Segmentation and Inpainting for Stereoscopic Videos

A dissertation submitted to the University of Dublin for the degree of Doctor of Philosophy

Félix Raimbault
Trinity College Dublin, February 2015
To my family.
Abstract

Nowadays, video processing technology is ubiquitous. It has found numerous applications from cinema post-production to television sets. Recent developments of digital stereoscopic videography call for new tools to help carry out post-production tasks for this medium. The flexibility offered by digital video is a key factor to the recent success of stereoscopic cinema, as most artifacts can be corrected or attenuated during post-production so as to deliver a pleasant 3D experience throughout.

In this thesis, we study two difficult fundamental aspects of stereo-video post-production, for which existing tools are far from matching the abilities of artists: segmentation and inpainting. Both algorithms need temporal stability as well as view consistency. We investigate the adaptation and extension of existing techniques for monoscopic videos so as to process stereoscopic sequences. Our work exploits stereo-disparity information from a stereo scene in conjunction with long-term trajectories to improve on state-of-the-art approaches.

Automatic video segmentation techniques are aimed at creating masks to delineate coherent regions in a video. The far reaching goal is that these segments correspond to semantically meaningful video objects. Current state-of-the-art techniques cannot replace semi-automatic rotoscoping, but they alleviate the work of artists by limiting the amount of required interaction. In this thesis, we present a novel sparse stereo-video segmentation technique that builds on existing 2D sparse segmentation frameworks. Our technique is based on the computation of an affinity measure between feature tracks, designed to process stereo videos. Our approach leads to greater temporal coherence and yields a more consistent segmentation across views, compared to existing techniques. Instead of requiring user interaction on a frame-to-frame basis to correct the segmentation output, we employ user interaction at the shot level, and on both views jointly.

Video inpainting is particularly important during post-production to create clean plates or remove rig attached to actors or objects, so as to create convincing visual effects. The goal of research in this domain is to create a fully automated approach. However, existing techniques need user intervention both during the creation of the mask delimiting the object to inpaint, and often to correct artifacts in the inpainted region. In this thesis, we focus on improving the quality of the results to reduce the need for user-driven retouching. Our main contribution is to present a hybrid method that combines an exemplar-based approach, which fills in the missing area via sampling, and a motion/disparity-based approach, which reconstructs the missing region by pulling the available data from other parts of the sequence. Our idea is to guide the selection of candidate pixel samples by using long-term motion and inter-view disparity information.
I hereby declare that this thesis has not been submitted as an exercise for a degree at this or any other University and that it is entirely my own work.

I agree that the Library may lend or copy this thesis upon request.

Signed,

______________________________
Félix Raimbault

February 21, 2015.
Acknowledgments

Thanks to my family and friends for their patience during the years it took me before I could submit my thesis. Thanks to my supervisors and colleagues in the Sigmedia lab, in Trinity College Dublin, for their support, wild discussions during coffee breaks or in the pub, and contributing to the good atmosphere in the lab. Thanks as well to my colleagues at Sony during my internship in Stuttgart, which gave me back more motivation to keep going on. Last, but not least, thanks to the research community in general for bringing passion and brilliance to the field of video processing.

This work has been funded from Science Foundation of Ireland as part of Project 08/IN.1/I2112, Content Aware Media Processing.
## Contents

1 Introduction 1

1.1 Main Contributions of this Thesis 2

1.2 Thesis Outline 3

1.2.1 Stereo-Video Segmentation 3

1.2.2 Stereo-Video Inpainting 4

1.2.3 Video Stabilisation 6

1.3 List of Publications 6

2 Towards Segmentation of Stereoscopic Videos 9

2.1 Dense Video Segmentation Techniques 10

2.1.1 Multi-Layer Segmentation 10

2.1.2 Bi-Layer Segmentation 12

2.2 Sparse Video Segmentation Techniques 15

2.2.1 3D Sparse Segmentation 15

2.2.2 2D Sparse Segmentation 17

2.3 Sparse-to-Dense Video Segmentation Approaches 20

2.3.1 Methods Based on Short-term Feature Point Associations 21

2.3.2 Methods Based on Long-term Feature Point Tracks 23

2.4 Supervised Video Segmentation Systems 25

2.4.1 Tightly-Supervised Video Matting 26

2.4.2 Loosely-Supervised Video Segmentation 27

2.5 Final Remarks 29

3 User-Assisted Sparse Stereo-Video Segmentation 31

3.1 Basic Definitions 33

3.2 Preprocessing Steps 34

3.2.1 Estimation of the Feature Point Trajectories 34

3.2.2 Temporal Smoothing of the Velocity and Disparity Values 35

3.2.3 Background Subtraction via Bi-layer Segmentation 36
3.3 Computation of the Affinity Matrix

3.3.1 Comparison of Overlapping and Disjoint Tracks in any View

3.3.2 Computation of the Pairwise Distance Matrix

3.3.3 Computation of the Spatial Distance

3.3.4 Computation of the Motion Distance

3.3.5 Computation of the Temporal Decay

3.3.6 Obtaining the Pairwise Affinity Matrix

3.4 Split-and-Merge Feature Point Clustering

3.4.1 Automatic Split Step using Normalised Cuts

3.4.2 Our Modified Stopping Criterion for Recursive Splitting

3.4.3 Automatic Merge Step via Min-max Cut Linkage

3.4.4 User-assisted Refinement Step

3.5 Final Remarks

4 Experiments on Sparse Stereo-Video Segmentation

4.1 Analysis of Our Automatic Segmentation Results

4.1.1 Experiments on the Hopkins 155 Dataset

4.1.2 Experiments on the Gpo_track_2 Sequence

4.1.3 Experiments on the Kylemore Sequence

4.1.4 Experiments on the Traffic Sequence

4.2 Analysis of Our User-assisted Segmentation Results

4.2.1 Walk-through Example on the Rose_garden_tracking Sequence

4.2.2 Walk-through Example on the Sphere_2 Sequence

4.2.3 Results on the Gpo_pan Sequence

4.2.4 Results on the Close_up_1 Sequence

4.3 Discussion

4.4 Final Remarks

5 From Image Inpainting to Stereo-Video Inpainting

5.1 Image Inpainting Techniques

5.1.1 The Genesis of Image Inpainting Algorithms

5.1.2 The Exemplar-based Framework for Image Inpainting

5.2 Video Inpainting Methods

5.2.1 From Image Inpainting to Video Inpainting

5.2.2 Video Inpainting Based on Motion Segmentation

5.2.3 Video Inpainting Based on Motion Repair

5.2.4 Video Inpainting Based on Feature Point Tracking

5.3 Inpainting for Stereoscopic Images and Videos

5.3.1 From Monoscopic Inpainting to Stereoscopic Inpainting
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Start Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.3.2</td>
<td>Image Inpainting Based on Disparity Completion</td>
<td>84</td>
</tr>
<tr>
<td>5.3.3</td>
<td>Video inpainting for Multi-view Displays</td>
<td>87</td>
</tr>
<tr>
<td>5.4</td>
<td>Final Remarks</td>
<td>90</td>
</tr>
<tr>
<td>6</td>
<td>Exemplar-based Stereo-Video Inpainting</td>
<td>91</td>
</tr>
<tr>
<td>6.1</td>
<td>Problem Statement</td>
<td>92</td>
</tr>
<tr>
<td>6.1.1</td>
<td>Overview</td>
<td>92</td>
</tr>
<tr>
<td>6.1.2</td>
<td>Notions of Pixel Neighbourhood and Data Patch</td>
<td>94</td>
</tr>
<tr>
<td>6.1.3</td>
<td>Usage of Motion and Disparity Vectors</td>
<td>96</td>
</tr>
<tr>
<td>6.2</td>
<td>A Probabilistic View of Examplar-based Inpainting</td>
<td>98</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Patch Matching and Sampling</td>
<td>98</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Order of Filling for Greedy Reconstruction</td>
<td>99</td>
</tr>
<tr>
<td>6.2.3</td>
<td>Our Extended Objective Function for Patch Matching</td>
<td>100</td>
</tr>
<tr>
<td>6.3</td>
<td>Our Stereo-Video Patch Matching Strategy</td>
<td>102</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Computing the Order of Filling</td>
<td>102</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Computing a Distance Measure for Image Patches</td>
<td>104</td>
</tr>
<tr>
<td>6.3.3</td>
<td>Exploiting the Amount of Useful Information in a Patch</td>
<td>105</td>
</tr>
<tr>
<td>6.3.4</td>
<td>Exploiting Prior Information for Source Frame Consistency</td>
<td>106</td>
</tr>
<tr>
<td>6.4</td>
<td>Candidate Search using both Motion and Disparity</td>
<td>107</td>
</tr>
<tr>
<td>6.4.1</td>
<td>Patch Tracking using Reconstructed Motion and Disparity Vectors</td>
<td>108</td>
</tr>
<tr>
<td>6.4.2</td>
<td>Local Refinement via Gradient Descent</td>
<td>110</td>
</tr>
<tr>
<td>6.4.3</td>
<td>Skip the Search via Coherent Patch Sewing</td>
<td>112</td>
</tr>
<tr>
<td>6.5</td>
<td>Image Reconstruction</td>
<td>114</td>
</tr>
<tr>
<td>6.5.1</td>
<td>Estimating the Colour Correction Coefficient</td>
<td>114</td>
</tr>
<tr>
<td>6.5.2</td>
<td>Colour Correction and Stereo-Temporal Blending</td>
<td>115</td>
</tr>
<tr>
<td>6.5.3</td>
<td>Update Rules</td>
<td>116</td>
</tr>
<tr>
<td>6.6</td>
<td>Summary of the Algorithm</td>
<td>117</td>
</tr>
<tr>
<td>6.7</td>
<td>Final Remarks</td>
<td>117</td>
</tr>
<tr>
<td>7</td>
<td>Experiments on Inpainting for Stereoscopic Videos</td>
<td>119</td>
</tr>
<tr>
<td>7.1</td>
<td>Our Experimental Setup</td>
<td>121</td>
</tr>
<tr>
<td>7.1.1</td>
<td>A Small Database of Stereoscopic Videos</td>
<td>121</td>
</tr>
<tr>
<td>7.1.2</td>
<td>Overview of the Inpainting Tests</td>
<td>122</td>
</tr>
<tr>
<td>7.1.3</td>
<td>Quality Metrics for Stereoscopic Videos</td>
<td>123</td>
</tr>
<tr>
<td>7.1.4</td>
<td>Implementation and Testing Environment</td>
<td>124</td>
</tr>
<tr>
<td>7.2</td>
<td>Analysis of the Results</td>
<td>125</td>
</tr>
<tr>
<td>7.2.1</td>
<td>Results on the Rose_garden Sequence</td>
<td>125</td>
</tr>
<tr>
<td>7.2.2</td>
<td>Results on the Red_barrier Sequence</td>
<td>125</td>
</tr>
<tr>
<td>7.2.3</td>
<td>Results on the Water_drop Sequence</td>
<td>129</td>
</tr>
</tbody>
</table>
7.2.4 Results on the Walking man Sequence ........................................ 131
7.2.5 Results on the Walking lady Sequence ....................................... 133
7.2.6 Results on the Green screen and Pub talk Sequences ................. 136
7.3 Discussion .................................................................................. 139
7.4 Final Remarks ........................................................................... 141

8 Conclusions .................................................................................. 143
  8.1 Towards User-assisted Dense Stereo-Video Segmentation .......... 143
  8.2 Towards Segmentation-based Inpainting for Stereo Videos ....... 144
  8.3 Final Words ............................................................................. 146

A Appendix: Video Stabilisation ..................................................... 147
  A.1 Related Work ........................................................................... 147
  A.1.1 The Three Steps of Video Stabilisation .................................. 148
  A.1.2 Rolling Shutter Removal Techniques .................................... 148
  A.1.3 Logo Detection Methods ...................................................... 149
  A.2 Video Stabilisation for Home Video and TV Broadcast ........... 149
  A.2.1 Dominant Motion Estimation ................................................ 149
  A.2.2 Adaptive Motion Compensation .......................................... 151
  A.2.3 Rolling Shutter Correction .................................................... 152
  A.2.4 Logo Detection .................................................................... 153
  A.3 Experimental Results ............................................................... 155
  A.4 Final Remarks .......................................................................... 156

Bibliography .................................................................................... 157
Recently, developments of digital stereoscopic videography have given new means of expression to professional filmmakers and, more and more, amateur film enthusiasts. Filming in stereo adds costs and technical constraints to the production, thus creating the need for new digital tools and new experts to help using this medium. For instance, the stereographer helps a crew to think in 3D during film pre-production. In stereo-3D cinema, two viewpoints are projected, one for each eye, so as to fool the spectator’s binocular vision. This process is more tiring for the brain than natural vision. Therefore, the rhythm of narration, the composition of a scene, and the transitions between shots must be thought of differently.

Producing 3D films seems to have become gradually unavoidable for the cinema industry to keep drawing crowds to the theatres. Stereoscopic cinema is attractive compared to traditional monoscopic cinema mainly because of the augmented immersion effect for the spectator. Even if it is primarily driven by super-productions such as Avatar (2009) by James Cameron and Gravity (2013) by Alfonso Cuarón, stereo-cinema can also be considered as a means of becoming closer to realism as advocated by the film theorist André Bazin. And it may well become the new standard for filming documentaries, considering recent works such as Pina (2011) by Wim Wenders and Cave of Forgotten Dreams (2010) by Werner Herzog.

During the previous bursts of popularity of stereo cinema in the 1950’s and in the 1980’s, the main issue was that the viewing experience could be literally sickening. The technical precision required by stereo cinema could simply not be attained on set. Therefore, problems due to colour imbalance, view mis-alignment and stereopsis outside of the comfort zone were widespread. Nowadays, most cinema productions, if not all, use digital video. The flexibility
offered by digital video is a key factor to the recent success of stereoscopic cinema, as most artifacts can be corrected or attenuated during post-production so as to deliver a pleasant 3D experience throughout.

Nowadays, video processing technology is ubiquitous. It has found numerous applications from television sets to cinema post-production. Frame in-bewteening, for instance, is now implemented on TV sets directly due to the high computational power available. Post-production for digital cinema has been facilitated by the development of compositing software such as Nuke\footnote{https://www.thefoundry.co.uk/products/nuke-product-family/}. Particularly, in 2006, the team which developed a plug-in suite for Nuke called Furnace\footnote{http://www.thefoundry.co.uk/products/plugins/furnace/} was awarded a Scientific and Engineering Award from the Academy of Motion Picture Arts and Sciences. The Academy considered the Furnace toolset’s modularity, flexibility and robustness to have set a high standard of quality for Optical Flow based image manipulation. More recently, a suite of plug-ins called Ocula\footnote{https://www.thefoundry.co.uk/products/ocula/} has been developed to integrate a stereo-processing toolkit within Nuke. During the course of this thesis, we use Nuke as a source of data for research and as a tool for visualisation.

1.1 Main Contributions of this Thesis

In this thesis, we choose to study two post-production applications open to research, for which existing tools are far from the abilities of artists: segmentation and inpainting. Both algorithms are often employed as stepping stones to enable user-assisted image retouching or enhancement tasks commonly performed during post-production. These algorithms represent two difficult fundamental aspects of stereo-video post-production, which both need temporal stability as well as view consistency. We investigate the adaptation and extension of existing techniques for monoscopic videos so as to process stereoscopic sequences. Our work exploits stereo-disparity information from a stereo scene in conjunction with long-term trajectories to improve on state-of-the-art approaches.

Automatic segmentation techniques are aimed at creating masks to delineate coherent regions in a video. The far reaching goal is that these segments correspond to semantically meaningful video objects. Current state-of-the-art techniques cannot replace semi-automatic rotoscoping, but alleviate the work of artists by limiting the amount of required interaction. The main challenges of stereo-video segmentation are obtaining consistent segments along time and across views, as well as determining the level of segmentation of the output. In this thesis, we present a novel sparse stereo-video segmentation technique that builds on an existing 2D sparse segmentation framework \cite{22,54}. Our technique is based on the computation of an affinity measure between pairs of tracks, which indicates how similar two tracks are to each other. Compared to similar state-of-the-art techniques, our approach leads to a greater temporal coherence and
yields a more consistent segmentation across views, due to our novel way of computing the affinity between each pair of tracks. Setting the level of segmentation automatically is an open problem. Most existing methods require user interaction on a frame-to-frame basis to mitigate this problem. Our contribution is to employ user interaction at the shot level instead, and on both views jointly. Our technique is restricted to the sparse domain, but it could be used as a first step to obtain a dense segmentation via a sparse-to-dense approach [11, 116].

Video inpainting is particularly important during post-production to create clean plates or remove rigs attached to actors or objects to create convincing visual effects. Interestingly, the industry has adopted automatic motion-based rig removal techniques early on. For instance, the method proposed by Kokaram et al. [88] is included in the Furnace suite of plug-ins for Nuke. This technique is a relatively fast way of reconstructing and using motion, to reveal the unknown data within the rig area, by pulling information from known locations in a video. However, spatiotemporal consistency problems occur if the reconstructed motion information is inaccurate. In parallel, the computer vision community has come up with exemplar-based video inpainting techniques [111, 144, 171] inspired by constrained texture synthesis for image completion [34, 45]. The main idea of exemplar-based approaches is to fill in a missing region in an image or a video with similar candidate pixels sampled from other regions. Image-based approaches are not suited to process videos, because temporal consistency is missing, and the speed of such techniques may not scale up to process a sequence of frames efficiently. However, they allow for preservation of the texture quality in large missing regions. In this thesis, our main contribution is to present a hybrid approach which intends to obtain the best of both motion-based [88] and exemplar-based [111, 144, 171] techniques. Within an iterative exemplar-based framework, our stereo-video inpainting method uses long-term motion and inter-view disparity information to guide the selection of candidate pixels. Processing time is the main drawback of our technique, but it is interesting to analyse the qualitative improvements we obtain compared to selected state-of-the-art approaches [88, 111, 112] in terms of spatiotemporal coherence and view consistency.

1.2 Thesis Outline

This thesis is divided into two parts. Firstly, the research topic of stereo-video segmentation is covered in chapters 2, 3, and 4. Secondarily, the research topic of stereo-video inpainting is covered in chapters 5, 6, and 7. Finally, work on video stabilisation that has been carried out during an internship at Sony’s Stuttgart Technology Centre is reported in appendix A.

1.2.1 Stereo-Video Segmentation

Chapter 2: This chapter reviews the state of the art of segmentation for videos and stereo videos. Dense automatic techniques [32, 89, 164] generate a labelling for each pixel but they often
lack long-term consistency. Sparse approaches [22,33,179], on the other hand, only assign a label to feature point tracks, but they allow for a longer-term consistency of the output. Sparse-to-dense methods [101,116,178] have then been designed to combine the strength of both approaches and obtain a long-term coherence for dense segmentation. Automatic techniques lack high-level understanding of the scene, so there is no guarantee the output is useful for a given application. This is the main motivation for the design of supervised approaches [5,11,62], which often necessitate the intervention of the user on each frame to correct a dense segmentation. User interaction on the sparse domain has potential for developments to reduce the amount of required user intervention. Some dense segmentation approaches use depth information to process stereo videos [2,86,93], but, particularly in the sparse segmentation literature, view consistency of the output segments has not been studied extensively.

Chapter 3 This chapter details our research on user-assisted sparse segmentation of stereoscopic videos. Our method is based on clustering feature point tracks via a spectral analysis of the pairwise affinity matrix, which holds information on how similar tracks are to each other. We introduce a novel affinity measure on both temporally overlapping and disjoint tracks, whereas state-of-the-art methods can only compare overlapping tracks [22,53,54]. Moreover, we process tracks on both views jointly, by exploiting inter-view disparity information. We then use an automatic clustering method, based on standard existing approaches [43,142], to construct a hierarchy of clusters. This creates a binary tree of clusters, in which each node represents a regrouping of two clusters. Finally, we employ a user-assisted split-and-merge method to refine the segmentation. The user-assisted refinement step allows users to explore the hierarchy of clusters in a few interactions, so that they can get an output with the level of segmentation that suits their needs without further processing.

