGGRAPH'04*, pages 689–694, 2004.

[85] A. Levin, D. Lischinski, and Y. Weiss. A closed-form solution to natural image matting. *Transactions on Pattern Analysis and Machine Intelligence*, 30(2):228–242, 2008.

[86] J. Lezama, K. Alahari, J. Sivic, and I. Laptev. Track to the future: Spatio-temporal video segmentation with long-range motion cues. In *CVPR'11: Computer Vision and Pattern Recognition*, pages 3369–3376, 2011.

[87] Y. Li, E. Adelson, and A. Agarwala. ScribbleBoost: Adding classification to edge-aware interpolation of local image and video adjustments. In *EGSR'08: Eurographics Conference on Rendering*, pages 1255–1264, 2008.

[88] Y. Li, J. Sun, and H. Y. Shum. Video object cut and paste. In *SIGGRAPH'05*, pages 595–600, 2005.

[89] Y. Li, J. Sun, C. K. Tang, and H. Y. Shum. Lazy snapping. In *SIGGRAPH'04*, pages 303–308, 2004.

[90] Z. Li, X. Cao, and Q. Dai. A novel method for 2D-to-3D video conversion using bi-directional motion estimation. In *ICASSP'12: International Conference on Acoustics, Speech and Signal Processing*, pages 1429–1432, 2012.

[91] Z. Li, X. Xie, and X. Liu. An efficient 2D to 3D video conversion method based on skeleton line tracking. In *3DTV-CON'09: Conference on The True Vision-Capture, Transmission and Display of 3D Video*, pages 1–4, 2009.

[92] M. Liao, J. Gao, R. Yang, and M. Gong. Video Stereolization: Combining motion analysis with user interaction. *Transactions on Visualization and Computer Graphics*, 18(7):1079–1088, 2012.

[93] W. N. Lie, C. Y. Chen, and W. C. Chen. 2D to 3D video conversion with key-frame depth propagation and trilateral filtering. *Electronics Letters*, 47(5):319–321, 2011.

[94] G. Lin, J. Huang, and W. Lie. Semi-automatic 2D-to-3D video conversion based on depth propagation from key-frames. In *ICIP'13: International Conference on Image Processing*, pages 2202–2206, 2013.

[95] D. Lischinski, Z. Farbman, M. Uyttendaele, and R. Szeliski. Interactive local adjustment of tonal values. In *SIGGRAPH'06*, pages 646–653, 2006.

[96] 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.

[97] J. Liu, J. Sun, and H. Y. Shum. Paint selection. In *SIGGRAPH'09*, pages 1–7, 2009.

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

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

[100] L. Q. Ma and K. Xu. Efficient manifold preserving edit propagation with adaptive neighborhood size. *Computers and Graphics*, 38:167–173, 2014.

[101] D. Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. Henry Holt and Co., Inc., 1982.

[102] S. Martull, M. P. Martorell, and K. Fukui. Realistic CG stereo image dataset with ground truth disparity maps. In *ICPR20'12: International Conference on Pattern Recognition*, pages 1038–1042, 2012.

[103] Microsoft Kinect. http://www.microsoft.com/en-us/kinectforwindows/. Accessed: 2014-05-28.

[104] E. N. Mortensen and W. A. Barrett. Intelligent scissors for image composition. In *SIGGRAPH'95*, pages 191–198, 1995.

[105] K. Moustakas, D. Tzovaras, and M. G. Strintzis. Stereoscopic video generation based on efficient layered structure and motion estimation from a monoscopic image sequence. *Transactions on Circuits and Systems for Video Technology*, 15:1065–1073, 2005.

[106] M. Nezveda, N. Brosch, F. Seitner, and M. Gelautz. Depth map post-processing for depth-image-based rendering: A user study. In *EI'14: Electronic Imaging, Stereoscopic Displays and Applications XXV*, pages 1–9, 2014.

[107] G. Noris, D. Sykora, A. Shamir, S. Coros, B. Whited, M. Simmons, A. Hornung, M. Gross, and R. Sumner. Smart scribbles for sketch segmentation. *Computer Graphics Forum*, 31(8):2516–2527, 2012.

[108] Nvidia. CUDA: Compute unified device architecture programming guide. Technical report, 2008.

[109] P. Ochs and T. Brox. Object segmentation in video: A hierarchical variational approach for turning point trajectories into dense regions. In *ICCV'11: International Conference on Computer Vision*, pages 1583–1590, 2011.

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

[111] T. Okino, H. Murata, K. Taima, T. Iinuma, and K. Oketani. New television with 2D/3D image conversion technologies. In *EI'96: Electronic Imaging, Stereoscopic Displays and Virtual Reality Systems*, pages 96–103, 1996.

[112] G. Palou and P. Salembier. Depth order estimation for video frames using motion occlusions. *Computer Vision*, 8(2):152–160, 2013.

[113] S. Paris. Edge-preserving smoothing and mean-shift segmentation of video streams. In *ECCV'08: European Conference on Computer Vision: Part II*, pages 460–473, 2008.

[114] S. Paris and F. Durand. A topological approach to hierarchical segmentation using mean shift. In *CVPR'07: Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[115] G. Petschnigg, R. Szeliski, M. Agrawala, M. Cohen, H. Hoppe, and K. Toyama. Digital photography with flash and no-flash image pairs. In *SIGGRAPH'04*, volume 23, pages 664–672, 2004.

[116] R. Phan and D. Androutsos. A semi-automatic 2D to stereoscopic 3D image and video conversion system in a semi-automated segmentation perspective. In *EI'13: Electronic Imaging, Stereoscopic Displays and Applications XXIV*, pages 1–12, 2013.

[117] R. Phan and D. Androutsos. Robust semi-automatic depth map generation in unconstrained images and video sequences for 2D to stereoscopic 3D conversion. *Transactions on Multimedia*, 16(1):122–136, 2014.

[118] R. Phan, R. Rzeszutek, and D. Androutsos. Semi-automatic 2D to 3D image conversion using scale-space random walks and a graph cuts based depth prior. In *ICIP'11: International Conference on Image Processing*, pages 865–868, 2011.

[119] R. Phan, R. Rzeszutek, and D. Androutsos. *Multimedia Image and Video Processing*, chapter Literature survey on recent methods for 2D to 3D video conversion, pages 691–716. CRC Press, 2nd edition, 2012.

[120] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[121] R. Poppe. A survey on vision-based human action recognition. *Image and Vision Computing*, 28(6):976–990, 2010.

[122] B. L. Price, B. S. Morse, and S. Cohen. LIVEcut: Learning-based interactive video segmentation by evaluation of multiple propagated cues. In *ICCV'09: International Conference on Computer Vision*, pages 779–786, 2009.

[123] S. J. D. Prince. *Computer Vision: Models, Learning, and Inference*. Cambridge University Press, 1st edition, 2012.

[124] F. Remondino and D. Stoppa. *ToF range-imaging cameras*. Springer, 2012.

[125] C. Rhemann, A. Hosni, M. Bleyer, C. Rother, and M. Gelautz. Fast cost-volume filtering for visual correspondence and beyond. In *CVPR'11: Conference on Computer Vision and Pattern Recognition*, pages 3017–3024, 2011.

[126] C. Richardt, D. Orr, I. Davies, A. Criminisi, and N. A. Dodgson. Real-time spatiotemporal stereo matching using the dual-cross-bilateral grid. In *ECCV'10: European Conference on Computer Vision*, pages 510–523, 2010.

[127] C. Rother, V. Kolmogorov, and A. Blake. GrabCut: Interactive foreground extraction using iterated graph cuts. In *SIGGRAPH'04*, pages 309–314, 2004.

[128] C. Rusu and S. A. Tsaftaris. Estimation of scribble placement for painting colorization. In *ISPA'13: International Symposium on Image and Signal Processing and Analysis*, pages 564–569, 2013.

[129] R. Rzeszutek and D. Androutsos. Label propagation through edge-preserving filters. In *ICASSP'14: International Conference on Acoustics, Speech and Signal Processing*, pages 599–603, 2014.

[130] R. Rzeszutek, R. Phan, and D. Androutsos. Semi-automatic synthetic depth map generation for video using random walks. In *ICME'11: International Conference on Multimedia and Expo*, pages 1–6, 2011.

[131] P. Sand and S. Teller. Particle video: Long-range motion estimation using point trajectories. In *CVPR'06: Conference on Computer Vision and Pattern Recognition*, pages 2195–2202, 2006.

[132] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nešić, X. Wang, and P. Westling. High-resolution stereo datasets with subpixel-accurate ground truth. In *GCPR'14: German Conference Pattern Recognition*, pages 31–42, 2014.

[133] D. Scharstein and C. Pal. Learning conditional random fields for stereo. In *CVPR'07: Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[134] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47:7–42, 2002.

[135] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. In *CVPR'03: Computer Vision and Pattern Recognition*, pages 195–202, 2003.