Chapter 4 This chapter presents the experimental results of our segmentation technique on the Hopkins 155 dataset [158], as well as on a selected corpus of stereo videos extracted from the Sigmedia database [31]. We show that, compared to existing similar approaches, our technique allows a greater temporal coherence of the output. Particularly, the computation of affinity between disjoint tracks enables our method to connect objects in time even under full occlusion, providing that their motion remains similar. Secondarily, we show that by processing and segmenting tracks jointly, on both left and right views, in a single framework, our technique yields a consistent segmentation across views. We also show that our user-assisted refinement approach allows for correction of errors in the output and increases the range of possible applications for our technique.

1.2.2 Stereo-Video Inpainting

Chapter 5 This chapter reviews the state of the art of inpainting methods, for images, videos and stereoscopic media. Image inpainting has been first developed to repair small artifacts with
diffusion methods [14], and then extended via exemplar-based sampling approaches to process larger missing areas [34]. However, temporal consistency must also be enforced when filling in missing data that can span several frames in videos. Video inpainting techniques [88, 111, 172] have been designed to ensure a temporally smooth reconstruction. Most state-of-the-art approaches are based on a combination of segmentation [144], motion repair [145] and patch tracking [74]. In the stereoscopic domain, the development of specialised inpainting methods has been mainly motivated by view synthesis applications for 3DTV and 2D-to-3D conversion [63]. Only a few techniques are concerned with the general problem of object removal in stereoscopic images [168]. A common denominator for stereo-inpainting is the need for view consistency in the output. One of the main challenges for the development of stereo-video inpainting is to maintain both spatiotemporal coherence and view consistency.

Chapter 6: This chapter details our research on stereo-video inpainting. Our method builds on an exemplar-based technique designed to fill in missing data in images [34, 45]. We extend this image-based framework to process stereo videos by using long-term motion information as well as inter-view disparity to guide a patch-based sampling mechanism. Motion and disparity vectors are repaired using a variational image inpainting method [14] during preprocessing. Our technique then fills in each frame sequentially. In each frame, missing pixels are processed in the order defined by a priority measure. For each missing pixel, the best matching candidate is searched around the point trajectory along time and across views. The trajectory is estimated with reconstructed motion and disparity information. The image patch surrounding each candidate is compared to the image patch surrounding the missing site using a distance measure based on the structural similarity [170], and taking into account stereo-temporal consistency constraints. The best match is then selected for replacement after local refinement to align the candidate picture information to the data around the missing pixel. The selected candidate picture data then goes through a stereo-temporal filter that includes colour correction. This filtering process ensures that the reconstruction varies smoothly along time and is consistent across views. A coherent patch sewing mechanism based on coherence search [4, 18] is used to speed up the technique by copying as much information as possible from the candidate patch while preventing artifacts.

Chapter 7: This chapter presents the experimental results of our inpainting technique on selected stereo videos extracted from the Sigmedia database [31] and associated masks delineating the target region to inpaint. We design tests where an unwanted object or person has to be removed from the sequence. The results of these tests are analysed qualitatively as no ground-truth data is available. We also design tests where the mask represents an artificial degradation. The results of these tests are analysed both qualitatively and quantitatively as the inpainted output must match the original sequence closely. Quantitative analysis is performed with a 2D and 3D structural similarity measure [27, 170]. Our results are compared to two state-of-the-
art techniques, namely Kokaram et al. [88] and Newson et al. [111, 112]. The former has been adopted as a standard by the industry, and is implemented in Nuke. The later is a recent best-in-class patch-based video inpainting technique. Our technique compares favourably to these two methods in terms of temporal and view consistency, but not in terms of processing time.

1.2.3 Video Stabilisation

In the course of this thesis, the topic of video stabilisation has been investigated during an internship at Sony’s Stuttgart Technology Centre, under the supervision of Yalcin Incesu. During this internship, Nuke has been used as a platform for research to develop a video stabilisation technique for applications in the TV domain. Results of this project are not part of this thesis, but have been presented in ICIP’13 and are included as an appendix.

Appendix A

In this appendix, we present a quick review of video stabilisation methods [109, 154, 180], details of our technique, and a description of our results. We present a video stabilisation system with integrated logo detection and rolling shutter correction. Logo detection [118, 165] is particularly useful to process sequences extracted from TV broadcast. Rolling shutter correction [6, 61, 64] is necessary when stabilising videos taken with most smartphones. Our method first estimates translational dominant motion in a way that enables treatment of a wide variety of camera motions, including zooming which remains difficult using prior similar techniques [153, 154]. We then remove motion jitter in a video with an adaptive low-pass filter. The filter retains intended motion while removing high frequency jitter by combining motion classification and on-line drift correction. The parameters of the filter can be tailored to the desired system latency, making it suitable for real-time applications. We extend our technique to attenuate rolling shutter artifacts via motion interpolation. The method does not need calibration to estimate active time of the sensor but works on a smaller class of videos. We add logo detection to stabilise television broadcast content while leaving static on-screen display unchanged. Results show good potential of our approach, and highlight the fact that video stabilisation can benefit from integrating a logo detection mechanism.

1.3 List of Publications

The work described in this thesis has led to the following publications:


1.3. List of Publications


We refer the reader to the associated websites mentioned above to view the results described in these publications.

All the video sequences presented in this thesis can be watched on our website, at the following address: [http://www.sigmedia.tv/Misc/ResultsThesisFelix](http://www.sigmedia.tv/Misc/ResultsThesisFelix). These videos are also available on Youtube:

- [http://www.youtube.com/playlist?list=PLWmBvOgByxjZmp4XnHZ300GQN0Dp0DB1T](http://www.youtube.com/playlist?list=PLWmBvOgByxjZmp4XnHZ300GQN0Dp0DB1T)
- [http://www.youtube.com/playlist?list=PLWmBvOgByxjZ3ElnxvWe1Kn6nM3BD8beR](http://www.youtube.com/playlist?list=PLWmBvOgByxjZ3ElnxvWe1Kn6nM3BD8beR)

The first link is associated with the segmentation part and the second link is associated with the inpainting part.
Towards Segmentation of Stereoscopic Videos

The recent revival of the stereo movie industry, as well as the infiltration of stereoscopic content in the consumer market, call for new tools to automatise post-production tasks for stereo videos. A key technology to enable the development of tools such as matte propagation for stereo-compositing is video segmentation. It has recently been applied to video matting [11], unsupervised learning [116], autonomous navigation [2] and driver assistance [86].

Video segmentation has received a lot of attention in scientific literature. Indeed, it is often used as a stepping stone to develop application-specific systems in digital video processing, pattern recognition and computer vision. The problem is to automatically decompose a video into coherent regions, also called segments, clusters, layers, labels, masks or mattes. The main challenge is to generate a partition that is suitable for the targeted application. Properties such as robustness to dynamic noise, temporal coherence and granularity are critical.

This chapter reviews how segmentation methods process monoscopic and stereoscopic videos. The review is designed to help the reader grasp the motivations for our work on stereo-video segmentation, described in chapter 3. We first review in section 2.1 dense techniques, showing how depth has been exploited in stereo imagery segmentation recently. In section 2.2 we detail sparse motion-based approaches, which are the most related to our work. Section 2.3 shows how sparse segmentation can be used as a first step to improve temporal coherence of dense segmentation. Finally, in section 2.4 we discuss the challenge of finding a suitable granularity for the output, which necessitates high-level information often obtained by user supervision.
2.1 Dense Video Segmentation Techniques

We differentiate between dense and sparse approaches to video segmentation depending on the precision that is targeted in the final output. In the former, a label is assigned to all pixels, whereas in the latter, only a limited number of feature points are considered. Implicitly or explicitly there is a low-level feature extraction process at the basis of any segmentation technique. In dense techniques, features are defined on all pixels whereas sparse techniques only compute features for a small number of interest points. Features are generally a combination of appearance, location, motion and depth cues. Segmentation can then be defined as the process of grouping points having similar features together to form coherent regions.

2.1.1 Multi-Layer Segmentation

The majority of dense video segmentation techniques employ the concept of layer representation [40, 164]. An example of multi-layer segmentation can be seen in figure 2.1. A layer refers to a planar region in which all pixels have similar features. Such techniques assume that a scene can be approximately described by multiple planar regions. It is an intuitive way of representing a video that follows the rules of compositing. Each layer contains information on the colour, transparency and local motion of pixels. To resolve mutual-occlusion problems, depth ordering between layers can be estimated a posteriori via occlusion reasoning [164] or given a priori by the user [103]. Determining automatically the number of layers is a difficult problem, and it must often be specified by the user as a parameter [75,103]. These techniques have found many applications such as video coding [164], object removal [75,164], and video stabilisation [75].

The Use of Motion: In the paper that popularised the notion of layers, Wang and Adelson [164] achieve dense segmentation by decomposing the Optical Flow [69] between adjacent pairs of frames into a set of planar regions in velocity space. The algorithm proceeds iteratively by estimating affine parameters for each layer, and using k-means clustering and segment merging until a stability criterion is reached. The technique segments one frame after another, using only two images at a time to determine local motion, which can cause temporal inconsistencies. The final number of layers is obtained by adaptively merging segments together, which does not guarantee that the resulting segmentation is useful. The technique can also produce an erroneous result if the affine model does not describe the motion in the scene well enough. Moreover, low-texture areas can cause problems as they do not contain any motion information.

The Use of Appearance: Jojic and Frey [75] use a probabilistic framework to infer layer parameters. A variational Expectation-Maximisation (EM) [42,175] algorithm is employed to learn a Gaussian mixture of layers from a video sequence. The number of layers must be specified by the user. Contrary to Wang and Adelson [164], which is based solely on motion information, here, only appearance and shape cues are exploited by the model. However, the shape of a
Figure 2.1: Illustration extracted from Abramov et al. [2]. (1) Disparity is estimated from the stereo pair of frames at time $t$, and image segmentation is applied on the left view. (2) Labels from the right view at time $t - 1$ are transferred to the right view at time $t$ using Optical Flow information. The labels from the left view at time $t$ are also transferred to the right view at time $t$ using disparity information. Both label warping steps are performed simultaneously, and conflicts are resolved by random selection of the label source. (3) The transferred labels are used as an initialisation to a relaxation process, during which erroneous bonds are corrected. (4) The segments are then extracted on the right view at time $t$ after convergence of the relaxation process.

layer, represented by a mask, is allowed to undergo a discrete set of predetermined possible 2D transformations from frame to frame. If the motions of objects cannot be approximated accurately by the model, the technique does not work correctly. The interesting aspect of Jojic and Frey [75] is the use of a variational probabilistic framework for layer extraction. However, EM methods are prone to local minima, particularly if objects are moving with similar motions. A more recent layer learning approach by Pawan Kuman et al. [120] introduces substantial improvements to the process. Their method combines motion and appearance models, and a spatial connectivity prior under a more robust energy minimisation framework.

**The Use of Disparity:** Abramov et al. [2] propose a real-time approach for online automatic segmentation of stereoscopic videos. The method is designed for applications in stereo vision for robotics. The workflow of this technique is illustrated and summarised in figure 2.1. Image segmentation is first applied on the current frame in one view. Then labels are assigned to each
region via a propagation technique. Previous labels are warped from both views using motion and disparity vectors. After removing segments that are too small, the warping step is followed by a relaxation step via energy minimisation. In Abramov et al. [2], long-term coherence can be violated as objects which become totally hidden and reappear would be assigned a different label. Fast motions can also cause label transfer to fail, and merging back large parts of objects that become temporally disconnected is not possible. Although the technique is designed for speed, and slower off-line approaches can reach a higher temporal coherence, these problems are common to many dense segmentation methods.

### 2.1.2 Bi-Layer Segmentation

It is assumed in some papers [32, 89, 93] that there are only two layers in the video because the targeted application needs two layers only. An example of bi-layer segmentation can be seen in figure 2.2. Generally, a foreground region is to be separated from the background, which is everything else. For instance, in Odobez and Bouthemy [117], motion of the background is assumed to be the global motion and all pixels which exhibit a different motion are labelled as foreground. This class of techniques is called bi-layer segmentation in general or video matting when a precise delineation of foreground objects is required. It is of particular interest in video conferencing [93, 177], where typically there is only one person of interest. And also in movie post-production [32, 89], where a few actors need to be segmented individually very precisely or a crowd can be considered as a single entity.

**The Use of Frame Difference:** Kokaram et al. [89] combine global motion estimates [90] with image-based information in a Bayesian framework, to automatically generate rough mattes for video post-production. The interest of this technique is the use, in the likelihood design, of motion-compensated forward and backward frame difference. This is shown to improve accuracy of segmentation along object boundaries. Unlike many previous methods, spatial smoothness is also explicitly encouraged by a Markov Random Field (MRF) prior. An important remark by the authors is that a greater discrimination power can be achieved by using longer-term information, i.e. frame differences over multiple time instants. The solution is obtained as a Maximum A Posteriori (MAP) estimate computed by Iterated Conditional Modes [16]. The method has difficulties in case the appearance of the background is similar to the foreground, and when several moving objects overlap.

**Segmentation with Graph Cuts:** In the meantime, the development of image segmentation methods [133, 155, 159] has gained momentum after the formulation of the bi-layer segmentation problem in a MAP-MRF framework solved by Graph Cuts [19, 21]. In many cases, the user is asked to provide the initial partition. For instance, in the image segmentation tool GrabCut [133], a box has to be drawn around the foreground object. In other techniques, the user has to indicate foreground and background regions by drawing scribbles on each region [155].
2.1. Dense Video Segmentation Techniques

![Frame picture with scribbles](image1)

![Disparity map](image2)

![Matting without using depth](image3)

![Matting using depth](image4)

Figure 2.2: Illustration extracted from Pitié and Kokaram [123]. User scribbles on the original frame are used to initialise the segmentation. Blue scribbles indicate the background and pink scribbles indicate the foreground. Comparison of the resulting mattes after segmentation shows that using depth information allows for a more comprehensive estimation of the outline of the persons in the foreground.

Corrigan et al. [32] extend GrabCut [133] for automatic processing of videos. Automatic initialisation is performed via motion-compensated frame difference similar to Kokaram et al. [89] to remove user-interaction.

An other improvement proposed by Corrigan et al. [32] is the use of colour and motion in a joint feature space. Foreground and background statistics are estimated via two Gaussian Mixture Models similar to GrabCut, but in the extended feature space containing backward and forward motion information. Furthermore, Mean Shift [29] is employed to infer the number of components in each model. This allows a better representation of the data complexity compared to an arbitrarily fixed number. The use of motion and colour allows to improve the result in case background and foreground have a similar colour or a similar motion.

In a more recent paper, it has been shown by Pitié and Kokaram [123] that depth maps can be efficiently used for matting stereoscopic videos, as illustrated in figure 2.2. Depth has also been shown to bring improvements on GrabCut-based approaches, for instance GrabCutD [159] has been proposed as a direct extension of GrabCut with depth information. Kolmogorov
et al. [93] use a combination of colour and depth for bi-layer segmentation of stereo videos. The problem is cast in a probabilistic framework, and two solutions are proposed: Layered Dynamic Programming and Layered Graph Cuts. Occlusions are explicitly taken into account in the model to enforce geometric constraints, but not temporal consistency, although some statistics are learnt from the segmentation in the previous frame. A user-assisted stereo image segmentation based on Graph Cuts has been proposed by Tasli and Alatan [155]. The user is asked to input scribbles only on one view, and the segmentation is carried out on both views jointly by propagating the user input on the other view via feature matching.

For multi-layer segmentation via Graph Cuts [178], an iterative approximate solution to a multiway cut is often used. Each label in turn is selected as foreground, all others being merged as background. The algorithm terminates when the energy cannot be reduced further for any label.

**Combining Multiple Cues:** All pixels within an object tend to move roughly in the same way, as stated by Kokaram et al. [89], so motion cues give a good indication of the presence of different objects. However, texture-less areas do not admit reliable motion information, and articulated and non-rigid objects contain sub-sections that move in a different fashion. To group these regions into a single segment automatically, it is necessary to incorporate further information. To this end, the most recent developments in dense segmentation techniques fuse multiple cues, using together spatial, temporal and stereo information.

Wu et al. [177] have shown that fusing colour, motion and depth cues yields a better segmentation than using one of the cues alone. Similarly, in the multi-layer image segmentation case, it has been shown by Dal Mutto et al. [37] that a joint normalised representation of colour and depth allows for a better image segmentation than using one of the two cues alone. For instance, in case objects with different colours are close to each other, an approach based on depth only would fail. Whereas if objects with similar colours are distant to each other, an approach based on colour only would fail.

**Remaining Challenges for Dense Segmentation Approaches:** The most recent multi-layer segmentation approaches, such as Abramov et al. [2], tend to yield an oversegmentation of the scene which could be refined subsequently. This avoids the difficult problem of estimating the number of layers, for which there is no definite answer to date.

The use of Optical Flow is at the basis of many dense video segmentation techniques [2, 32, 164]. It causes problems at the boundaries of objects, because motion estimates are often erroneous around occlusion and disocclusion areas. Xiao and Shah [178] show that using an occlusion constraint increases the precision of dense segmentation along object boundaries. Occluded pixels are detected and explicitly labelled when incompatible layer assignments are made. The occlusion problem has been mainly studied previously in the context of stereo algorithms [46], concerning the precision of disparity estimation along object edges.


2.2 Sparse Video Segmentation Techniques

Most dense segmentation approaches lack long-term temporal coherence, because they only exploit short-term motion information such as Optical Flow. Typically, temporary stopping objects would be fused with the background if only motion information is used. Large displacements and disconnected objects are also major challenges to dense segmentation techniques. In the next section, we review sparse segmentation approaches which do not label each pixel in the video but have the advantage of allowing longer coherence.

2.2 Sparse Video Segmentation Techniques

Most sparse segmentation techniques are based on the analysis of feature point trajectories. Therefore, they need to employ a feature point tracker [10,136,143,152] to generate the trajectories or tracks as a preprocessing step. Not all feature point trackers perform equally in terms of spatial density and temporal track length. But they all allow longer-term correspondences between points compared to Optical Flow [69], which computes pairwise frame correspondences only. Exploiting long-term point trajectories seems to be a promising way of obtaining temporally consistent clusters over many frames. A review of sparse segmentation techniques can be found in Zappella et al. [183].

Most state-of-the-art sparse video segmentation approaches compute a pairwise affinity matrix to measure how similar tracks are to each other during their time of overlap. This allows to cluster tracks even under partial occlusions. Spectral clustering methods [52,113,142] are often employed to analyse the eigenvalues of the affinity matrix, and form groups of similar tracks. Dal Mutto et al. [37] have compared several clustering approaches in the context of stereo-image segmentation using colour and depth, combined with several stereo vision algorithms to obtain the depth information. They conclude that spectral clustering with the Nyström method [52] is the most robust segmentation method.

Grouping feature points constitutes an intermediate representation, which enables more advanced processing methods to derive parameters or models to represent each segment. Sparse segmentation techniques can be classified into two groups depending on their underlying motion model, namely 3D and 2D sparse segmentation.

2.2.1 3D Sparse Segmentation

The segmentation methods detailed in this section are designed to group together trajectories having a similar 3D motion, and are used mainly for applications to dynamic scene understanding and reconstruction. Generally, the affine camera model is used. In one of the seminal works in the field, Costeira and Kanade [33] introduce a factorisation method, to derive a shape-interaction matrix from the stacked spatial locations of all trajectories. Clustering is then operated by thresholding the entries of the shape interaction-matrix, as they should be null for a pair of trajectories belonging to the same motion. However, this procedure considers that all
Towards Segmentation of Stereoscopic Videos

trajectories are complete, is sensitive to noise, and assumes that a scene is made of rigid objects undergoing independent motions. Subsequent publications have endeavoured to generalise these assumptions and increase robustness to noise, outliers and missing data [47, 130]. Methods based on Costeira and Kanade [33] are often called multibody factorisation.

**Subspace Separation:** Sugaya and Kanatani [149] propose a multi-stage learning approach. The technique builds on Costeira and Kanade’s factorisation [33] and Kanatani’s subspace separation methods [78, 80]. Subspace separation is based on the fact that trajectories of points which belong to one 3D object are constrained to lie in a subspace of dimension at most four. However, previous approaches assume that the motions are independent and non-degenerate, which means that the full space is the direct sum of all subspaces. This assumption does not hold in many practical cases where motions are degenerate and partially dependent. Therefore, Sugaya and Kanatani [149] present a technique which first fits to the data a degenerate model, assuming motions lie in 2D parallel planes. The fitting is subsequently refined using the general 3D motion model [79]. However, the algorithm uses EM so it can take a long time to converge if the initialisation is not good enough [158]. Sugaya and Kanatani [149] show how outliers due to inaccurate feature tracking can be pruned with RANSAC [51]. Subspace decomposition has also been used for dense motion layer extraction [85].

Vidal and Hartley [161, 163] cast segmentation as a subspace clustering problem as well. Unlike Sugaya and Kanatani [149], they also deal with the issue of missing data. Feature points may disappear and reappear, yielding incomplete trajectories. To deal with these, Vidal and Hartley [161, 163] project the data in a subspace of dimension five via the PowerFactorization method. Since the maximum dimension of each motion subspace is four, as stated in the previous paragraph, linear projection onto a five-dimensional subspace preserves the structure of the motion subspaces. The authors then follow the idea of Zelnik-Manor and Irani [184] to separate motion subspaces via spectral clustering techniques on an affinity matrix, but here affinities are based on the angles between hyperplanes containing each point. They employ a polynomial fitting technique via Generalised Principal Component Analysis [162] to find a set of hyperplanes. The technique is computationally efficient and can deal with both independent and partially dependent motions. However, performances of the fitting method deteriorate for more than three motions [158, 179]. Moreover, factorisation would fail to converge for a long image sequence with only short tracks.

Yan and Pollefeys [179] present a Local Subspace Affinity (LSA) approach. Similar to Vidal and Hartley [161, 163], they use linear projection and spectral clustering [113, 142]. However, instead of a *global* polynomial model, they exploit the fact that neighbouring points tend to lie in the same linear subspace to fit a *local* model to projected points and their nearest neighbours. Locality can be defined practically after projecting points to a unit sphere. Affinity between a pair of points is based on computing principal angles between their respective fitted subspaces. This is better adapted to partially dependent motions than previous measures [184]. Although
local fitting allows for more flexibility than previous methods, misclassifications occur near the intersection of different subspaces. Previous techniques based on factorisation assume that objects undergo rigid motions. However, non-rigid motions can lie in subspaces of dimensions higher than four. To allow efficient processing of these cases, adaptive estimation of the subspace rank which determines the model for projection is proposed by Yan and Pollefeys. This is a difficult problem, more so because the data can be corrupted by noise and outliers. Recent developments on automatic model selection have been proposed to extend LSA. However, in practice, parameters controlling the model selection often have to be tuned by the user.

### 2.2.2 2D Sparse Segmentation

The 3D sparse segmentation techniques we have reviewed heretofore are often limited to segmentation of rigidly moving objects. The quality of the output also deteriorates as the number of objects in the scene increases, as it becomes an increasingly difficult task to estimate this number precisely. On the other hand, 2D sparse segmentation techniques seek to group together trajectories of coherent 2D motion in the image plane. The number of 2D motion groups may not correspond to the number of 3D objects in the scene. These 2D motion-based segmentation approaches have found many applications such as action recognition and surveillance. An example of 2D sparse segmentation is displayed in figure 2.3.

Most 3D segmentation techniques assume completeness of all feature tracks. However, in practice feature point trackers yield trajectories with variable lifespans, that can be corrupted by noise. Key points may also disappear and reappear due to occlusions, new objects entering the scene, as well as illumination and viewpoint changes. To mitigate these problems, the 3D segmentation approach from Rao et al. fills in missing data in shorter trajectories, which can produce erroneous results if the completion is under-constrained.

**Comparing Tracks with Variable Lifespans:** Fradet et al. propose to compare trajectories with variable lifespans on their temporal overlap to avoid resorting to trajectory completion. Their approach uses an intermediate affine trajectory representation of a cluster of points. An affine trajectory contains parameters for backward and forward affine warping at each time instant. Its lifespan is defined as the maximum lifespan for which parameters of an affine warp can be estimated, i.e. as long as at least three trajectories overlap in the cluster. Following an agglomerative approach based on the J-linkage algorithm, trajectories are compared to estimated affine models on the current clustering, and the most similar clusters are merged iteratively. The grouping algorithm starts by randomly selecting sets of three tracks, with sufficient overlap, to oversegment the sequence. To compare a trajectory to an affine motion model, during their time of overlap, a mean residual error is estimated between the warped trajectory and the original trajectory. Robust estimation is achieved by taking the minimum error over all possible starting time instants for warping. The final number of motions is determined autom-
Figure 2.3: Illustration extracted from Brox and Malik [22]. In this example, the motion information to separate the man and the background into two distinct clusters reside in the motion difference: only when the man moves can they be told apart. Long-term feature point tracking ensures that the clustering information is consistent throughout the scene. The labels on the sparse tracks are displayed in the bottom row.

Brox and Malik [22] also compare trajectories of varying lifespans on their common time of overlap. One example result from this technique is displayed in figure 2.3. This bottom-up approach defines a pairwise distance between every pair of tracks that have at least one frame in common. Distance is turned into affinity, and then the affinity matrix is segmented via spectral clustering [52, 113, 142]. The key contribution of Brox and Malik [22] is to define the distance between two trajectories as the maximum difference of their motion over time. This deals with the problem of temporary stopping objects. The authors show the example of a man rising up from his seat (see figure 2.3): no motion cues can tell that person apart from the background until he moves. Therefore, contrary to what Gestalt principles [131] of common fate would suggest, the information does not reside in the common motion but in the motion difference. Pairwise distances can only compare trajectories on the basis of translational motion models. However, translational models are a good approximation for spatially close points only. So, Brox and Malik [22] multiply the motion-based distance between tracks with a spatial normalisation term to ensure only close points can have high affinities. Further normalisation based on the
local Optical Flow variation allows for slow and fast motions to be processed together. Affinities define a graph upon which to run spectral clustering. Due to transitivity of the graph, even tracks that do not share common frames or that are spatially distant can be grouped together. The spectral clustering method proposed includes a spatial regularity constraint to automatically determine the final number of clusters. However, the proposed solution is based on a heuristic combination of several runs of k-means, with many parameters chosen arbitrarily.

**The Use of Occlusions and Depth:** Lezama et al. [101] build on the work of Brox and Malik [22], but change the cost function to introduce occlusion reasoning on trajectories. Relative depth ordering is established between clusters of tracks based on whether occlusions or disocclusions are detected at T-junctions. The added constraint is formulated alongside spatial and motion similarity between tracks in an energy minimisation problem. The energy function is not submodular so Graph Cuts cannot be applied and a solution is found by using Tree-reweighted Message Passing [92]. Limitations of the technique are that the number of labels must be specified by the user, and the depth ordering of objects is assumed to be constant over a video clip.

Fragkiadaki et al. [54] use bottom-up trajectory grouping information to increase the robustness to partial occlusions of a top-down pedestrian tracker. The technique uses an approach similar to Brox and Malik [22] for sparse segmentation. Affinity is defined in a similar fashion, however, a term based on the maximum stereo-disparity difference is also employed between tracks in one view of a stereo video alongside the maximum velocity difference.

Angeli and Davison [3] use depth information obtained from a real-time structure-from-motion algorithm [148], in conjunction with appearance cues, to derive the affinity between pairs of tracked feature points. Their clustering method uses a constrained agglomerative approach based on Medoid Shift [139] and Min-max Cut linkage [43]. Determination of the number of clusters is achieved automatically with the Akaike Infomation Criterion [3]. The method is fast and robust to changes of scale in the scene, but is also sensitive to outliers and needs careful parameter tuning.

Depth information can be extracted from stereo pairs of images and it offers an additional cue which can be combined with position, motion and colour cues to improve the quality of segmentation techniques. Balancing colour and depth cues [3,37] can prove very hard in practice and need manual tuning. In Fragkiadaki et al. [54], stereo-depth information is employed to perform segmentation on one view only, and not to enforce view consistency between the left and right video streams. A general issue with segmentation is that it is an under-constrained problem, so the addition of depth or disparity-based constraints opens new possibilities that have not been explored that much in the literature so far.

**Regrouping Disjoint Tracks:** State-of-the-art feature trackers associate, by design, features that appear and reappear in the video with different tracks, thereby losing important informa-
tion on the long-term motion. Sparse 2D segmentation approaches based on pairwise affinity of overlapping tracks \cite{22,53,54,101} can deal with partial occlusions and tracking loss but break down for fully occluded objects. The problem is tackled by the track repair mechanism presented by Rubinstein and Liu \cite{134}, in the context of feature point tracking. Rubinstein and Liu \cite{134} connect temporally disjoint but similar tracks based on appearance and dynamics models. However, track repair has not been used in the context of sparse segmentation so far. The problem is formulated as a combinational assignment that is defined and optimised globally via Loopy Belief Propagation \cite{49} over the entire sequence. A compatibility measure between disjoint tracks is estimated based on their appearance, motion at the extremities, and predicted position during occlusion. The technique therefore needs video stabilisation as a preprocessing step as camera jitter may introduce arbitrary motions that are difficult to model. A regularisation procedure also enforces neighbouring tracks to be linked to spatiotemporally close candidates, and also penalises links that cross trajectories behind occlusions.

**Why Using Sparse Video Segmentation:** Computer vision approaches to segmentation try to emulate the human visual system. Evidence suggest that motion cues are essential to the way humans can naturally segment objects in a scene. Sparse approaches offer a framework geared towards the analysis of motion interactions between objects in a scene, even though the performance of most techniques still falls far behind human perception \cite{183}.

Promising approaches presented in Fradet et al. \cite{53} and in Fragkiadaki et al. \cite{54} try to combine top-down model-driven approaches with bottom-up grouping-driven approaches. This allows for complex motion models to be used within a sparse framework which is robust to partial occlusions. Indeed, as feature points represent only parts of an object, a major advantage of sparse segmentation approaches is that the object and its properties can still be tracked as long as some feature points attached to it are still visible.

Maintaining temporal consistency from frame to frame, and coherent clusters over many frames are important goals for video segmentation. Indeed, the human visual system is very sensitive to temporal inconsistencies that may occur in a video. Sparse segmentation approaches are promising techniques to achieve these goals in real videos where lighting changes, object occlusions, complex camera motions and sensor noise are commonplace. However, these techniques do not label all pixels in the video. In the next section we review how sparse-to-dense approaches can turn sparse clusters of tracks into dense pixel regions.

### 2.3 Sparse-to-Dense Video Segmentation Approaches

As seen in section \ref{2.1}, state-of-the-art dense segmentation techniques \cite{2} based on frame-to-frame propagation and relaxation using Optical Flow have troubles dealing with partial occlusions, spatially disconnected objects and large displacements. Indeed, these problems are challenging to motion estimation which is at their core. On the other hand, as detailed in section \ref{2.2} the use
of feature point tracking in sparse segmentation techniques \cite{22,182} allows longer-term accurate correspondences between points to be exploited to overcome these issues. However, the ultimate goal of video segmentation is to assign every pixel to a cluster, i.e. generate a dense segmentation of each and every video frame. The most promising techniques to enforce long-term temporal coherence in dense video segmentation use feature point trajectories in a first step, to obtain a sparse representation of objects in the scene. We review in this section these two-step sparse-to-dense techniques. They exploit rich, long-term object position constraints and motion models, that can be obtained from sparse video segments, to perform segmentation by turning clusters of tracks into dense video segments.

### 2.3.1 Methods Based on Short-term Feature Point Associations

In these sparse-to-dense techniques, the use of sparse feature points is limited to pairs of frames or very short clips. They are advantageous for real-time applications where there is no access to the entire video. Even though long-term analysis of trajectories allows for better temporal consistency, these techniques illustrate how feature points can be used as seeds to avoid manual initialisation of dense segmentation methods such as Graph Cuts.

**Region Growing Approaches:** Xiao and Shah \cite{178} propose a two-step approach to multi-layer dense segmentation, by making a distinction between computing layer descriptors and pixel assignment to each layer. During a first step, feature points are tracked over a short time window and a set of affine or projective transformations are estimated between the first and last frames. A patch around each feature in the reference frame is used as a seed to grow a layer region of arbitrary shape. A Graph Cuts method with integrated level set representation is used to expand seed regions. During a second stage, segmentation is performed by analysing the frame difference between a set of consecutive frames in the short time window and the reference frame. Occlusions are detected in these frames and used to constrain the layer assignment process. Occlusion reasoning assumes that the depth ordering remains constant and there is no change of direction in object motions during the time window. The layer assignment is then made by combining the data from pairwise frame differences, and detected occlusions in a multiway Graph Cuts framework using a three-state pixel graph. However, only pairs of frames over short periods of time are considered, and the number of layers is estimated on a reference frame. This could cause temporal inconsistencies in the case of new objects entering the scene or full occlusions.

Wills et al. \cite{173} propose a similar approach to Xiao and Shah \cite{178} which separates layer parameter estimation and dense pixel assignment. They use sparse feature-based motion estimation to increase the robustness of multi-layer segmentation. The method is particularly useful for low frame rate videos, where large displacements are commonplace and Optical Flow is hard to estimate accurately. The technique uses a variant of RANSAC \cite{51} for multiple-motion estimation between pairs of frames using noisy feature correspondences. Layers are merged if
Towards Segmentation of Stereoscopic Videos

their motion is similar. Then a planar homography motion model is estimated, and each pixel is assigned to a candidate layer via Graph Cuts. Forward and backward assignments are computed and occlusions are detected as a mismatch between assignments. However, only pairs of frames are considered so there is a lack of temporal consistency.

Bugeau and Pérez [25] also use a two-step approach, which first clusters feature points via Mean Shift [29] before assigning every pixel to a layer using Graph Cuts. Sparse feature points are selected on moving objects only, by doing a motion detection step to remove points whose motion is only due to the dominant motion. Mean Shift clustering with automatic bandwidth selection then groups together points having similar appearance and motion. The coordinates of points are also used during clustering for spatial smoothness. Finally, dense segmentation is obtained via Graph Cuts using the sparse feature points as seeds. The segmentation is done in each frame independently, so there is no temporal consistency. A method that combines segmentation and tracking has been proposed by the same authors [24] to enforce temporal consistency in their framework. If correspondences are found from frame to frame between two segments, they are assigned the same label. A new label is created if there is no correspondence. The approach is robust to partial occlusions but not to full occlusions, and loses objects when their motion is temporary close to the motion of the background.

Monoscopic versus Stereoscopic Approaches: Klappstein et al. [86] also present a method using sparse tracked feature points as seeds for Graph Cuts segmentation. Their goal is to detect and segment moving objects on the road, viewed through cameras inside a car, for a driver assistance system. An interesting aspect of this paper is the comparison of monoscopic and stereoscopic computer vision approaches. For an embedded system, using only one camera has the advantage of being cheaper and less cumbersome. With two cameras, range measurements can be readily obtained, but calibration errors can become an issue.

In both cases, dense segmentation is performed via a metric that indicates how much the motion of a point, belonging to an object in the scene, is different from the ego-motion of the vehicle. In stereo vision, 3D coordinates and 3D velocities can be computed accurately on both static and dynamic objects. For each tracked point, the computation fuses position, disparity and Optical Flow information over time, using a Kalman filter [76]. In mono vision, 3D reconstruction is accurate only for static points. So the metric employed for segmentation estimates the reconstruction error based on a set of constraints, using two and three frames. The later is known as the trifocal constraint and it has been exploited previously for 3D sparse motion segmentation [65].

The stereoscopic approach is shown to outperform the monoscopic approach in terms of accuracy, but at the expense of a higher computational cost. The technique can benefit from using longer-term feature tracks, and a more complex segmentation method should be used to separate objects moving in different directions.
2.3. Sparse-to-Dense Video Segmentation Approaches

Figure 2.4: Illustration extracted from Baugh and Kokaram [11]. The sparse segmentation obtained with this technique is displayed in the top row. Together with user-input mattes on key frames, motion and position information from the sparse labelling are used to obtain the dense segmentation displayed in the bottom row. Several sparse clusters can be merged into one dense segment exhibiting complex and articulated motions.

2.3.2 Methods Based on Long-term Feature Point Tracks

Feature point trajectories have emerged as a promising means to attain high quality and unsupervised dense video segmentation. They allow sparse-to-dense approaches to exploit the long-term motion characteristics of objects. These techniques can generate dense segments that represent an object for many frames, however, user interaction is still often required to correct mis-matching around object borders. An example of sparse-to-dense segmentation based on long-term tracking can be seen in figure 2.4.

Constrained Label Assignment via 2D Sparse Segmentation: One of the first techniques to propose a sparse-to-dense approach to video segmentation is Liu et al. [103]. The method is used as a preprocessing step for layer-based content manipulation, the final application being to magnify small motions in a sequence. During the sparse segmentation step, normalised correlation between trajectories is used to compute affinities. Spectral clustering via Normalised Cuts [141] groups tracks having motions generated by a common cause into a predefined number of clusters. A dense Optic Flow field is interpolated for each cluster via locally weighted linear regression [135] and dense segmentation follows. The dense problem is cast in an energy minimisation framework which exploits motion, colour and spatial cues together. A global solution is estimated sequentially for each frame with Graph Cuts [21]. The dense layered representation of the video is created by assigning every pixel in every frame to one of the motion clusters defined during sparse clustering. As the layer assignment is done for each frame independently, differences may arise from frame to frame, which requires user corrections.
Moreover, the definition of a layer in Liu et al. [103] is very rigid, keeping a fixed colour for each pixel along its trajectory.

Baugh and Kokaram [11] use long-term point trajectories to guide a dense segmentation process. This technique is illustrated in figure 2.4. Both spatial locations of sparse feature points and motion models obtained from each cluster are exploited in a Bayesian framework. Sparse segmentation is initialised with Mean Shift, which allows to find the number of objects automatically and estimate initial motion models for each cluster. Then, the proposed solution iterates between Graph Cuts and a merging step based on the similarity between motion models to obtain the final sparse labelling. The dense segmentation step exploits prior information from labelled tracks and user-drawn mattes on key frames to increase spatiotemporal smoothness of the result.

Lezama et al. [101] extend the graph-based dense video segmentation technique of Grundmann et al. [62] by incorporating to it constraints from sparse segmentation of feature point tracks. This sparse-to-dense approach overcomes limitations due to analysis at a local level in the dense technique. However, the dense step is sensitive to errors in the sparse track labelling at object boundaries.

**Label Propagation after 2D Sparse Segmentation:** Ochs and Brox [116] introduce a hierarchical variational approach for propagating the labels of feature point trajectories to dense image regions. The sparse segmentation step follows Brox and Malik [22]. This step generates labels for a set of tracks with a high temporal coherence. However, there are few or erroneous points in homogeneous regions with no features to match. Colour and edge information is complementary to the motion cues used during sparse segmentation. Specifically, colour-based image segmentation such as Mean Shift [29] works the best in homogeneous areas. Ochs and Brox [116] exploit this complementary data to propagate labels to every pixel in the video, via a variational diffusion approach based on Lellmann et al. [100]. For each frame in the shot, a hierarchy of superpixel partitioning is first constructed. Superpixels are piecewise constant image regions which are represented by their mean colour. Generally, superpixels are used to create an edge-aware oversegmentation of a video [160], and need further processing or user intervention to form coherent object regions. Here, the original sparse labelling is used as a source to spread label information within superpixels. Inaccurate track clustering can be corrected to a certain extent but errors due to misclassifications at the sparse level can propagate to the dense level. Although the labels are temporally coherent in the long term, there is no short-term temporal smoothness in the dense process, which can cause flickering of the segments from frame to frame.

**Remaining Challenges for Sparse-to-Dense Approaches:** When turning sparse clusters into dense regions, the techniques studied in this section can use the higher-level information given by a cluster of points to recover from local errors while assigning temporally coherent labels to every pixel in a video [116]. Temporal coherence has drawn a high amount of attention in
the literature, but coherence of the segmentation across views for segmentation of stereoscopic videos is still to be explored. Some works, in particular Klappstein et al. [86], have already shown the interest of using two views to increase the quality of extracted dense layers.