[136] T. Schausberger. Temporally coherent cost volume filtering-based depth propagation in videos. Master's thesis, Vienna University of Technology, Institute of Software Technology and Interactive Systems, 2015.

[137] M. Seymour. Art of stereo conversion: 2D to 3D - 2012. https://www.fxguide.com/featured/art-of-stereo-conversion-2d-to-3d-2012/. Accessed: 2014-04-29.

[138] E. Sharon, M. Galun, D. Sharon, R. Basri, and A. Brandt. Hierarchy and adaptivity in segmenting visual scenes. *Nature*, 442(7104):810–813, 2006.

[139] J. Shi and J. Malik. Normalized cuts and image segmentation. *Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[140] Sky 3D. http://www.sky.at/3D. Accessed: 2014-04-25.

[141] A. Smolic, P. Kauff, S. Knorr, A. Hornung, M. Kunter, M. Muller, and M. Lang. Three-dimensional video postproduction and processing. *Proceedings of the IEEE*, 99(4):607–625, 2011.

[142] A. N. Stein, T. S. Stepleton, and M. Hebert. Towards unsupervised whole-object segmentation: Combining automated matting with boundary detection. In *CVPR'08: Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[143] StereoD. http://www.stereodllc.com/. Accessed: 2014-04-29.

[144] Stereoscopic Suite X. http://www.emotion3d.tv Accessed: 2014-08-12.

[145] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by GPU-accelerated large displacement optical flow. In *ECCV'10: European Conference on Computer Vision: Part I*, pages 438–451, 2010.

[146] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag New York, 1st edition, 2010.

[147] W. J. Tam, C. Vázquez, and F. Speranza. Three-dimensional TV: A novel method for generating surrogate depth maps using color information. In *EI'09: Electronic Imaging, Stereoscopic Displays and Applications XX*, pages 1–9, 2009.

[148] R. Tarjan. Depth-first search and linear graph algorithms. In *Journal on Computing*, volume 1, pages 146–160, 1972.

[149] D. Terzopoulos and R. Szeliski. Tracking with kalman snakes. In *Active Vision*, pages 3–20. MIT Press, 1993.

[150] A. N. Tikhonov and V. Y. Arsenin. *Solutions of ill-posed problems*. V. H. Winston & Sons, 1977.

[151] D. A. Tolliver and G. L. Miller. Graph partitioning by spectral rounding: Applications in image segmentation and clustering. In *CVPR'06: Computer Vision and Pattern Recognition*, pages 1053–1060, 2006.

[152] C. Tomasi and T. Kanade. *Detection and Tracking of Point Features*. Shape and motion from image streams. School of Computer Science, Carnegie Mellon Univ., 1991.

[153] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *ICCV'98: International Conference on Computer Vision*, pages 839–846, 1998.

[154] E. Turetken and A. Alatan. Temporally consistent layer depth ordering via pixel voting for pseudo 3D representation. In *3DTV-CON'09: 3DTV Conference: The True Vision-Capture, Transmission and Display of 3D Video*, pages 1–4, 2009.

[155] A. van Doorn, J. Koenderink, and J. Wagemans. Rank order scaling of pictorial depth. *Iperception*, 2(7):724–44, 2011.

[156] C. Varekamp and B. Barenbrug. Improved depth propagation for 2D to 3D video conversion using key-frames. In *IETCVMP'07: European Conference on Visual Media Production*, pages 1–7, 2007.

[157] S. Vicente, V. Kolmogorov, and C. Rother. Graph cut based image segmentation with connectivity priors. In *CVPR'08: Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[158] A. Voronov, D. Vatolin, D. Sumin, V. Napadovsky, and A. Borisov. Methodology for stereoscopic motion-picture quality assessment. In *EI'13: Electronic Imaging, Stereoscopic Displays and Applications XXIV*, pages 1–14, 2013.

[159] D. Wang, J. Liu, Y. Ren, C. Ge, W. Liu, and Y. Li. Depth propagation based on depth consistency. In *WCSP'12: International Conference on Wireless Communications Signal Processing*, pages 1–6, 2012.

[160] J. Wang, P. Bhat, R. A. Colburn, M. Agrawala, and M. F. Cohen. Interactive video cutout. In *SIGGRAPH'05*, pages 585–594, 2005.

[161] J. Wang, B. Thiesson, Y. Xu, and M. Cohen. Image and video segmentation by anisotropic kernel mean shift. In *ECCV'04: European Conference on Computer Vision*, pages 238–249, 2004.