Sparse point trajectories do not cover some of the most critical areas in a scene, especially near occlusions due to object boundaries and into low-textured regions. And even if there are points tracked in these regions, their trajectories are often noisy, which can cause inaccurate labelling with sparse segmentation methods. These errors can be corrected to some extent, but can also be propagated to the dense segments. These errors would then have to be corrected manually by the user in the dense result. The next section reviews how the user is kept in the processing loop by supervised segmentation approaches.

2.4 Supervised Video Segmentation Systems

A common problem in most state-of-the-art segmentation techniques is the lack of high-level understanding of a video scene. Defining what is an object is not trivial and each object could be represented by one or several clusters of pixels. An articulated object, for instance an actor, could be segmented into several parts exhibiting different motions, for instance head, arms, torso. A valid segmentation could be to group the actor into one single object, which could be useful for compositing applications [32]. Each body part could also be assigned a separate label, which could be useful for surveillance applications [25]. Ill-posed problems must be solved when seeking a fully automated technique to set the level of segmentation.

The video segmentation techniques reviewed so far are automatic or require minimal direct user intervention during initialisation or for parameter tuning. However, applications in cinema post-production for instance, require a very precise matte for video compositing which cannot be obtained in practice by these techniques. Automatic approaches are often used as an initialisation, and manual corrections of the generated mattes are required in a later stage [89]. Temporal coherence of the segments is important in this case to limit the amount of subsequent user intervention.

From video data, segmentation techniques extract motion, colour, texture or edges features and exploit them to group pixels into coherent motion or appearance regions. Fully automatic techniques can therefore provide at most a mid-level representation of a video as stated by Wang et al. [164]. But a solution to high-level segmentation in a fully automatic manner is still elusive to date. Indeed, bottom-up approaches have no notion of what constitutes an object, which can be made of several articulated or non-rigid parts, that may have inconsistent motion and appearance over time. In other words, an object perceived as a whole by the human brain can contain several video segments. Achieving object-level segmentation needs to bridge a semantic gap, which can only be done through user input by current state-of-the-art approaches. In this section we review methods that keep the user in the loop to attain high quality object-level video segmentation in difficult real-world sequences.
2.4.1 Tightly-Supervised Video Matting

Depending on the effort required by the user we distinguish between tightly-supervised and loosely-supervised approaches. The former include interactive techniques and require a more intensive and continuous user intervention than the latter. One of the fall-back solutions to dense video segmentation in the cinema industry has been to use manual rotoscoping to cut out foreground objects from a scene, which is highly time consuming. Otherwise, a constrained segmentation problem can also be solved by blue or green screen keying. The keying solution does not need as much user intervention during post-production but the scene needs to be captured in a constrained environment, which is not always feasible.

Image-based Approaches: Interactive segmentation approaches have been designed first to segment still images. In these methods, the user is asked to draw a few scribbles into the image to mark foreground and background regions. A trimap can also be provided, on which an additional label is assigned to unknown regions. Then the approach automatically propagates these labels into the non-marked areas. Several techniques based on Graph Cuts [19] or Random Walks [58] have been designed according to this idea. Unlike labels obtained from unsupervised sparse segmentation, which can be erroneous, and must be used with caution by sparse-to-dense approaches [116], these user-generated scribbles can be considered as dependable. Processing videos frame by frame with such techniques would generate temporal inconsistencies, particularly along object edges, and would require tedious user interaction. It has been shown that using motion information can reduce user interaction and decrease inconsistencies [5, 32, 132].

Video-based Approaches: Users are often asked to supply reference mattes on key frames, to provide the objects of interest to the system. These key frames can be used as constraints in a top-down approach to discriminate better between low-level motion and appearance features. For instance, the semi-automatic segmentation approach proposed by Baugh et al. [11] uses mattes drawn by the user every 10 to 20 frames to propagate colour and motion models corresponding to objects. Video SnapCut [5] uses local motion information from Optical Flow and SIFT [106] feature matching to propagate appearance information from a segmented frame to the next unsegmented frame. Appearance is captured by local classifiers in overlapping rectangular windows along the contour of the foreground. Graph Cuts is then used to solve the matting problem posed as MAP estimation. The user input in this case is a full segmentation of the first frame and corrections along the propagation. Overall, the technique performs well but has issues with disocclusions in the background. Feature-Cut [132] is a similar supervised approach. The algorithm uses SIFT [106] feature matches between frames to propagate a collection of user-segmented binary mattes from key frames to unsegmented frames. The final matte is generated in a MAP-MRF framework solved via Graph Cuts. Appearance is modelled as intensity of matched pixels in SIFT neighbourhoods. Temporal consistency issues can be seen in the output due to frame by frame propagation. Also the technique struggles with low-textured
images as there would not be enough SIFT feature matches to propagate the matte.

2.4.2 Loosely-Supervised Video Segmentation

Fully automated segmentation techniques need to infer the number of objects in the video, this is related to defining the level of segmentation in the output. It is a difficult problem that requires high-level understanding of the scene. Many automatic solutions have been proposed such as using Mean Shift [11, 25], Minimum Description Length [40], the Akaike Information Criterion [3, 77], distributions of the model hyperparameters [81, 82], or spectral graph theory theorems [182]. But none can predict what the actual number of clusters desired by the user is, which ultimately depends on the targeted application. For instance, only two clusters are necessary for video matting [86, 89]. On the other hand, oversegmentation is acceptable in some applications [2, 62], as long as it provides temporally coherent associations while respecting object boundaries. As determining the number of segments in a scene is an ill-posed problem, many techniques [101, 103, 149] rely on the user to provide that information. For instance, Liu et al. [103] let the user set the number of clusters so that the output gives a sensible representation of the motion in the scene. Instead of segmenting a scene at a fixed level, a versatile technique would ideally output a hierarchy of clusters so that the level of fragmentation can be increased or diminished by the user, depending on their needs, in a loosely-supervised way.

Applications to Video Cartoonisation: Semi-automatic video segmentation has found interesting applications in non-photorealistic rendering approaches. For instance, Video Tooning, presented by Wang et al. [167], stylises a real-world video to look like a cartoon. An automatic video segmentation technique based on anisotropic kernel Mean Shift [166] is at the core of this technique. The approach segments the video into continuous regions of similar colour in the space-time volume as a whole. This avoids the common robustness problems of segmentation around object boundaries related to motion estimation, but it requires a significant amount of memory as all video data must be processed at once. However, segments derived from the Mean Shift procedure are typically too low-level for the final cartoonisation application. Therefore, the user is asked to merge segments together on key frames, every 10 to 15 frames, to provide a higher-level semantic grouping. Then, smooth trajectories of the semantic clusters of low-level segments, created by the user, are interpolated between key frames. Jagged edges are filtered out to avoid temporal artifacts. In addition to the space-time surface geometry, each semantic region is annotated with a colour and an edge importance supplied by the user to determine how they should be rendered.

A related approach for video cartoonisation has been presented by Collomosse et al. [28] in the Video Paintbox system. In this technique, all video frames are first segmented independently into regions of homogeneous colours using Mean Shift [29], yielding an oversegmentation of each frame. Then, associations between regions are formed between adjacent frames heuristically, based on spatial proximity as well as colour and shape similarity. Links between regions over
Figure 2.5: Illustration extracted from DeCarlo and Santella [41]. This figure shows the use of loosely-supervised image segmentation for stylisation of photographs. The level of detail of colour segments on the output is increased on the areas of the image where the eyes of the viewer are the more attracted (e.g. faces).

Hierarchical Graph-based Segmentation: Grundmann et al. [62] extend a hierarchical graph-based image segmentation technique [50] to handle videos. In the original image segmentation technique, pixels are modelled as nodes in a graph, and edge weights are derived from colour differences. Nodes are merged into image regions by traversing the edges and evaluating whether the edge weights are smaller than a local threshold. The threshold is estimated accord-
ing to local image texture. The proposed extension to process videos define a spatiotemporal neighbourhood for each pixel. In the temporal dimension, neighbours are located along motion trajectories obtained by Optical Flow. Various segmentation levels ranging from oversegmentation at the finest level to undersegmentation at the coarsest level are obtained by varying the local thresholds. The user chooses the segmentation level by moving along a tree of solutions. Only a global level of segmentation can be selected, whereas some objects of interest would need to be segmented at a finer level than the background for instance, as in DeCarlo and Santella [41]. Implementation of Grundmann et al. [62] requires sophisticated work-arounds for computational memory issues. Temporal incoherence can occur along time as the technique mainly relies on colour information to derive video segments.

Remaining Challenges for Supervised Segmentation Approaches: No supervised techniques dealing with stereoscopic videos exist to date. When processing multiple-view videos, the amount of user interaction should not be proportional to the number of views. As the left and right view streams in a stereo video are very similar, it would be desirable to exploit geometric properties to allow a joint processing of both views that preserves the stereo coherence of objects. A supervised stereo image segmentation that works along these lines has been proposed by Tasli and Alatan [155], but it cannot be used to segment videos as it is. Lastly, note that most supervised methods described in this section require user interaction at the dense level. There is a lack of research on the influence of user interaction at the sparse level, before a sparse-to-dense approach assigns labels to every pixel.

2.5 Final Remarks

One of the first choices that has to be made when developing a video segmentation algorithm is the representation of the scene. Dense approaches described in section 2.1 assign a label to every pixel in the sequence. On the other hand, sparse approaches reviewed in section 2.2 detect and track salient feature points throughout the image sequence, before operating a motion-based clustering of these points. Dense approaches yield a more precise segmentation but suffer more from problems due to occlusions and large displacements. Moreover, they generally lack spatial consistency from frame to frame, and temporal consistency in the long term. Sparse segmentation methods efficiently exploit long-term motion information to produce temporally coherent clusters, however, they only generate a partial labelling that can miss details in the scene. These techniques are therefore convenient for applications that do not require an accurate extraction of object boundaries such as object detection, action recognition, tracking, or surveillance.

However, many applications, such as video compositing for cinema post-production, require a high-quality, temporally consistent dense segmentation of the scene. Segmenting feature point trajectories tracked throughout a video sequence into independent motion clusters is used as a first step to dense segmentation by sparse-to-dense approaches that we review in section 2.3.
These techniques try and improve the temporal coherence of dense approaches by exploiting long-term associations of point trajectories from the sparse representation. A robust solution remains elusive for segmenting sequences with complex dynamic scenes containing lighting changes, and both camera and multiple object motions, especially non-rigid motions. Furthermore, high-level information about the number of objects in a scene or semantic associations between clusters are missing and very hard to infer by fully automatic methods.

As evidenced in section 2.4, creation of the best segmentation results tend to require a combination of low-level image processing tools and high-level user-interaction. A tight control on the segmentation can be imposed by letting the user draw rough mattes on key frames. In these techniques, high quality is wanted in terms of object delineation, as ultimately a continuous soft matte is desired. Soft mattes account for ambiguous pixels at object boundaries, and semi-transparent areas such as shadows and reflections. This can be automatically deduced from a binary hard matte created by supervised segmentation. A looser control on the output can be preferable for different applications such as video cartoonisation. In these techniques, a meaningful segmentation is desired. In general, a hierarchy of clusters is constructed automatically and user input is required to refine some segments by manipulating a tree structure.

In the context of stereo vision or stereoscopic cinema post-production, coherence of the segmentation across views should be enforced as well as temporal coherence, because a cluster should correspond to a physical object that is imaged on both views. Left and right video streams of a stereo sequence are very similar. The main differences between views arise in binocular half-occlusion areas which are hidden in only one view and should be segmented coherently with respect to the rest of the pixels. Depth information can be extracted from stereo pairs of images and it offers an additional cue which can be combined with position, motion and colour cues to improve the quality of segmentation techniques. A general issue with segmentation is that it is an under-constrained problem, so the addition of depth or disparity-based constraints opens new possibilities that have not been explored that much in the literature so far.

In the next chapter, we present our work on sparse segmentation for stereoscopic videos that takes into account view consistency of the generated segments. Our algorithm does not yield a dense labelling but this could be attained by one of the methods described in section 2.3. We design our method to be a stepping stone to enforce both long-term temporal consistency and stereo consistency across views for further applications.
User-Assisted Sparse Stereo-Video Segmentation

In the previous chapter we have discussed how the problem of segmentation is solved by state-of-the-art techniques. Firstly, we have made a distinction between sparse and dense approaches. Then, we have shown that sparse-to-dense segmentation techniques are very promising as they can combine the strength of both approaches. Finally, we have reviewed how user interaction can be used to improve the usability of the output obtained by automatic segmentation.

In this chapter, we present a user-assisted sparse segmentation algorithm for stereoscopic videos. Our goal is to maintain temporal coherence and view consistency of the segments throughout the video. Our technique builds on the automatic sparse segmentation technique for monoscopic videos presented by Brox and Malik [22]. The first step of our technique is to track feature points on both views with a generic tracker. Then, the affinity between pairs of tracks is analysed so as to cluster similar tracks together.

As explained in section 2.2, most sparse video segmentation approaches rely on the computation of a pairwise affinity matrix, which allows to compare feature point tracks. Usually, spectral clustering methods [142] are then employed to find coherent groups of tracks based on this affinity matrix. State-of-the-art techniques [22,53] allow tracks to have different lengths and compute the affinity between temporally overlapping tracks. This maintains a coherent segmentation even if objects are partially occluded, but not if they become fully occluded.

In Rubinstein and Liu [134], a track repair mechanism is proposed to connect similar tracks that are temporally disjoint, because of occlusions or tracking loss. This technique is used to increase the length of feature point trajectories for tracking applications. We incorporate this idea in our segmentation technique, by exploiting the affinity of both temporally overlapping
Figure 3.1: Example of sparse stereo segmentation on a stereo-pair of frames from the green_sscreen sequence, in the Sigmedia stereo video database [31]. Feature points on the actor can form a single cluster (middle row) or several clusters corresponding to the different motions of his head, torso and arms (bottom row). Position history of the trajectories over the past five frames is indicated with lines that extend from the corresponding dot marking the position of the feature point on the current frame. To observe the tracks, we invite the reader to watch this video online at http://www.sigmedia.tv/Misc/ResultsThesisFelix.

and disjoint tracks. This reduces the problems caused by tracking loss and full occlusions in sparse segmentation approaches.

In segmentation techniques for monoscopic videos, occlusion reasoning can be used to infer the relative depth of objects in a scene [101,178]. In stereoscopic videos, however, there are ways of estimating the actual depth of objects, so it is more natural to exploit this information. Previous techniques [54,86] use this depth information as a cue that is exploited during clustering on one view. However, in these techniques, consistency of the segmentation across views has not
been considered. In our method, we are interested in processing both views in a stereo video jointly. We achieve this by using stereo-disparity to map feature tracks from one view to the other, so as to compute the affinity between tracks residing in different views.

A common problem in most state-of-the-art segmentation techniques is the lack of high-level understanding of a scene. We acknowledge that a segmentation system is not useful if the user cannot interact with it. First, our technique generates a decision tree with a standard automatic clustering technique. The user can then quickly navigate through the tree to determine the final segmentation output.

The output of our technique is illustrated on one example in figure 3.1. One stereo-frame from the video is shown in the first row from the top. The feature points obtained after tracking are shown in the second row. For some application, one cluster could be assigned to all points belonging to the actor. In the third row, we show the final segmentation after user interaction. A different cluster is created for the head, the arms and the torso of the actor.

The following sections delve into the details of our approach. Firstly, in section 3.1 we introduce basic definitions. Then, in section 3.2 we present necessary preprocessing steps to obtain the feature tracks. Section 3.3 explains how affinities are then computed between pairs of tracks. Finally, section 3.4 details our clustering method, which is based on an automatic split-and-merge approach followed by a user-assisted refinement step.

3.1 Basic Definitions

A stereo video is made of two views, left $L$ and right $R$. Our segmentation technique uses a sparse motion-oriented representation of the picture data in the stereo video. This representation is made of $N_t \approx 5000$ trajectories or tracks, on each view $v \in \{L, R\}$. Trajectories are noted $T^v_i$, with $i \in \{1, \ldots, N_L\}$ if $v = L$, and $i \in \{N_L + 1, \ldots, N_L + N_R\}$ if $v = R$. They represent the temporal evolution of the position of corresponding feature points. Figure 3.1 shows some of the tracked feature points in a stereo video.

Each track $T^v_i$ begins at frame $f^v_{ib}$ and ends at frame $f^v_{ie}$. At each frame $f \in \{f^v_{ib}, \ldots, f^v_{ie}\}$, the trajectory contains the spatial coordinates $x^v_i(f)$ and $y^v_i(f)$, and the stereo-disparity value $d^v_i(f)$ on the $x$ axis for the corresponding feature point. Therefore, the position vector along the trajectory at frame $f$ is noted:

$$T^v_i(f) = [x^v_i(f), y^v_i(f), d^v_i(f)]$$ (3.1)

At each frame along the trajectory, we thus obtain a 3D representation of the feature point. According to Duan et al. [44], using the disparity information as a feature favours clustering of tracks within the same depth layer.

Following state-of-the-art methods such as Brox and Malik [22], our sparse segmentation technique uses the motion of feature points as a cue for clustering. The velocity of a feature point is defined as the temporal derivative of its position. We extend the definition to stereo
videos by including the temporal derivative of the disparity in the definition. At a given frame $f$, the velocity of the feature point $i$ in view $v$ is noted as follows:

$$\frac{\partial T^v_i}{\partial t} (f) = \left[ \frac{\partial x^v_i}{\partial t} (f), \frac{\partial y^v_i}{\partial t} (f), \frac{\partial d^v_i}{\partial t} (f) \right]$$ (3.2)

Our sparse segmentation technique needs to be able to compare the motion of feature points that can reside in different views. To make this possible, we map a trajectory $T^v_i$ in view $v$ to the other view $w$, using the stereo-disparity. We then obtain the stereo-counterpart of this trajectory, noted $T^{w\rightarrow v}_i$. At each frame $f$ along the trajectory, the stereo-counterpart position vector is defined as follows:

$$T^{w\rightarrow v}_i (f) = \left[ x_{i}^v (f) + d_{i}^v (f), y_{i}^v (f), -d_{i}^v (f) \right]$$ (3.3)

Computation of the stereo-counterpart velocity vector $\frac{\partial T^{w\rightarrow v}_i}{\partial t} (f)$ is then derived from the stereo-counterpart position vector. For convenience in the definitions employed later on, we define $T^{v\rightarrow v}_i (f) = T^v_i (f)$.

Given a pair of tracks $i$ and $j$, which can be in any view, we explain in section 3.3 how to compute the pairwise affinity $A(i,j)$. In section 3.4 we then explain how similar tracks are clustered together via spectral analysis of the affinity matrix $A$, followed by user-assisted refinement. However, before going any further, in section 3.2 we present some preprocessing steps that are necessary to obtain the feature point trajectories.

### 3.2 Preprocessing Steps

In this section, we detail three preprocessing steps needed to compute our sparse feature representation of a stereo video. First, we use feature point tracking and stereo-disparity estimation to compute the trajectories, on each view separately. Then, we temporally smooth out the raw velocity and disparity values so as to remove the influence of camera motion and remove the influence of erroneous values. Finally, we only keep non-static points for further clustering after a bi-layer segmentation step. After these steps, in section 3.3 we detail the computation of the affinity matrix.

#### 3.2.1 Estimation of the Feature Point Trajectories

To generate feature point tracks $T^v_i$ on a stereo video, we employ the standard KLT feature tracker \[143\] on each view independently. We use the C++ implementation provided by Stan Brichfield\[1\]. Detected feature points can be observed on one frame of a stereo video in figure 3.1. Very short tracks of length lower than five frames are discarded. Stereo-disparity information

---

\[1\]http://www.ces.clemson.edu/~stb/klt/
3.2. Preprocessing Steps

is computed independently via the disparity estimator available in Ocula\textsuperscript{2}, a suite of stereoprocessing plug-ins for Nuke\textsuperscript{3}.

A stereo feature tracker\textsuperscript{114} that simultaneously computes feature point displacement in both views could also be used advantageously in our technique. However, no feature points would be tracked on binocular half occlusion areas, i.e. points visible in one view only. Our approach does not guarantee that corresponding feature points are tracked on both views. However, it has the advantage of reusing an existing standard tracker developed for monoscopic videos, without requiring a special treatment for half-occluded feature points. Some stereo-segmentation algorithms such as Dal Mutto et al.\textsuperscript{37} discard points that are occluded on one view only, and rely on additional processing to assign them to a cluster.

We also estimate the local Optical Flow field with Nuke, so as to discard the most noisy feature tracks. These tracks exhibit a motion which is drastically different from corresponding local motion vectors. We compute the velocity along trajectories, using backward frame-to-frame difference of the position vector. For each frame along the trajectory, we then compute the minimum error between the velocity and the estimated local backward motion field in a 3x3 patch around the feature point position. We then keep the maximum error along time for each track. Then we compute the mean and standard deviation of the error for all tracks. And finally, we consider as inliers the tracks for which the absolute value between the error and the mean is below the standard deviation.

3.2.2 Temporal Smoothing of the Velocity and Disparity Values

The estimated position of a feature point along its trajectory can be inaccurate and cross disparity layer boundaries, thereby causing spurious fluctuations of the disparity value. To lower these fluctuations we apply a three-tap temporal median smoothing filter on the disparity values along trajectories, using the previous and next frames.

Feature point velocities $\frac{\partial T}{\partial t}$ are preprocessed so as to remove the influence of the camera motion jitter. This step is similar to the use of video stabilisation in the track repair mechanism presented by Rubinstein and Liu\textsuperscript{134}. Indeed, in our technique, velocities of temporally disjoint tracks must be comparable by the affinity measure we detail in section 3.3. They would not be comparable if they are corrupted by motion jitter, e.g. when the camera is handheld. Consider a sequence in which the camera moves quickly up and down during a limited amount of time and then remains perfectly static. Disjoint tracks on the same object, but existing during and after the frame interval corrupted by camera motion noise would have different apparent motion behaviours, so they could be misclassified as belonging to different objects.

Our procedure to attenuate the influence of camera motion noise is based on the video stabilisation technique presented in appendix A\textsuperscript{4} applied to each view separately. First, the

\textsuperscript{2}version 3.0 – https://www.thefoundry.co.uk/products/ocula/
\textsuperscript{3}version 6.3 – https://www.thefoundry.co.uk/products/nuke-product-family/
Dominant Motion Vector (DMV) is estimated at each frame on the first two components of the velocities of tracked feature points. The DMV is then subtracted from all velocity values, on each view separately. Finally, the registered velocity values are temporally smoothed out with a two-pass, forward and backward, one-tap IIR filter with a parameter value of 0.75. This low-pass filtering step increases robustness in subsequent computation in case the camera motion noise is not estimated accurately enough.

### 3.2.3 Background Subtraction via Bi-layer Segmentation

Once velocities \( \frac{\partial T}{\partial t} \) are registered and filtered, we perform a basic bi-layer segmentation to label points that are moving due to camera motion only as background and moving objects as foreground. Figure 3.2 illustrates the result of this bi-layer segmentation step on one example. Tracks coloured in yellow are classified as static background and tracks coloured in white are classified as dynamic foreground. Velocities are registered to the camera motion as explained in the previous paragraph. Therefore, tracks belonging to the background have a registered velocity that remains close to zero along time. We detect them by a simple thresholding on the motion amplitude of the first two components of the velocity. If the motion amplitude remains below 2 pixels in every frame along the trajectory, we classify the track as belonging to the background.

Static tracks belonging to the background are discarded in further processing, as we are mainly interested in segmenting dynamic objects in a scene. However, we do not discard any background tracks if the amount of background trajectories is less than 50% of the total number of tracks, so as to avoid inaccurate background labelling. This can happen under camera zooming for instance, as our motion registration step is translational only. We found this bi-layer pre-segmentation step useful to reduce computation speed and balance the size of the clusters, as the presence of a very large cluster can bias some clustering techniques. After this step, we...
3.3 Computation of the Affinity Matrix

Following Gestalt psychology principles [131], sparse segmentation techniques seek to group together feature points exhibiting a similar motion and a close proximity in space. To do so, the first step of our approach is to compute pairwise affinities $A(i, j)$ between tracks. Affinities ex-
exploit feature point trajectories $T^v_i$ and velocities $\frac{\partial T^v_i}{\partial t}$, as well as their stereo-counterparts $T^{w\rightarrow v}_i$ and $\frac{\partial T^{w\rightarrow v}_i}{\partial t}$. The affinity values are gathered in an affinity matrix noted $A$. This section explains how this matrix is computed. In section 3.4 we then detail how the affinity matrix is subsequently analysed to form clusters of tracks.

3.3.1 Comparison of Overlapping and Disjoint Tracks in any View

As stated in chapter 2, previous video segmentation methods compute the pairwise affinity between overlapping tracks, using samples from the trajectories and velocities taken only at frames for which both tracks exist. For example, in figure 3.3 tracks 1 and 2 in the left view are overlapping during three frames, from frame 3 to frame 5.

A key novelty of our method is to compute affinities between disjoint tracks as well, using samples from the trajectories and velocities taken at the extremity of each track. For instance, in figure 3.3 tracks 3 and 4 in the right view are disjoint. Track 3 stops before track 4 begins, so we use the last three frames at the end of track 3, from frame 2 to frame 4, and the first three frames at the beginning of track 4, from frame 6 to frame 8. The number of samples used at the extremities of disjoint tracks is a parameter that we note $n_f$. In this example $n_f = 3$ frames, however, in our experiments $n_f = 4$ frames.

In our technique, pairwise affinities are computed between each and every pair of tracks, regardless of the view in which they reside. We use stereo-disparity to process jointly feature tracks on both views. In case both tracks do not lie in the same view, a key point of our work is to map one of the tracks to the other view, as explained in section 3.1. Our technique is able to compare these tracks by adding corresponding disparity offsets to the position of all tracked features points. For instance, in figure 3.3, we map track 5 in the left view to its stereo-counterpart in the right view so as to compare it to track 3 and 4 in the right view.

3.3.2 Computation of the Pairwise Distance Matrix

Following the framework of previous 2D sparse segmentation methods [22,101,182] we perform track clustering on a pairwise affinity matrix indicating how similar tracks are to each other. In our technique, the affinity matrix is derived from a pairwise distance matrix $D$, which is made of two terms:

$$D(i,j) = \left( 1 + \frac{D_s(i,j) - m_s}{\sigma_s} \right) \left( \frac{D_t(i,j) - m_t}{\sigma_t} \right)$$

For a given pair of tracks $(i, j)$, $D_s(i,j)$ is the pairwise spatial distance and $D_t(i,j)$ is the pairwise motion distance. The spatial distance is based on the position values of the trajectories $T^v_i$, $T^v_j$ and their stereo-counterparts $T^{w\rightarrow v}_i$, $T^{w\rightarrow v}_j$. The motion distance is based on the velocity values $\frac{\partial T^v_i}{\partial t}$, $\frac{\partial T^v_j}{\partial t}$ and their stereo-counterparts $\frac{\partial T^{w\rightarrow v}_i}{\partial t}$, $\frac{\partial T^{w\rightarrow v}_j}{\partial t}$. In the equation above, $m_s$ is the minimum value of $D_s$, and $\sigma_s$ is the standard deviation of the values in $D_s$. The values of $m_t$ and $\sigma_t$ are defined similarly for $D_t$. 
3.3. Computation of the Affinity Matrix

As stated by Brox and Malik [22], pairwise track distances can only compare the compatibility of trajectories on the basis of translational motion models. Such low-complexity models are valid only for tracks that are close enough in space, for which \(D_s(i, j)\) is low. In our formulation, we add one to the leftmost term in equation 3.4 so that tracks for which the spatial distance is null have a low distance only if their motion distance \(D_t(i, j)\) is low. This is necessary as disjoint tracks may have a low spatial distance but a high motion distance.

### 3.3.3 Computation of the Spatial Distance

In this section we explain the computation of the spatial distance \(D_s(i, j)\) used in equation 3.4, given a pair of tracks \((i, j)\). When the tracks belong to different views, the distance is set as the average comparison value between the two possible stereo-mappings. If the tracks are overlapping between frame \(f_b\) and frame \(f_e\), the distance is computed as follows:

\[
D_s(i, j) = \frac{1}{2(f_e - f_b + 1)} \sum_{f \in \{f_b, ..., f_e\}} \| T^v_i(f) - T^{w-v}_j(f) \|_2 + \| T^{v-w}_i(f) - T^w_j(f) \|_2 \tag{3.5}
\]

If the tracks are disjoint, let us assume that track \(i\) ends before track \(j\) begins. The distance is computed similarly if track \(j\) ends before track \(i\) begins, reversing the roles of each track. We note \(f^i_k\) the last frame of track \(i\), and \(f^j_k\) the first frame of track \(j\). In this case, the distance is computed as follows:

\[
D_s(i, j) = \frac{1}{2n_f} \left\| \sum_{f \in \{f^i_{k-n_f+1}, ..., f^i_k\}} T^v_i(f) - \sum_{f \in \{f^j_{k-n_f+1}, ..., f^j_k\}} T^{w-v}_j(f) \right\|_2 + \frac{1}{2n_f} \left\| \sum_{f \in \{f^i_{k-n_f+1}, ..., f^i_k\}} T^{v-w}_i(f) - \sum_{f \in \{f^j_{k-n_f+1}, ..., f^j_k\}} T^w_j(f) \right\|_2 \tag{3.6}
\]

This spatial distance ensures that only overlapping tracks which remain close to each other on average along their common lifespan or disjoint tracks which reside nearby at their extremities can yield a low spatial distance value. Note that points that are far apart can still be grouped in the same cluster by our clustering process via transitivity of the affinity graph.

### 3.3.4 Computation of the Motion Distance

In this section we explain the computation of the motion distance \(D_t(i, j)\) used in equation 3.4, given a pair of tracks \((i, j)\). If the tracks are overlapping between frame \(f_b\) and frame \(f_e\), the distance is computed as follows:

\[
D_t(i, j) = \frac{1}{2} \max_{f \in \{f_b, ..., f_e\}} \left\| \frac{\partial T^v_i}{\partial t}(f) - \frac{\partial T^{v-w}_j}{\partial t}(f) \right\|_2 + \frac{1}{2} \max_{f \in \{f_b, ..., f_e\}} \left\| \frac{\partial T^{v-w}_i}{\partial t}(f) - \frac{\partial T^w_j}{\partial t}(f) \right\|_2 \tag{3.7}
\]

If the tracks are disjoint, the distance is computed similarly to \(D_s(i, j)\), in equation 3.6 but replacing the position vectors by the respective velocities.
This motion distance follows the formulation of Brox and Malik [22] for overlapping tracks, with the addition of the disparity term. We also assign a low motion distance to disjoint tracks which exhibit a similar motion on average at their extremities. Note that only stationary disjoint tracks would legitimately have a low distance in our formulation. It is important to repeat that the velocity values computed for disjoint tracks assume that the effects of camera jitter are removed, following the idea presented by Rubinstein and Liu [134] for track repair. As explained in section 3.2, this is achieved during preprocessing.

3.3.5 Computation of the Temporal Decay

The approximation of stationarity made in the previous section for computation of the motion distance of disjoint tracks holds only for tracks that are not too far apart in time, so we define an exponential decay factor $\alpha(i, j)$, given a scale parameter $\sigma_\alpha = 100$ frames, to lower the affinity of disjoint tracks as the length of the temporal gap between them widens.

If tracks $i$ and $j$ are overlapping, there is no decay, and $\alpha(i, j) = 1$. Let us now assume that the tracks are disjoint and that track $i$ ends before track $j$ begins. The computation is similar if track $j$ ends before track $i$ begins, reversing the roles of each track. We note $f_i^e$ the last frame of track $i$, and $f_j^b$ the first frame of frame $j$. In this case, the decay factor is computed as follows:

$$\alpha(i, j) = \exp\left(-\frac{|f_j^b - f_i^e|^2}{2\sigma_\alpha^2}\right)$$

(3.8)

3.3.6 Obtaining the Pairwise Affinity Matrix

The normalised pairwise distance $D(i, j)$, computed according to equation 3.4, is then turned into an affinity $A_D(i, j)$ as follows:

$$A_D(i, j) = \exp (-D(i, j)^2)$$

(3.9)

The final affinity values are computed by combining $A_D(i, j)$ and a constant value $\mu_D$, using the decay factor $\alpha(i, j)$:

$$A(i, j) = \alpha(i, j)A_D(i, j) + (1 - \alpha(i, j)) \mu_D$$

(3.10)

The value $\mu_D$ is computed as the mean of $A_D(i, j)$, computed on pairs of overlapping tracks only. It is the value towards which disjoint tracks that are far apart converge. The decay factor ensures that more credit is given to affinities of disjoint tracks that are not too far apart in time. The combination with $\mu_D$ ensures that when tracks are far apart in time their affinity converges towards a neutral value.

Our experiments show that computing the affinity for pairs of disjoint tracks helps grouping together trajectories on objects that undergo short-term full occlusions. A temporal gap of about ten frames is common. Under favourable circumstances, it is possible to cluster together
disjoint tracks across longer intervals. In particular, if the velocity of the tracked object remains the same before and after the occlusion, the motion distance between the tracks on this object would be low, favouring high pairwise affinities.

In the following section, we explain our track clustering algorithm based on spectral clustering of the matrix \( A \), followed by a linkage step and a subsequent user-assisted refinement step.

### 3.4 Split-and-Merge Feature Point Clustering

This section describes our clustering algorithm, based on analysis of the affinity matrix \( A \), which is computed as explained in the previous section. Firstly, a decision tree is created automatically via a split-and-merge clustering technique similar to Angeli and Davison \[3\]. Our split step is based on spectral clustering with Normalised Cuts \[142\]. The aim of this step is to oversegment the video by creating a large number of small but consistent clusters of tracks. The merge step then uses Min-max Cut linkage \[43\]. This step merges all clusters together, one by one, forming a binary decision tree in which each node corresponds to a split move when reading the tree from the root. Lastly, the user is asked to refine the segmentation by navigating the decision tree to generate the final result.

#### 3.4.1 Automatic Split Step using Normalised Cuts

The next step of our segmentation algorithm is to perform spectral clustering on the pairwise affinity matrix \( A \) computed for any pair of tracks, as explained in section 3.3. Figure 3.4 illustrates the principles of clustering via the affinity matrix. Groups of tracks having a high affinity are formed, while the affinity between pairs of tracks belonging to different groups must remain low.

Spectral clustering via Normalised Cuts \[37,52,142\] is used in many sparse video segmentation techniques \[22,103,179\]. In our experiments, we use the original formulation of Normalised Cuts by Shi and Malik \[142\]. In this framework, the affinity values \( A(i, j) \) are considered as weights connecting nodes in a graph. We note the set of all nodes \( C_0 \), where each node represents a feature track. Let us consider a bi-partition of the graph, i.e. two disjoint sets \( C_1 \) and \( C_2 \) such that \( C_1 \cup C_2 = C_0 \). A cut between these sets is defined as:

\[
cut(C_1, C_2) = \sum_{i \in C_1, j \in C_2} A(i, j)
\]

The total connection from nodes in a set \( C_1 \) to all the nodes in the graph is given by \( \cut(C_1, C_0) \).

The normalised cut between \( C_1 \) and \( C_2 \) is an unbiased measure of disassociation, which is defined as follows:

\[
\text{Ncut}(C_1, C_2) = \frac{\cut(C_1, C_2)}{\cut(C_1, C_0)} + \frac{\cut(C_1, C_2)}{\cut(C_2, C_0)}
\]

The objective of the algorithm presented in Shi and Malik \[142\] is to find the optimal partition that minimises this cost function. The authors show that minimising this cost allows to simulta-
Figure 3.4: Illustration of the clustering procedure on the affinity matrix. The leftmost illustration on the top row shows the affinity matrix as computed in section 3.3 for any pair of tracks. High affinity values are represented with bright values. To its right, the affinity matrix is displayed again, but after rearranging its rows and columns according to the output of our clustering method. The image on the bottom row displays the three clusters on one frame from the cars1 sequence in the Hopkins 155 dataset [158].

neously minimise the disassociation between the sets and maximise the association within each set. They recast the problem of minimising equation 3.12 as solving a generalised eigenvalue system. First of all, let us define a diagonal matrix $B$, such as the elements on the diagonal contain the total connection from one node to the others:

$$B(i, i) = \sum_{j \in C_0} A(i, j)$$  \hfill (3.13)

Shi and Malik [142] show that the sparse bi-partitioning problem can be turned, via relaxation, into the problem of solving following generalised eigenvalue system:

$$(B - A)x = \lambda Bx$$  \hfill (3.14)
3.4. Split-and-Merge Feature Point Clustering

The eigenvector $\mathbf{x}$ with the second smallest eigenvalue $\lambda$ is chosen as an indicator to bi-partition the graph. Thresholding the values of this vector with a splitting point gives a bi-partition of the graph. Several splitting points are tested, and the partition which minimises the normalised cut is finally selected.

3.4.2 Our Modified Stopping Criterion for Recursive Splitting

Then, each partition or cluster is further divided by applying the technique recursively on the sub-graphs made of the nodes in $C_1$ and $C_2$, until a stopping criterion is reached. A simple stopping criterion to end the subdivision of a cluster is presented by Shi and Malik [142]. This criterion is based on the stability of the chosen eigenvector that solves equation 3.14. However, it does not control the size and coherence of the generated clusters. Therefore, we modify this stopping criterion to have a direct control on the coherence of the generated clusters in terms of affinity values.

Let us define the binary matrix $H$ which locates pairs of tracks with low affinities that should not be grouped together during clustering:

$$H(i, j) = \begin{cases} 
1, & \text{if } A(i, j) \leq \theta_A \\
0, & \text{otherwise.}
\end{cases}$$  
(3.15)

The threshold $\theta_A$ gives an indication on what is a high affinity value. We define it as the mean value of values of $A(i, j)$ greater than the mean value of $A(i, j)$. Let us consider the cluster $C_1$. We then compute the proportion $h_1$ of bad associations in $C_1$. It is given by:

$$h_1 = \frac{1}{|C_1|^2 - |C_1|} \sum_{i \in C_1, j \in C_1} H(i, j)$$  
(3.16)

We subdivide the cluster $C_1$ only if $h_1$ is above a threshold that we fix to 0.2 in our experiments. Furthermore, we stop the algorithm if the number of tracks in a cluster is lower than 3.

3.4.3 Automatic Merge Step via Min-max Cut Linkage

At the end of the split step described above we obtain a large number of coherent clusters. Our technique then iteratively merges these clusters together two by two. To do so, we employ the Min-max Cut linkage algorithm [43], which is used in Angeli and Davison [3]. At each iteration the algorithm merges the clusters $C_p$ and $C_q$ that yield the greatest linkage score:

$$\text{link}(C_p, C_q) = \frac{\text{cut}(C_p, C_q)}{\text{cut}(C_p, C_p) \cdot \text{cut}(C_q, C_q)}$$  
(3.17)

By keeping track of the successive merge moves, we form a binary decision tree in which each node corresponds to a split step, when reading the tree from the root.

According to the literature [3,186], simple recursive merging algorithms based on linkage are prone to degenerate clusters in the presence of outliers in the data. Errors due to mis-associations
in the early stages of the merging procedure can have a dramatic impact on the merging in later stages. One way of avoiding early mis-associations is to first process data globally, using a split step, so that the input of the merge step is already consistent.

### 3.4.4 User-assisted Refinement Step

Defining what is a video object is not trivial, and each object could be represented by one or several clusters of tracks. This problem is known as setting the level of segmentation. Fully automated segmentation techniques \[3, 182\] need to infer the number of objects in the scene. However, as stated in chapter \[2\] this is an ill-posed problem because the desired segmentation level ultimately depends on the application. Instead of segmenting a scene at a fixed level \[3, 62\], a versatile technique would output a hierarchy of clusters so that the level of segmentation can be increased or diminished by the users, depending on their needs.