[162] J. Wang, Y. Xu, and M. F. Shum, H. Y. Cohen. Video tooning. In *SIGGRAPH'04*, pages 574–583, 2004.

[163] O. Wang, M. Lang, M. Frei, A. Hornung, A. Smolic, and M. Gross. StereoBrush: Interactive 2D to 3D conversion using discontinuous warps. In *SBIM'11: Eurographics Symposium on Sketch-Based Interfaces and Modeling*, pages 47–54, 2011.

[164] B. Ward, S. Kang, and E. P. Bennett. Depth Director: A System for Adding Depth to Movies. *Computer Graphics and Applications*, 31(1):36–48, 2011.

[165] 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-CON'08: Conference on The True Vision-Capture, Transmission and Display of 3D Video*, pages 65–68, 2008.

[166] J. Wulff, D. J. Butler, G. B. Stanley, and M. J. Black. Lessons and insights from creating a synthetic optical flow benchmark. In *ECCV'12: European Conference on Computer Vision*, pages 168–177, 2012.

[167] L. Xu, Q. Yan, and J. Jia. A sparse control model for image and video editing. *Transactions on Graphics*, 32(6):197:1–197:10, 2013.

[168] P. Xu, H. Fu, O. K. C. Au, and C. L. Tai. Lazy Selection: A scribble-based tool for smart shape elements selection. *Transactions on Graphics*, 31(6):142:1–142:9, 2012.

[169] X. Xu, L. Po, K. Cheung, K. Ng, K. Wong, and C. Ting. Watershed and random walks based depth estimation for semi-automatic 2D to 3D image conversion. In *ICSPCC'12: International Conference on Signal Processing, Communication and Computing*, pages 84–87, 2012.

[170] L. Yatziv and G. Sapiro. Fast image and video colorization using chrominance blending. *Transactions on Image Processing*, 15(5):1120–1129, 2006.

[171] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *Journal of Computing Surveys*, 38, 2006.

[172] YouTube, LLC. https://www.youtube.com. Accessed: 2014-04-25.

[173] J. Zhang and Z. Zhang. Depth map propagation with the texture image guidance. In *ICIP'14: International Conference on Image Processing*, pages 3813–3817, 2014.

[174] L. Zhang, C. Vázquez, and S. Knorr. 3D-TV content creation: Automatic 2D-to-3D video conversion. *Transactions on Broadcasting*, 57(2):372–383, 2011.

[175] Z. Zhang, C. Zhou, B. Xin, Y. Wang, and W. Gao. An interactive system of stereoscopic video conversion. In *MM'12: International Conference on Multimedia*, pages 149–158, 2012.

[176] Avatar. Dir. J. Cameron, *20th Century Fox*. Film, 2009.

[177] Toy Story 2. Dir. J. Lasseter, *Disney Digital 3-D*. Film, 2010.

[178] Star Wars: Episode I. Dir. G. Lucas, *20th Century Fox*. Film, 2012.

NICOLE BROSCH

Home address:
Wallrissstrasse 64/6
1180 Vienna, Austria

E-mail: nicole.brosch@tuwien.ac.at
Birthday: 05 / 05 / 1985
Birth place: Vienna, Austria

Education

03 / 2010 - 11 / 2016

Vienna University of Technology, Institute of Software Technology and Interactive Systems
Doctoral College on Computational Perception. Dissertation: 2D-to-3D conversion of videos.

01 / 2006 - 05 / 2010

Vienna University of Technology, Mag.a rer. soc. oec.
Master program in Computer Science Management. Graduated with honors.

03 / 2006 - 09 / 2009

Vienna University of Technology, DI.in
Master program in Media Informatics. Graduated with honors.

10 / 2003 - 03 / 2006

Vienna University of Technology, Bakk.a techn.
Bachelor program in Media and Computer Science.

09 / 1995 - 06 / 2003

Schopenhauer School, Vienna
Secondary school with focus on mathematics and science (*Realgymnasium*).

Work Experience

03 / 2010 - 09 / 2015

Research and Teaching Assistant, **Vienna University of Technology**, Vienna
Research and teaching at the **Institute of Software Technology and Interactive Systems**.

Teaching: *VO Video Analysis*, *SE Seminar on Media Informatics, SE Seminar on Image and Video Analysis and Synthesis, SE Seminar on Computer Vision and Pattern Recognition, PR From Design to Software 2, PR Visual Computing Project;* Supervision of various Bachelor and Master theses.