Using our clustering technique, it is possible that the solution generated at a given level in the decision tree is suboptimal for the purpose of a given application, i.e. the output can be oversegmented or undersegmented. Remember that our track comparison method described in section \[3.3\] is only able to model translational motion. Therefore, the automatic clustering method described above can generate spurious associations between tracks in sequences where there is heavy camera rotation around the optical axis. In these cases, the local motion of tracks at the centre of the scene differs drastically from the motion of tracks at the edges, but they may belong to the same video object.

Therefore, we provide users of our method with an interactive tool to navigate the decision tree. This allows them to correct segmentation errors and reach the output that suits their needs. Ideally, the technique should be implemented as a graphical user interface, to play back the video while refining the segmentation. One click on a marker at the centroid of a cluster at a given frame would allow to split it in two, during its lifespan and on both views. The procedure can be repeated until there is no undersegmented clusters remaining. Then a selection tool would allow the users to click on several clusters to be merged back together.

### 3.5 Final Remarks

In this chapter we have presented our work on stereo video segmentation. Our novel sparse segmentation technique computes an affinity measure between pairs of tracks that can be overlapping and also disjoint, in any view. We then use a split-and-merge clustering technique made of standard spectral clustering and linkage methods to generate a hierarchy of clusters automatically.

Within each cluster, temporal coherence is maintained throughout the video thanks to our novel affinity measure on disjoint tracks, that allows recovery from short total occlusions, whereas only partial occlusions could be dealt with in previous techniques. View consistency is main-
tained by a joint processing of features residing in both views, as well as the use of disparity as a cue when computing affinity.

However, we recognise that ill-posed problems must be solved when seeking for a fully automated technique. Particularly, the final number of clusters must be determined automatically. Instead of doing this, we maintain the users in the processing loop by letting them navigate a decision tree so as to refine the segmentation. Using this interactive tool, the users can explore possible split moves before merging back some clusters. As the clusters generated by our technique are coherent across views, the refinement is performed on both views jointly. In the next chapter we present experimental results that illustrate the versatility of our technique.
Experiments on Sparse Stereo-Video Segmentation

This chapter presents experimental results using our sparse stereo-video segmentation technique. We compare the basic performance of our technique to state-of-the-art methods on the Hopkins 155 dataset [158]. We also show the versatility of our approach on various videos taken from the Sigmedia stereo-video database presented in Corrigan et al. [31]. For illustration purposes, we choose challenging videos containing multiple moving objects occluding each other or exhibiting complex camera motion.

The study is divided into two parts. Section 4.1 presents our results without user-assisted refinement. In this section, we detail the impact of joint processing of both left and right views in a stereo video. We also show the advantages of a unified processing of both overlapping and disjoint tracks in our affinity-based clustering approach. Then, section 4.2 focuses on showing the benefits of including user interaction to correct the segmentation results.

4.1 Analysis of Our Automatic Segmentation Results

In this section, we analyse the results of our automatic feature track clustering algorithm, i.e. everything described in the previous chapter, excluding the user-assisted refinement module. More precisely, the core of our clustering technique as described in sections 3.4.1 to 3.4.3.

We first use the Hopkins 155 dataset [158], which is a widely used database of monoscopic videos, so as to assess the basic performances of our technique. Then, we detail our segmentation results on a few sequences extracted from the Sigmedia stereo-video database [31]. We use a Matlab implementation of our technique in all our tests. The number of clusters in the output
Experiments on Sparse Stereo-Video Segmentation

### Table 4.1: Segmentation results on the Hopkins 155 database in % of misclassification.

The mean and median classification errors within each group of test sequences are displayed in this table. In this test, the affinity is computed on overlapping tracks only, using the tracking data provided with the videos. We use the output of our automatic method, with the number of clusters set to the ground-truth value. Our method performs similarly to state-of-the-art techniques on the traffic and articulated sequences. However, it yields a high misclassification score on the chequerboard sequences as they mainly contain 3D motion outside of the image plane. Our results are compared with the results of two state-of-the-art 3D sparse segmentation methods [47, 130]. We select these methods for their good performance on this dataset, according to the results published in the associated website.

<table>
<thead>
<tr>
<th>sequences</th>
<th>error</th>
<th>2 motions</th>
<th>3 motions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ALC&lt;sub&gt;sp&lt;/sub&gt;</td>
<td>SSC</td>
</tr>
<tr>
<td>chequerboard</td>
<td>mean</td>
<td>1.49</td>
<td>1.12</td>
</tr>
<tr>
<td></td>
<td>median</td>
<td>0.27</td>
<td>0.00</td>
</tr>
<tr>
<td>traffic</td>
<td>mean</td>
<td>1.75</td>
<td>0.02</td>
</tr>
<tr>
<td></td>
<td>median</td>
<td>1.51</td>
<td>0.00</td>
</tr>
<tr>
<td>articulated</td>
<td>mean</td>
<td>10.7</td>
<td>0.62</td>
</tr>
<tr>
<td></td>
<td>median</td>
<td>0.95</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Table 4.1 shows the corresponding misclassification scores of our automatic clustering technique compared to two state-of-the-art sparse segmentation methods [47, 130] mentioned in chapter 2. For each method, the table shows the mean and median labelling errors, computed is set manually for each test.

### 4.1.1 Experiments on the Hopkins 155 Dataset

In order to test the basic performances of our segmentation technique we have experimented on the Hopkins 155 dataset [158]. This dataset is made of short monoscopic video clips. Each video is provided with manually corrected feature point tracks along with ground-truth labelling of the tracks. The videos in this dataset are classified into three groups, namely chequerboard, traffic and articulated. For this experiment, we use the automatic clustering method described in the previous chapter, without user refinement. We also deactivate the bi-layer segmentation step of our technique. The number of clusters in the output is set manually to the ground truth, by selecting the level of segmentation in the decision tree. The tracks provided do not contain missing data, so that they are all overlapping. We compare the results using our technique to the results of two state-of-the-art 3D sparse segmentation techniques [47, 130] which are provided in the associated website.

4.1. Analysis of Our Automatic Segmentation Results

Figure 4.1: Bi-layer segmentation on the gpo_track_2 sequence. The tracks marked in red indicate the static background cluster. The tracks marked in green belong to the dynamic foreground.

Within each group of sequences. On the traffic and articulated groups, our results are on par with the state-of-the-art scores. However, the two methods we compare to use a 3D sparse segmentation approach. This explains why they perform better than our 2D approach on the sequences from the chequerboard group, which contain large rotations outside of the image plane.

The chequerboard sequences from this dataset are too specialised to test multibody factorisation techniques and they do not correspond to natural scenes. An alternative dataset to assess the performances of video segmentation techniques on monoscopic videos has been presented by Brox and Malik [22]. However, we have not tested our method on these videos as we are mainly interested in stereoscopic-3D videos. Therefore, we test our technique on the Sigmedia stereo-video database [31]. These tests are presented in the rest of this chapter. The results of these tests can be viewed online at http://www.sigmedia.tv/Misc/ResultsThesisFelix.

4.1.2 Experiments on the Gpo_track_2 Sequence

For our tests on the gpo_track_2 sequence, we have extracted 100 frames from the original sequence, starting at frame 50. In this video, the camera rig tracks two walking men. Many pedestrians move in the background and foreground around them. A van crosses the field of view from right to left, and it fully occludes in turn passers-by and the two walking men. In this sequence, there is also a significant amount of camera jitter because the camera operator is walking. The purpose of our segmentation tests using this sequence is to illustrate the ability of our technique to recover from full occlusions.

Figure 4.1 shows the result of our bi-layer segmentation method described in section 3.2.
Most of the foreground tracks are kept for further processing. However, some tracks are misclassified as foreground, e.g. on the tree and on the pavement. This is partly due to errors in tracking, as some tracks show apparent motion when an object partially occludes their associated feature points. These tracks could be fused back to the background after the subsequent segmentation steps. Some tracks are also misclassified as background, because they exhibit a motion amplitude which is too small to be detected. They could be included in the foreground by tuning the parameters of our technique. No tracks are kept on the white van as the KLT tracker we use performs poorly on fast moving objects, generating very short tracks that are discarded during preprocessing of our segmentation method.

Figure 4.2 shows the result of our automatic clustering technique described in section 3.4. In this test, we set the number of clusters to 4, for illustration purposes. We show three variations of our method. Firstly, we show our results when each view is processed independently and affinity is computed on overlapping tracks only. Then, we show the segmentation output when processing both views jointly, but still computing affinities on overlapping tracks only. Finally, we show the output of our technique where tracks are processed jointly on both views and affinities are computed on both disjoint and overlapping tracks. The following three paragraphs present detailed observations on each variation of our technique.

In the first two rows from the top of figure 4.2, we show the output of our technique when processing each view independently, and computing the affinity for overlapping tracks only. The way affinities are computed in our modified algorithm is then similar to 2D sparse segmentation techniques such as Brox and Malik [22] and Fragkiadaki et al. [54]. The most obvious remark is that clusters that should be labelled identically have a different label on each view. The other observation worth mentioning is that the tracks on the two walking men are assigned to two different clusters before and after they have been occluded by the white van. They are assigned successively to the green and blue clusters on the left view, and to the yellow and blue clusters on the right view.

In the next two rows from the top of figure 4.2, we show the output of our technique, when processing both views jointly, but only computing the affinity of overlapping tracks. As can be observed, this yields a consistent labelling across views. Apart from the obvious colouring of each cluster, let us point out a more subtle difference to the viewer. Observe the tracks on the window corners above the van on frame 45. If processing each view separately, as can be seen on the first two rows of the figure, these tracks are assigned to the same green cluster as the tracks on the walking men on the left view. However, they are assigned to the red cluster on the right view, which does not contain any of the tracks on the two walking men. In comparison, in the output on the second row, it can be observed that a consistent labelling choice is made on both views for these tracks, due to our joint processing of both views.

In the last two rows of figure 4.2, we show the output of our technique when processing both views jointly, as well as computing affinities on both overlapping and disjoint tracks. Not only is the segmentation consistent across views, but it is also more consistent along time. In this
Figure 4.2: Results of automatic sparse segmentation on the gpo_track_2 sequence with 4 clusters. The first rows show the result if each view is processed independently and affinities are computed on overlapping tracks only. The next rows show the output when processing both views jointly. It is more consistent across views. The last rows show the output when affinities are also computed between disjoint tracks. It is more temporally consistent.
4.1.3 Experiments on the Kylemore Sequence

For our tests on the Kylemore sequence, we have extracted the first 100 frames from the original sequence. In this video, the camera rig is predominantly static. Many pedestrians walk in various directions, occluding each other. The purpose of our segmentation tests using this sequence is to illustrate the ability of our technique to maintain view consistency and to recover from multiple occlusions.

Figure 4.3 shows the result of our bi-layer segmentation method described in section 3.2. Overall, the tracks belonging to the background are identified correctly. Only moving pedestrians that are far away from the camera are not included in the foreground cluster, as their motion amplitude is too low. Figure 4.4 shows the result of our automatic clustering technique described in section 3.4. In this test, we set the number of clusters to 5, for illustration purposes. The figure shows the outputs of three variations of our method as explained previously.

On figure 4.4 observe the tracks on the woman with a hat walking in the foreground, from the left to the right of the scene. In the output displayed on the first two rows, segmentation is computed on each view separately. On the left view, the tracks are consistently assigned to the blue cluster. However, on the right view they are assigned to the yellow cluster in frame 5 and to the magenta cluster in frame 40. This shows a clear inconsistency of the segmentation across views. In the output displayed on the next two rows, segmentation is computed on both views jointly. The same tracks on the walking woman with a hat are consistently assigned to the cyan...
4.1. Analysis of Our Automatic Segmentation Results

Cluster on both views in every frame.

Observe now the blue and red clusters on the two rows in the middle of figure 4.4. In this case, segmentation is performed on both views jointly, but the affinity is set to zero for pairs of disjoint tracks, as in Fragkiadaki et al.  [54]. This causes a cut to be made along time after occlusions and loss of tracking occur for many tracks in the blue cluster. For instance, observe the tracks in the man pushing a pram, from the left to the right of the scene. These tracks are occluded by several other pedestrians. Tracks on this man first belong to the blue cluster and then to the red cluster, in frame 80, as they are disjoint in time.

Observe now the same tracks on the last two rows. The tracks are consistently assigned to the red cluster. In this experiment, affinity is also computed on disjoint tracks. Instead of a cut along time, the cut is made here between tracks on pedestrians moving from left to right in the red cluster and tracks on pedestrians moving in the other direction, which are assigned to the green cluster. This example illustrates that computing the affinity between overlapping and disjoint pairs of tracks increases the temporal consistency of the segmentation results.

4.1.4 Experiments on the Traffic Sequence

For our tests on the traffic sequence, we have extracted the first 100 frames from the original sequence. In this video, the camera rig is predominantly static. Many vehicles move in both directions along the road, temporarily occluding each other. The purpose of our segmentation tests using this sequence is to illustrate temporal and view consistency of the segmentation with our technique.

Figure 4.5 shows the result of our bi-layer segmentation method described in section 3.2. Overall, the tracks belonging to the background are identified correctly, but there is a significant amount of outliers. Some of these tracks located on the road show no motion at first, until the reflection of a moving vehicle on the wet ground is wrongly tracked. Incorrect tracks are also labelled as foreground on the wall on the left of the scene. These tracks exhibit apparent motion when a vehicle passes in front of them. Removing these outliers could possibly be done by tuning the parameters of our algorithm in favour of a more aggressive pruning.

Figure 4.6 shows the result of our automatic clustering technique described in section 3.4. In this test, we set the number of clusters to 6, for illustration purposes. The figure shows the outputs of three variations of our method as explained previously.

The first two rows illustrate the segmentation output when processing each view independently. In this case, view inconsistencies can be observed on the green and red clusters. In frames 75 and 95, in the left view, the tracks on the bus and white van coming towards the camera are assigned to the green cluster, and so are the tracks on the wall in the left of the scene. Whereas on the right view, the tracks on the wall are assigned to the green cluster and the tracks on the vehicles are assigned to the red cluster. In the next two rows on the figure, both views are processed jointly and these inconsistencies can no longer be observed. The tracks
Figure 4.4: Results of automatic sparse segmentation on the *kylemore* sequence with 5 clusters. The first rows show the result if each view is processed independently and affinities are computed on overlapping tracks only. The next rows show the output when processing both views jointly. It is more consistent across views. The last rows show the output when affinities are also computed between disjoint tracks. It is more temporally consistent.
4.1. Analysis of Our Automatic Segmentation Results

Figure 4.5: Bi-layer segmentation on the traffic sequence. The tracks marked in red indicate the static background cluster. The tracks marked in green belong to the dynamic foreground.

on the wall are assigned to the blue cluster and the tracks on the white van and bus coming towards the camera are assigned to the red cluster.

Observe now the tracks on the silver car moving from left to right. The silver car is temporarily occluded by the black car moving in the opposite direction between frames 75 and 95. On the two rows in the middle of figure 4.6, the tracks on this car are assigned to the cyan cluster in frame 75, before they are occluded, and to the yellow cluster afterwards, in frame 95. They are assigned to two different clusters in this case because the affinity between disjoint tracks is ignored. The last two rows show the output of our technique when tracks are processed jointly on both views and affinities are computed for both overlapping and disjoint tracks. In this case, the tracks on the silver car are assigned to the same magenta cluster throughout.

However, in the last rows of figure 4.6 notice that the tracks on the bus on the left of the scene are divided between two clusters in frame 30, depending on their apparent motion and location. There is one magenta cluster for the tracks with higher apparent motion amplitude moving from left to right and a green cluster for the tracks moving in the same direction but with a lower apparent motion amplitude. The same phenomenon can be observed on the tracks belonging to objects moving in the other direction. The cyan cluster regroups the tracks with a higher apparent motion amplitude, closer to the camera, and the red cluster regroups the tracks with a lower apparent motion amplitude, further away from the camera.

This example shows that it is difficult for our algorithm to identify each vehicle based on motion and position alone, but we show that general trends can be extracted from a scene. Depending on the application, the clusters can be further subdivided or merged together using our user-assisted approach. In the next section, we show examples to illustrate the benefits of this refinement step.
Figure 4.6: Results of automatic sparse segmentation on the traffic sequence with 6 clusters. The first rows show the result if each view is processed independently and affinities are computed on overlapping tracks only. The next rows show the output when processing both views jointly. It is more consistent across views. The last rows show the output when affinities are also computed between disjoint tracks. It is more temporally consistent.
4.2 Analysis of Our User-assisted Segmentation Results

This section shows the results of our user-assisted segmentation technique on stereo videos taken from the Sigmedia stereo-video database \cite{31}. In our tests, we have implemented the user-assisted refinement step described in section 3.4.4 as a command-line tool in Matlab. The user is first asked which clusters should be split in two. Then the computer asks which clusters should be merged back together. We refer to the number of split and merge steps needed to segment a scene as an indicator of the amount of user interaction necessary for a given segmentation task. However, remember that no further computation is made during this step, as every allowed split moves are stored in the decision tree.

4.2.1 Walk-through Example on the Rose_garden_tracking Sequence

For our tests on the Rose_garden_tracking video, we have extracted 51 frames from the original sequence. In this video, the camera tracks a woman walking towards a bench in the far background in front of her. There is a significant amount of camera rotation in this sequence, so that our bi-layer segmentation step fails to extract the background. Our goal is then to separate the tracks on the static background from the tracks on the walking woman.

The first two rows in figure 4.7 show the automatic segmentation output of our technique on this sequence, with the number of clusters set to 2. In this case, the rotation of the camera and the significant depth differences between the background tracks cause misclassifications. Due to camera rotation, the tracks in the lower-left corner of the scene have a significantly different motion than the tracks in the lower-right corner of the scene, as can be seen in frame 48. Moreover, the tracks on the far background plane have a significantly lower apparent motion compared to the tracks on the grass closer to the camera, as can be seen in frames 12 and 24. The selected automatic cut assigns the tracks on the woman and on parts of the background to the red cluster. The tracks on the grass in the right of the scene are assigned to the green cluster.

The automatic result is not what we want in this case. We then use our user-assisted refinement step to correct the segmentation output. The red cluster is selected to be split in two. The two rows in the centre of figure 4.7 show the output of segmentation after the split move. There are now two clusters corresponding to the background, coloured in red and green. There is also one cluster corresponding to the walking woman, coloured in blue. The last two rows of figure 4.7 show the final segmentation output after merging the two background clusters. We now have one cluster corresponding to the background, coloured in red and one cluster corresponding to the walking woman, coloured in blue.
Figure 4.7: Results of user-assisted sparse segmentation on the rose_garden_tracking sequence. The first rows show the result of automatic segmentation with 2 clusters. The tracks on the woman are grouped in the same cluster as the tracks on the background. The next rows show the output after one user-assisted split move. Two clusters correspond to the background and one cluster corresponds to the woman. The last rows show the output after one user-assisted merge move regrouping the two background clusters.
4.2. Analysis of Our User-assisted Segmentation Results

4.2.2 Walk-through Example on the Sphere_2 Sequence

For our tests on the sphere_2 video, we have extracted the first 100 frames from the original sequence. In this video, the camera is predominantly static, with some motion jitter. The sphere on the left of the scene is rotating around its axis. A man is walking from the left to the right of the scene. Our goal is to have three clusters, one for the sphere, one for the walking man, and one for the background.

The first two rows in figure 4.8 show the automatic segmentation output of our technique on this sequence, with the number of clusters set to 3. In this case, the motion of the tracks on the walking man and the rotating sphere are locally similar. This causes wrong associations to be made between the two objects in the green cluster, even though the difference in position and disparity allows for separation of most of the tracks.

We then use our user-assisted refinement step to correct the segmentation output. The two rows in the centre of figure 4.7 show the output of segmentation after two split moves: first on the green cluster, and then on the resulting cluster that still contains wrong associations. There are now three clusters corresponding to the sphere, coloured in blue, yellow and magenta. There is also one cluster corresponding to the walking man, coloured in green, and one red cluster corresponds to the background.

The last two rows of figure 4.7 show the final segmentation output after merging the three clusters corresponding to the sphere. We now have one cluster corresponding to the background, coloured in red, one cluster corresponding to the walking man, coloured in green, and one cluster corresponding to the sphere, coloured in blue.

4.2.3 Results on the Gpo_pan Sequence

For our tests on the gpo_pan video, we have extracted the first 100 frames from the original sequence. In this video, the camera is panning from left to right. Our bi-layer segmentation removes most of the tracks in the background, but a significant number of tracks remain. Various vehicles and pedestrians move in different directions. We illustrate a possible application of our user-assisted refinement step to subdivide and merge some clusters to obtain a more meaningful segmentation than the automatic output.

The first two rows of figure 4.9 show the result of automatic segmentation on this sequence, with the number of clusters set to 5. The cyan cluster regroups the tracks on the two pedestrians walking from the left to the right of the scene as well as the tracks on the bus moving in the same direction further away. There is also a cut for the far background tracks, dividing them between the green and black clusters. The tracks on the bus in the left of the scene are also divided between the white and red clusters.

In this case, user-assisted refinement is a more involved procedure than in the two previous examples. Some of the clusters obtained by automatic segmentation are oversegmented and others are undersegmented. In our test, we start the refinement procedure with one cluster
Figure 4.8: Results of user-assisted sparse segmentation on the sphere_2 sequence. The first rows show the result of automatic segmentation with 3 clusters. The tracks on the man are grouped in the same cluster as some tracks on the sphere. The next rows show the output after two user-assisted split moves. Three clusters correspond to the sphere, one cluster corresponds to the man, and an other corresponds to the background. The last rows show the output after one user-assisted merge move regrouping the three clusters corresponding to the sphere.
Figure 4.9: Segmentation of the \texttt{gpo} sequence. The top two rows show the result of our automatic split-and-merge clustering, with the number of clusters set to 5. The bottom two rows show one result of our user-assisted refinement producing 5 clusters. In this case 14 split steps and 5 merge steps are needed in total.

containing all the tracks and explore the decision tree from its root.

The last two rows in figure 4.9 show the final output after 14 split steps and 5 merge steps. User-assisted exploration of the decision tree allows to merge back oversegmented clusters and differentiate between the various moving objects in the scene. The black cluster corresponds to the background. The red cluster corresponds to the bus on the left of the scene. The magenta cluster corresponds to the bus on the right of the scene. The blue cluster corresponds to the walking man, visible in frames 15 and 65. And finally the yellow cluster corresponds to the woman crossing the road behind the bus.
4.2.4 Results on the close_up_1 Sequence

For our tests on the close_up_1 video, we have extracted the first 100 frames from the original sequence. The camera is static, recording two actors having a conversation. This example illustrates a possible application of our user-assisted refinement step to correct the automatic segmentation results. As too few tracks are observed on the background, our bi-layer segmentation technique does not work in this case. Our goal is then to form one cluster for the tracks in the background, and one cluster for each of the two actors.

The first two rows of figure 4.10 show the result of automatic segmentation on this sequence, with the number of clusters set to 3. It can be noticed that the yellow cluster, which mainly corresponds to the actor on the left of the scene, also includes the tracks on the background close to the actor. The other two clusters, coloured in white and cyan, regroup the tracks on
4.3. Discussion

It is important to ensure that when processing multiple-view videos, the amount of required user interaction is not proportional to the number of views. Accordingly, in the tests with our user-assisted technique described in section 4.2, the user never has to manually connect clusters across views. This shows that our technique exhibits strong stereo-consistency. When segmenting each view independently, in section 4.1 we show that clusters cannot simply be connected across views. View inconsistencies would have to be corrected beforehand.

Our method computes affinity between disjoint pairs of tracks. In our tests, disjoint pairs of tracks represent between 35% and 65% of the total number of possible pairs. This shows that a significant portion of the available data remains unexploited when computing affinity between overlapping tracks only. However, computing the full affinity matrix can become a computational bottleneck. A *Matlab* implementation using a for loop takes about three hours to compute the affinity matrix in our tests. After vectorisation, our implementation only takes about three minutes to accomplish the same task.

Fradet et al. [53] also use KLT tracks to test their sparse segmentation system. They mention a problem that we have observed in our data as well. Some trajectories really contain two trajectories, when a track jumps from one object to an occluding object. We mention this problem when describing the results of our bi-layer segmentation technique. It would be useful to detect such a phenomenon and split the trajectory into two or discard it as an outlier.

Our technique is based on position, disparity and motion of feature tracks only. Although image data is used during tracking, we do not use it for segmentation purposes. Exploiting colour and texture cues along tracks could help increase the separation of different objects having a similar motion. It could additionally help reduce the ambiguity when comparing disjoint tracks. Addition of colour to our method could be designed by following its usage in the track repair mechanism presented by Rubinstein and Liu [134].
Finally, remember that we use a 2D motion model when computing the motion distance between tracks. We show how user-assisted refinement can help overcome the limitations of this model, particularly when there is a significant amount of camera rotation. Alternatively, extensions to our technique could consider how previous works have included affine motion estimation in similar segmentation frameworks [11,22,53]. Motion models can be computed for each cluster after running our segmentation technique, and model-based refinements can then be performed to modify the result.

4.4 Final Remarks

Our results show that the novel user-assisted sparse stereo-video segmentation technique we describe in chapter [3] is able to efficiently segment stereo sequences while maintaining temporal coherence and accurate view consistency in the output. We have showed in section [4.1] that our joint processing of both left and right views allows a better stereo-consistency of the segmentation compared to independent processing of each view. We have also showed that computing affinity on both overlapping and disjoint tracks allows for the segmentation output to be more stable along time. Finally, we have illustrated in section [4.2] how the user-assisted refinement step gives our method a great versatility, as the output of the segmentation can be corrected or adapted to application needs.
Digital inpainting refers to the process of automatically recovering unknown information within a defined missing area in an image or a video. Inpainting has been used for many applications such as restoration of photographs [14] and film [18], rig removal for cinema post-production [88], and view generation for 3DTV displays [63].

Depending on the application, the picture region treated as missing data is called hole, gap, scratch, artefact, rig, occluding object or disoccluded region. It is generally marked by a binary mask. The mask can be obtained automatically for restoration applications [91]. It is often defined by the user manually, via rotoscoping. Rotoscoping consists in tracing the contours of the objects to be inpainted.

The main challenge for inpainting methods is to generate a reconstructed output without perceivable difference between the synthesised information and the known surrounding data. In this chapter, we detail how various techniques proceed to fill in missing picture data in images, videos, and stereoscopic media. We classify the methods along the chapter by increasing dimensionality of the processed data, from still images to stereoscopic videos.

The review is designed to help the reader understand the underlying motivations for the novel stereo-video inpainting technique we describe in chapter 6. We first present in section 5.1 the early evolutions of image inpainting. Then, in section 5.2 we describe how these techniques have been extended to fill in data in videos while enforcing temporal consistency of the output. Finally, in section 5.3, we focus on inpainting of stereoscopic media, which needs to consider additional view consistency constraints.
5.1 Image Inpainting Techniques

State-of-the-art image inpainting algorithms are able to automatically fill in large missing areas in images while maintaining continuity of both structure and texture of the surrounding data. They are often called image completion methods. As explained in section 5.1.1 these methods have emerged by combining the strengths of variational inpainting and texture synthesis. We review in section 5.1.2 the most successful image completion methods. They maintain spatial consistency of the output by using the exemplar-based framework to reconstruct both structure and texture information simultaneously.

5.1.1 The Genesis of Image Inpainting Algorithms

The term inpainting has been traditionally used in artwork restoration. It first appeared in the digital image processing community over a decade ago in a paper by Bertalmio et al. [14]. Although this paper is not the first in the domain, it is perhaps the most popular reference. Image inpainting has found many applications such as restoration of old photographs [14], filling-in of lost blocks during data transmission [15], removal of overlaid text or graphics [18], and removal of unwanted objects [34]. Two distinct categories of techniques have first emerged: variational inpainting [14] and constrained texture synthesis [45]. A few years later, image completion methods [15, 34, 94] have been developed to combine aspects of the former two approaches.

Variational image inpainting methods [14, 48] focus on the reconstruction of images with a structured background. The main idea is to propagate isophote lines joining points of equal intensity values, from the outskirts of the missing region towards its centre. These techniques are based on diffusion methods, which are inspired by principles of physics such as heat transfer or fluid dynamics. Their main drawback is that the diffusion process introduces visible blurring. They are therefore suitable for filling-in of small and narrow gaps, such as scratches or overlaid text. They are also effective in cartoon-like images, made of curves delimiting uniformly coloured areas, where loss of texture information is less noticeable.

Constrained texture synthesis techniques [45, 87] focus on statistical generation of picture data via sampling methods. The missing area is replaced with synthetic data that has similar statistical features than a sample surrounding known information. Many techniques use parametric models to generate the missing picture data. For instance, Kokaram [87] uses a 2D Autoregressive model. Efros and Leung [45] model the image with a Markov Random Field (MRF), so as to impose constraints on the conditional distribution of a pixel given its neighbours. The idea of MRF image modelling dates back from Geman and Geman [56]. The contribution of Efros and Leung [45] is to introduce a practical method to measure the conditional density function, using picture data in a non-parametric fashion. In general, these methods are efficient at replicating texture patterns in large missing areas within consistent image regions. However, they struggle at boundaries between regions of different textures. They can also fail at preserving higher-level structure of the image such as perspective geometry or
edge continuity of objects.

**Image completion** algorithms \cite{15,34,94} combine the strengths of variational inpainting and texture synthesis. They can deal with the problem of filling in large gaps while preserving both texture and structure of the surrounding image. One of the first image completion techniques was presented by Bertalmio et al. \cite{15}. In this technique, the input image is divided into its structure and texture components in a preprocessing step. Image decomposition is based on total variation minimisation for image denoising that keeps only structure information and oscillating functions for modelling textures. Then, variational inpainting and texture synthesis are applied on the respective component separately. The final output is obtained by combining both resulting images back together. The algorithm still yields a blurry output and is limited to small missing regions.

Many state-of-the-art image completion techniques use the exemplar-based framework presented in the seminal work by Criminisi et al. \cite{34}. This technique is based on greedy texture growing algorithms \cite{4,45} with added structural constraints. A global optimisation approach for exemplar-based inpainting has been proposed by Komodakis and Tziritas \cite{94} to solve quality issues in previous heuristic techniques. In section 5.1.2, we review exemplar-based methods in greater details, as a similar approach is at the core of the technique we present in chapter 6.

### 5.1.2 The Exemplar-based Framework for Image Inpainting

Exemplar-based image inpainting techniques \cite{34,94} fill in missing regions by copying information in patches of pixels from known surrounding areas. Image structure information is used to guide a texture synthesis process. Missing pixels within the hole are processed in order of priority so as to maintain structure of the surrounding image. The core of the technique then consists in searching for matching patches and copying pixels from known regions to reconstruct the missing texture information.

**Order of Filling:** Criminisi et al. \cite{34} remark that the order of filling of unknown pixels is of the utmost importance to preserve a coherent structure of the synthesised data. Their filling process is illustrated in figure 5.1. Previous approaches give equal priority to every point on the border or fill front by using concentric layer filling, also known as onion peel. This ordering can cause visible artifacts, such as broken object edges, as can be observed in figure 5.1. To mitigate this problem, higher priority is assigned to points having more known neighbours in Bornard et al. \cite{18}. For further improvements, in subsequent methods \cite{34,176}, propagation along linear structures is explicitly encouraged as well by assigning a higher priority to points lying on the continuation of isophotes. This is achieved by multiplying a confidence term counting the number of valid neighbours and a data term based on image information. The priority values are updated throughout the process. Luo et al. \cite{108} use addition of data and confidence terms to reinforce the role of the data term in the priority measure, compared to multiplication of the terms. The heuristic ordering discussed heretofore is employed in greedy image completion...
methods [18, 34, 176]. Komodakis and Tziritas [94] employ a priority mechanism in a Belief Propagation framework [49] to define the order of message transmissions. This enables their technique to infer the underlying structure of the image via Bayesian principles.

**Patch Comparison:** Although they can preserve image structure, the main drawback of variational inpainting techniques [14] is the use of colour diffusion mechanisms. This causes the reconstruction to be blurry, producing noticeable artifacts when processing large missing regions. Therefore, in exemplar-based approaches [34, 94], the reconstruction is made of selected candidate pixels that are copied from patches from known source areas in the image, similar to texture synthesis methods [4, 45]. The candidate which yields the minimum distance to the target missing site is selected for replacement. Distance is generally computed via the Sum of Squared Differences of known pixels in a patch around the missing site [34, 94], weighted by a Gaussian kernel [18] or by the amount of edges in a patch [144]. To avoid artificial repeating patterns in the reconstruction, an additional term is employed in Nie et al. [115] to penalise pixels that are used too frequently. Reliability of the reconstruction can be evaluated by the similarity between

---

Figure 5.1: Illustration extracted from Criminisi et al. [34]. On the first row from the top, the onion peel order of filling is used, which results in visible artifacts in the output. On the second row, the structure-guided filling from Criminisi et al. [34] is used, and the sign pole is reconstructed accurately.

---

*In Criminisi et al. [34], pixel colours are represented in the CIELab colour space [83] because it has been designed so that perceived colour differences correspond approximately to Euclidean distances.*
the selected patch and the patch around the reconstructed data. Shih et al. use this confidence value to update the order of filling. Choosing the scale of a patch is an open problem that has not yet been addressed thoroughly. Ideally it should be adapted locally depending on the size of the hole and characteristics of the local texture. Band-based inpainting allows for patches of arbitrary shapes to be completed simultaneously.

**Candidate Selection:** Pixels within patches of image data in the surrounding known region are considered for replacement of each missing site. Exhaustive search can be used on the whole image, albeit at a prohibitive computational cost, especially if a sequence of images is to be processed. So the known source image region, to be explored for candidates, can be chosen as a square area centred around the current pixel to be filled in. It has to be wide enough to capture the statistical variations of the texture being synthesised. An empirical decision is made to choose its size in Shih et al. Whereas each pixel is processed independently in Bornard et al., corresponding pixels within the selected candidate patch are replaced in Criminisi et al. Pixels at the centre of a patch only are replaced in Shih et al., so that an annulus of points around a patch is used to provide contextual information. This increases the impact of surrounding image structure when computing distances. A drawback of these techniques is that large patches should be used to increase the probability of finding a good match, which can yield important computation cost. For completion of natural images, the principle of coherence search is used in Bornard et al. to find good candidates among the neighbours of previously selected pixels. This avoids exhaustive search of the source area, and therefore speeds up the technique. Many techniques extend the process to obtain a seamless reconstruction by merging the source and selected target patches via Graph Cuts.

**Local versus Global Consistency:** Copying pixels from patches of known data in a greedy fashion has the advantage of being a simple and fast process. However, the reconstruction is only locally optimal. In practice, this means that boundaries of copied blocks can be noticeable, and there is no guarantee that lines crossing the hole are not broken. Such artifacts can be observed in figure. These issues can be mitigated by a non-local mean approach to mix the contribution of several best matching patches at the reconstructed site. An other approach uses local optimisation to maintain a list of candidate patches to delay the selection decision. The quality of the output can be further improved by using a global optimisation framework to take into account all inter-dependencies between patches to be copied. A qualitative comparison between the greedy approach of Criminisi et al. and the global optimisation method of Komodakis and Tziritas is illustrated in figure. A semi-automatic approach is used by Sun et al. In this technique, high-level cues on image structure from lines drawn by the user are incorporated in a global optimisation framework, to guide the reconstruction. Similar cues can be obtained automatically by a combination of image segmentation and extrapolation of curves delineating homogeneous regions across the hole.
5.2 Video Inpainting Methods

Digital cinema has dramatically facilitated video content manipulation. Convincing visual effects are commonly created during post-production, by removing gear such as wires and cranes supporting objects or actors. In other cases, an unwanted video object such as a passer-by, a car, or even a member of the film crew is visible in a shot and must be removed during post-production. The task used to be carried out manually by artists, who would trace a mask over the unwanted object along the shot and fill in the missing data. However, manual rotoscoping and inpainting of each frame is a very tedious and expensive process. Therefore, a great deal of research has been undertaken on its automation, to alleviate the work of artists and make the process more cost-efficient.

If a clean plate sequence showing the background layer is available, it is rather simple to paste in background pixels on the missing foreground region with techniques such as video matching [135]. However, arranging a clean plate image capture can be a tedious exercise outside a studio. This is why video inpainting tools have been developed, to automatically remove selected objects using the remaining information in the sequence. Apart from rig or object removal [88], video inpainting has found numerous applications such as restoration of old films [91] and completion of missing or corrupted frames during data transmission [145,171].

A major challenge for these techniques is to produce a reconstruction that is temporally coherent, i.e. free from perceivable artifacts when the video is played to a viewer. In section 5.2.1 we explain how patch-based image completion techniques described in section 5.1.2 can be extended directly to process videos, without explicit handling of motion. In the following sections, we focus on detailing how state-of-the-art techniques enforce temporal consistency within the
exemplar-based framework by explicitly handling the motion information.

5.2.1 From Image Inpainting to Video Inpainting

One of the pioneering works on video restoration, Kokaram et al. [91] uses interpolation based on 3D autoregressive models to reconstruct small and short-lived defects, such as scratches or blotches on old film material. A few years later, Bertalmio et al. [13] experiment on frame-by-frame variational inpainting for videos. The later allows to repair longer holes but can generate visible flickering artifacts, due to temporal mismatch of the reconstructed areas from one frame to the next. Bornard et al. [18] improve frame-by-frame reconstruction by including temporal information in their exemplar-based image completion method. The idea is to use global motion vectors to fetch candidate patches in the previous and next frames of a video. This helps preserving temporal consistency for short enough gaps.

Most state-of-the-art video inpainting techniques use at their core an image completion method, along the lines of Bornard et al. [18] or one of the other algorithms discussed in section 5.1.2. However, a great deal of research has been carried out to adapt the framework to video data. The approach taken by most video inpainting techniques [60, 96, 102, 111, 172] boils down to recovering the missing pixels by putting together matching spatiotemporal patches from different parts of a video. Compared to image inpainting, several additional problems have to be considered such as temporal coherence, computational efficiency and motion estimation in the presence of unknown data.

Patch-based Video Inpainting: Kumar et al. [96] present a spatiotemporal texture synthesis approach which is a direct extension of exemplar-based image inpainting [34] to process videos. The main contribution of this paper is to use the Fast Fourier Transform to speed up distance computations to compare spatiotemporal patches. Short-term picture information from neighbouring frames are used in the patch matching process. The technique is fast and works well for short gaps but longer-term information needs to be taken into account to fill in longer gaps. Another contribution of the authors is to modify the order of filling to process videos. Their order of filling assigns a higher priority to points with more valid neighbours in space and time, but favours propagation of structural information in space only. Indeed, it would not be possible to directly extend the priority measure to propagate structures along time. This is due to the fact that temporal edges are much more abrupt than spatial edges. For instance, a pixel site can contain data from the background on one frame and from the foreground on the next. Shih et al. [144] use a data term that gives higher priority to patches with a high proportion of edge points and high colour variation. This data term is advantageous to process videos as it is quicker to compute than the original formulation proposed by Criminisi et al. [34], which is based on isophotes.

Wexler et al. [171, 172] introduce one of the first exemplar-based video completion methods that processes a video in its entirety as a 3D volume. Their main idea is to search for the
most coherent assemblage of spatiotemporal patches or video cubes, taken from known parts of the video, to fill in the missing volume. The reconstructed information at each pixel is then computed iteratively, by a weighted combination of colours from the most similar patches. This can cause the output to look blurred, contrary to exemplar-based methods which select the best match for replacement. Apart from that, the technique is a straightforward extension of image completion to the space-time domain. It is computationally intensive because of the use of 3D exhaustive search to solve the problem within a global optimisation framework. A simple mechanism is proposed to reduce the computation by searching candidates only on pixels having a similar motion to the target. To this purpose, the ratio of temporal and spatial derivatives of the image intensity are used as a crude estimation of local motion.