Participation in research projects: *Evaluation and Design of Energy Functions for Global Stereo Matching* (2010 - 2011, FWF), *Doctoral College on Computational Perception* (2010 - 2014, TU Vienna), *An Industrial Implementation of Modern Inpainting Techniques for Depth-based 3D Film Editing* (2012 - 2014, ZIT), *Temporal-Consistent Stereo Matting for High-Quality Novel View Synthesis and Visual Effects* (2009 - 2015, WWTF), *Intelligent Workflow Design for Low-Cost 3D Film Production* (2013 - 2015, FFG).

03 / 2007 - 03 / 2011

Content Design, JoinVision E-Services GmbH, Vienna
Web- and content design (2007 - 2011, FFG).

06 / 2007 - 08 / 2007

Internship, Siemens AG Austria, Vienna
Testing, implementing and documenting a system for medical studies.

06 / 2005 - 07 / 2009

Various Web Projects, Vienna
For example:
02 / 2009, *Architekt Fritz Göbl Ziviltechniker GesmbH* - architect Fritz Göbl
06 / 2008, *Wirtschaftspolitische Akademie* - politico-economic Academy
06 / 2007, *Lokomotive Döbling* - a football club
06 / 2006, *London Club Vienna* - a night club

Personal Skills and Competences

Languages

German (native), English (intermediate), French (beginner), Italian (beginner)

Computing Skills

C/C++, MATLAB, Java, PHP; OpenCV, … Various office and image editing applications.

NICOLE BROSCH

Home address:
Wallrissstrasse 64/6
1180 Vienna, Austria

E-mail: nicole.brosch@tuwien.ac.at

Selected Publications

02 / 2016	N. Brosch, T. Schausberger, M. Gelautz. *Towards Perceptually Coherent Depth Maps in 2D-to-3D Conversion*. In Proceedings of EI, 2016, pp. 1-11.
12 / 2014	M. Ivancsics, N.Brosch, M. Gelautz. *Efficient Depth Propagation in Videos with GPU-acceleration*. In Proceedings of VCIP, 2014, pp. 1-4.
02 / 2014	N. Brosch, M. Nezveda, M. Gelautz, F. Seitner. *Efficient quality enhancement of disparity maps based on alpha matting.* In Proceedings of EI, 2016, pp. 1-10.
02 / 2014	S. Ghuffar, N. Brosch, N. Pfeifer, M. Gelautz. *Motion Segmentation in Depth and Intensity Videos*. Integrated Computer-Aided Engineering, 21(2014), 3, pp. 303-218.
08 / 2012	N. Brosch, A. Hosni, C. Rhemann, M. Gelautz. *Spatio-temporally Coherent Interactive Video Object Segmentation via Efficient Filtering*. In Proceedings of DAGM/ÖAGM, 2012, pp. 418-427.
05 / 2011	N. Brosch, C. Rhemann, M. Gelautz. *Segmentation-Based Depth Propagation in Videos.* In Proceedings of ÖAGM,11, 2011, pp. 1-10. (**Best Paper Award**)

Reviewing

2012 - 2015	**ÖAGM’15**: Symposium of Austrian Association for Pattern Recognition. **TVCG’14:** Transactions on Visualization and Computer Graphics. **HOT3D’14**: The 5th International Workshop on Hot Topics in 3D. **TransImProc’14:** Transactions on Image Processing. **CVWW’13 & ’15** : The 18th and 19th Computer Vision Winter Workshop. **ICAEJ’13**: Integrated Computer-Aided Engineering (Journal). **IWSSIP’12:** International Conference on Systems, Signals and Image Processing. **DAGM/ÖAGM’12**: Joint Symposium of German Association for Pattern Recognition and Austrian Association for Pattern Recognition.

Further Training

07 / 2012	**International Computer Vision Summer School** (Italy). Recognition, Registration, Reconstruction.
03 / 2012	**Workshop**. „Die FWF Fördermaschine oder wie präsentiere ich meine Idee richtig und erfolgreich“.
12 / 2009 - 03 /2010	**Language Training**. Upper Intermediate English. Language Centre, University of Vienna.

Accomplishments

09 / 2011	**Student Mobility and Conference Grant**. IEEE Austria Section.
05 / 2011	**Best Paper Award**. ÖAGM Workshop 2011.
09 / 2008 - 09 / 2009	**Scholarship** in Visual Computing. Vienna University of Technology.

Professional Societies

01 / 2012 - present	**Women in Engineering (WIE) officer** of IEEE Austria Section.

Talks

03 / 2014	**Gender Imbalance Event.** “Life of a PhD student…“