Recent similar techniques [59, 60, 111] cite Wexler et al. [171, 172] as an inspiration, but design strategies to alleviate its computational load and improve the quality of the reconstruction. In these methods, optimisation is not performed on the image data directly, but on the shift map, which indicates the offsets between missing pixels and corresponding selected candidates in the known region. Granados et al. [60] incorporate temporal smoothness constraints to the energy minimisation framework. Newson et al. [111] show a significant speed-up of the technique with a fast approximate nearest neighbour search. They also show that selecting the best match for replacement instead of a combination of colours avoids blurring artifacts.

Li and Zheng [102] formalise a patch-based video processing framework to embed motion-related information in the relationship among video patches. Most patch matching techniques such as Kumar et al. [96] assign a probability of one to the best match, ignoring uncertainty from estimation errors. In Wexler et al. [171], estimation errors are masked via a blurring process. Li and Zheng [102] argue in favour of patch clustering instead, to keep multiple hypothesis in the process and construct a video model. Given the patch-based video model, variational Bayesian inference allows parallel reconstruction of missing data. This is computationally intensive but avoids error propagation problems in sequential video inpainting techniques. This implicit motion representation via patch clustering provides an elegant solution to video reconstruction where explicit estimation becomes difficult as part of the data is unknown. However, in practice, motion estimation and recovery is a crucial step in many video inpainting methods.

**Extending the Exemplar-based Framework with Temporal Smoothness:** The main challenge for video inpainting techniques, compared to image inpainting, is to enforce temporal smoothness of the output. Most state-of-the-art algorithms incorporate basic building blocks of the exemplar-based framework, similarly to the methods reviewed above, but also use motion information. In the following sections, we review in more detail three promising categories of methods, which exploit motion segmentation, motion repair and object tracking.
5.2.2 Video Inpainting Based on Motion Segmentation

Video inpainting methods are concerned with the reconstruction of large gaps or removal of objects that can span many frames. Searching for known source data to repair the missing region in the whole video is inefficient, because it typically contains dozens or hundreds of frames. Wexler et al. [171] use approximate instantaneous velocity to prune the search for the best patches from all available patches to patches with the most similar motion. In this section, we review techniques that constrain the search within motion segments. Motion segments can be defined as video regions of pixels exhibiting a similar motion. They can be obtained by video segmentation methods. We refer the reader to chapter 2 for a review on the topic.

Most patch-matching techniques reviewed heretofore implicitly assume that background pixels remain similar throughout the trajectory of a moving foreground object being repaired. Therefore, without any information constraining only foreground pixels to match, the completed patches may contain errors if the background is complex. In this section, we first study methods that perform bi-layer segmentation to differentiate moving foreground from static background to avoid artifacts at the boundary of objects [59, 71, 119, 140]. However, this is likely to fail in videos where both background and foreground are dynamic. So we then detail methods that perform multi-layer segmentation to constrain the reconstruction under more general assumptions [144, 185].

Approaches Based on Bi-Layer Segmentation: Several techniques [71, 119, 140] assume that the scene consists of a moving foreground over a predominantly static background. Bilayer segmentation is performed to separate both components. Each of them are then inpainted separately. Patwardhan et al. [119] use an automatic block-based segmentation. A block is assigned to the dynamic foreground if its motion is sufficiently different from the median of all blocks. A semi-automatic approach is used by Shen et al. [140]. In this method, the user has to supply a coarse boundary of the foreground on the first frame. The mask is then propagated via Mean Shift tracking [30]. Finally, to refine the segmentation, pixels from the foreground are detected with a statistical background subtraction method that models pixel colour distributions as mixtures of Gaussians [147]. The same approach is used by Jia et al. [71], in which the user has to draw the contour of foreground objects on several key frames. Multiple foreground objects can be handled by the semi-automatic method.

In Patwardhan et al. [119], all foreground frames are motion-compensated, using 2D translational global motion, and aggregated to form a mosaic. This mosaic is used to find candidate source patches. During reconstruction, segmentation is used to copy source pixels from the foreground only. In Shen et al. [140], objects in the foreground are also rectified to compensate for perspective projection. In Jia et al. [71], homographies are used to align and merge a damaged moving object with samples of the same object along time. The static background is completed by aligning the frames, to copy directly available pixels from temporal neighbours whenever
possible. Remaining pixels are never revealed in the sequence, so they can be filled by image completion and warped back to each frame. The repaired foreground layer is then composited over the reconstructed static background. This can cause the result to look unrealistic, e.g. if the shadow of a walking person is not included in the segmentation.

In Granados et al. [59], a static background is reconstructed behind selected objects after frame alignment. This technique allows for complex, unconstrained camera motions. However, remaining dynamic objects in the scene must be segmented so that they are left out of the frame alignment and background reconstruction processes. A semi-automatic dense video segmentation method [5] is used to obtain the masks delimiting the dynamic foreground. Image data is then reconstructed automatically from aligned candidate frames where the data in missing regions is visible. First of all, a set of homographies is estimated for each pair of frames. Then, warped pixels are selected for replacement of missing data within a global optimisation framework, designed to minimise colour discrepancy while maintaining epipolar geometry constraints. Possible illumination changes in outdoor scenes are compensated via Poisson image editing [121].

Approaches Based on Multi-Layer Segmentation: Zhang et al. [185] use a more general assumption, considering that the video is made up of several planar motion layers. This technique is illustrated in figure 5.3. Level set and Graph Cuts is used to estimate the motion layers [178]. Occluded pixels and layer ordering are explicitly estimated by occlusion reasoning, but the ordering must remain the same throughout the video and no cross-occlusion between layers are tolerated. Then, missing areas in each layer are located and completed with the available data on other frames by affine motion compensation to a reference frame. Image completion is used on the remaining pixels, rectifying patches before similarity computation to handle projective deformations from frame to frame. Finally, the output is synthesised by warping and compositing the restored layers. The technique cannot handle dynamic textures, and works only for rigid moving objects.

Shih et al. [144] constrain the reconstruction within motion segments to enforce temporal consistency. A modified Four-Step Search block-based motion estimation [124] is employed to pre-compute the motion of patches. Blocks with similar motions are merged to form motion layers. During completion, reconstructed areas are warped from one frame to the next and these pixels are used as source of candidates to maintain temporal continuity in the overlapping missing region. However, segmentation is a difficult problem, because motions can be complex due to fast-moving objects, motion blur and non-rigid bodies. Therefore, confusion can arise at the junction of motion segments. Moreover, the technique is sensitive to errors in both segmentation and motion parameters estimation. Unlike Zhang et al. [185], occlusions have to be dealt with manually by setting the ordering of motion layers in Shih et al. [144].
5.2. Video Inpainting Methods

In this example, two motion layers are estimated: one corresponds to the car and the other to the background. A third mask is displayed, corresponding to the board in the foreground, which is the selected object to be removed by inpainting. The two motion layers are restored independently and composited together in the output.

5.2.3 Video Inpainting Based on Motion Repair

Natural videos contain complex camera motions, multiple objects that can occlude each other, fast objects that cause motion blur and illumination changes. These phenomena are challenging for motion estimation techniques, which are at the core of many video inpainting methods. Illumination consistency is taken into account by Jia et al. [71] by intensity normalisation via tensor voting [73]. To simplify the problem, limitations are often imposed on the scenes that can be processed, the most common of which is that motion should be approximately parallel to the plane of projection [71, 74, 119, 185]. The presence of unknown data adds even more challenges to the motion estimation problem. In this section, we review techniques that infer the unknown motion inside the missing area given available information in its surroundings.
Motion inpainting has been developed in parallel to image inpainting to correct defects in Optical Flow estimation due to multiple motions, aperture problems or occlusions. Berkels et al. [12] propose a total variation anisotropic diffusion method that explicitly takes into account discontinuities at the underlying image edges. However, it supposes the image data is known. In this section we are interested in techniques that use motion restoration in the missing region to achieve temporal consistency while reconstructing unknown picture information. Once the motion vectors are recovered, unknown pixels can be filled in automatically by pulling in known image data when it is revealed. We first review techniques that pose the problem in a Bayesian framework [88,97] before detailing exemplar-based motion completion methods [23,105,145]

**Approaches based on Bayesian Motion Interpolation:** First of all, assume that in all the techniques described hereafter the motion field is computed in known parts of the video, via a method such as the hierarchical Lucas-Kanade Optical Flow estimator [107]. Lauze et al. [97] then use diffusion methods to reconstruct Optical Flow and picture data jointly. The motion field is inpainted with a multi-resolution variational approach, solved by Maximum A Posteriori estimation. The technique assumes spatial smoothness of the Optical Flow but cannot model motion discontinuities at the boundaries of objects.

The rig removal technique presented by Kokaram et al. [88] uses a Bayesian framework to reconstruct the underlying backward and forward motion fields in the missing area while handling occlusions. This paper is little known within the computer vision community. The technique deals with occlusions in a more consistent way than segmentation-based methods, by estimating an occlusion map jointly along with the motion vectors. The binary occlusion map allows modelling of temporal discontinuities at the boundaries of moving objects. Away from occlusion areas, assuming there is little acceleration between two frames, temporal smoothness is encouraged. This is done by penalising candidate motion vectors that do not match well with their counterparts in adjacent frames. Spatial smoothness is also encouraged on the interpolated motion and occlusion fields. Indeed, objects tend to be locally well connected, and the direction of the most likely motion vector tends to be aligned with the majority of its spatial neighbours. Local conditional maximisation [16] allows a solution to be generated at each pixel conditioned on the state of its spatial neighbourhood to simplify the problem. To reduce the set of possible candidates and speed up the convergence, three sources are used for initialisation: the output of a weighted motion estimator, motion vectors from adjacent frames, and spatially interpolated vectors. After the motion field is reconstructed, video completion is performed using weighted motion compensated adjacent frames. Blurring occurs due to repeated motion compensation of the progressively reconstructed rig region. Fast movements in the background and lighting changes can also cause problems to this method.

**Approaches based on Exemplar-based Motion Completion:** Motion Field Transfer [145] is an application of the exemplar-based framework to reconstruct local motion vectors. Results
5.2. Video Inpainting Methods

Figure 5.4: Illustration extracted from Shiratori et al. [145]. The first row from the top shows five frames from the original video. The second row shows the masked region to be inpainted, corresponding to the child in the foreground. The third row displays resulting frames from the video after inpainting with Motion Field Transfer [145]. These results exhibit blurring artifacts which are inherent to most motion-based inpainting techniques.

of this technique are displayed in figure 5.4. The technique consists in assembling spatiotemporal patches of known motion vectors to reconstruct the Optical Flow field in the missing area. Higher priority is assigned to sites which have more valid neighbours. A specialised distance measure has to be employed to compare motion patches. The average angular difference [9], over valid motion vectors in a patch, allows to account for differences in both direction and magnitude of vectors in the candidate selection process. Multi-scale search using a Gaussian pyramid is used to speed up the search for candidates. Bugeau et al. [23] use the exemplar-based technique of Criminisi et al. [34] on motion patches to reconstruct backward and forward Optical Flow fields. To correct possible erroneous flow vectors at the boundary of the hole, the mask indicating the location of missing points should be enlarged before reconstruction.

Once the motion field is repaired, the task is to reconstruct missing picture data in the video. Shiratori et al. [145] use motion vectors as links representing pixel correspondences among frames so as to propagate colour information from the known regions. The reconstructed colour values are computed as a weighted average of temporal neighbours. More precisely, a least-squares estimate is used, with weights depending on the reliability of the closest point. Reliability is computed as the similarity between the target motion patch and the selected source patch during vector field reconstruction. Motion vectors point to fractional locations of pixels so the weights are a mix of reliabilities depending on their proportion of overlap. The algorithm does not work well for completion of videos with large motions and is more sensitive to noise than directly using colour sampling, because motion is a first order derivative of the video data. Resulting
blurring artifacts with this technique can be observed in figure 5.4.

To avoid the blurring artifacts in the output, due to the weighted average scheme for colour propagation, Liu et al. [105] combine motion field transfer [145] with an exemplar-based video reconstruction. A global optimisation method is used to consider both spatial and temporal coherence relationships. The spatial term in the energy function ensures that the overlapping regions between adjacent patches have consistent texture and structure information. To emphasise structure preservation, a term is added to the usual computation of the Sum of Squared Differences, to compare the strongest horizontal and vertical gradients within a patch. The temporal term constrains two corresponding patches in two sequential frames to have consistent colours. For each patch, the four nearest points to its temporal correspondent in the next frame are considered in the energy function. Distance is then computed on the overlapping area of motion-compensated patches. A coarse-to-fine Belief Propagation solution is employed to deal with the large amount of label candidates.

5.2.4 Video Inpainting Based on Feature Point Tracking

Compared to images, the greater amount of data in videos introduces a lot of redundancy, which should be exploited to make the reconstruction process tractable. In exemplar-based approaches, finding a similar source patch along a video is facilitated if the missing data reoccurs at an other time, e.g. for objects with cyclic motions [71]. In the general case, motion vectors are nevertheless natural candidates to guide the reconstruction process. Feature point tracking is used in some exemplar-based techniques to restrain the search for the best patch. Moreover, point trajectories provide longer-term associations than motion vectors that can be exploited to enforce constraints on the reconstructed pixels.

Automatic object tracking is used in some techniques [71, 140, 144] to relieve the user from manually selecting the missing area over the whole video. Shih et al. [144] use average motion vectors within a motion segment to propagate a bounding box around the object to be removed, followed by inpainting-based refinement. In other methods [71,140], the user is asked to supply a mask delimiting the unwanted object on key frames and Mean Shift tracking [30] is used to propagate the mask on the whole sequence. However, tracking can also be used during video completion. For instance, a user-assisted object tracking mechanism is used in Granados et al. [60] to reduce the size of the search area for candidates.

In this section, we first review techniques that use feature tracking to find good candidates for replacement to extend the exemplar-based image completion framework [74, 119, 140]. Then, we detail how temporal smoothness can be explicitly enforced in the reconstructed region along trajectories [23,151].

**Candidate Search via Feature Tracking:** Jia et al. [74] base their technique on non-parametric sampling [45]. Moving objects over a static background without camera motion are tracked using Mean Shift [30], to restrain the search for source candidates. Patches on
the boundary of the missing area that have more trackable pixels in their neighbourhood are assigned a higher priority, as a similar source patch is more likely to be found. Pixels to reconstruct a missing patch fragment are searched in all possible source frames found by tracking. To encourage temporal consistency, neighbouring source fragments are preferentially chosen to complete temporal neighbours in the target area, similar to coherence search for spatial consistency. If tracking cannot provide candidates that are good enough, completion is postponed, as a more similar patch may become available later on. Un-trackable fragments that cannot be found elsewhere in the video are filled in by image completion. Therefore, temporal consistency may be violated when tracking is lost.

Patwardhan et al. base their technique on the exemplar-based framework of Criminisi et al. For a missing region with limited temporal extent, higher priority is given to points in the outer frames, because they have more un-damaged information in their close temporal vicinity, which is easier to fetch. After a bi-layer segmentation step described in section 5.2.2, global 2D translational camera motion is employed to align all frames and build foreground, background and Optical Flow mosaics. Overlapping components are averaged in each mosaic, and background motion is first subtracted before accumulating motion vectors. This scheme allows for a quick search for candidate frames using a patch matching strategy directly on the mosaic with both colour and motion features. Subsequent search for matching candidates is limited to these frames instead of the whole sequence.

Shen et al. employ a similar approach to Patwardhan et al., extended to deal with perspective motions. Their bi-layer motion segmentation is described in section 5.2.2. Camera motions such as panning, tilting and zooming are considered by warping each frame to a reference view with homographies. Head and feet of a moving person to be removed from the shot are tracked, to compute affine rectification parameters for the foreground component. The search space is reduced from 3D to 2D by slicing the video volume along motion manifolds of moving objects. Video completion then proceeds based on Sun et al., with additional explicit constraints to enforce smoothness along point trajectories. Then, the rectified layers are warped back to the original video, which allows periodic changes in illumination and minor periodic motion to occur in the background.

**Temporal Smoothness of the Reconstruction along Trajectories:** Video completion in Shen et al. is based on a global image completion technique, with a modified energy function to enforce temporal consistency along point trajectories. Curves parallel to the trajectory of an object, estimated via global motion, are used as propagation paths. Curve fitting is applied to foreground points to obtain trajectories automatically. Patches are then sampled along a narrow area close to 2D propagation paths, which greatly reduces the search for candidates compared to full 3D search. Image completion is used in case missing pixels remain.

Bugeau et al. propose to explicitly smooth out the reconstruction along trajectories, as
From Image Inpainting to Stereo-Video Inpainting

Figure 5.5: Illustration extracted from Bugeau et al. [23]. Three frames from the original video are displayed on the first column from the left. The corresponding masks delineating the region to be inpainted (the wake-boarder) can be seen on the second column. Inpainting results with Bugeau et al. [23] are displayed on the third column. The fourth column shows results obtained with Wexler et al. [171], which exhibit significant blurring artifacts. The motion-guided filtering process from Bugeau et al. [23] allows a better preservation of the water texture.

A postprocessing step of frame-by-frame image completion. Results of this technique can be observed in figure 5.5. Computation of the trajectories of missing points relies on the estimation and repair of the forward and backward dense optical flow fields as described in section 5.2.3. Bilinear interpolation is used to compute motion vectors at non-integer locations during tracking. Temporal filtering of the reconstructed frames is then performed via Kalman smoothing [76] along the estimated trajectories of recovered points. It is assumed that a near-perfect reconstruction of the first and last frames is available, otherwise errors would propagate throughout the sequence. Therefore, manual correction of those frames is often needed in practice. Low-pass filtering can yield a loss of texture detail in reconstructed dynamic backgrounds. However, as illustrated in figure 5.5 the use of motion-guided filtering allows for a better preservation of dynamic textures than methods such as the patch-based approach of Wexler et al. [171].
5.3 Inpainting for Stereoscopic Images and Videos

The end of the year 2009 has seen a breakthrough for stereoscopic-3D cinema, notably with the release of James Cameron’s *Avatar*. The year 2013 has seen the technology behind 3D cinema reach a high point with the critical acclaim of Alfonso Cuaron’s *Gravity*. In the meantime, sales of 3DTV sets have steadily increased as contents such as sports and musical events in stereoscopic 3D have gained popularity. As the production of stereoscopic content has been rapidly increasing over the past few years, so has been the need for new tools to manipulate that content, such as stereo-video inpainting.

In addition to spatiotemporal coherence constraints described heretofore, inpainting for stereoscopic data has to take into account *view consistency*. In section 5.3.1, we explain the motivations for using and extending image and video inpainting techniques described in sections 5.1 and 5.2 to process stereoscopic content. The following sections detail how state-of-the-art stereoscopic inpainting techniques adapt these methods to exploit inter-view disparity and depth information.

5.3.1 From Monoscopic Inpainting to Stereoscopic Inpainting

Stereoscopic videos are made of two streams, one for each eye. In general, compared to monoscopic videos, it would require at least twice as much work to manually touch them up. Applying image inpainting to both images of a stereoscopic pair independently can produce discrepancies in texture, structure or colour in the reconstructed area. Asymmetry between the two images of a stereo pair causes binocular rivalry [146]. This phenomenon hampers fusion of the images by the brain, and can lead to eyestrain and visual fatigue for the viewer. Differences in geometry between views can also result in unnatural depth perception. Therefore, preserving a comfortable and sensible sensation of depth on the inpainted output calls for additional attention: it is important to ensure consistency of the reconstructed data across views. The purpose of inpainting for stereoscopic content is then to fill in the missing area in both left and right views so that the synthesised data looks like a projection from a real 3D scene.

There are two different kinds of applications for inpainting of stereoscopic media, for which two categories of techniques have been developed. The first category extends monoscopic inpainting to process stereoscopic images and videos in the general case of object removal, where the object can be selected by the user anywhere in both views. The second category is more specialised. It consists in applying inpainting to fill in disoccluded background areas at the border of foreground objects after synthesis of a new view. It has received a lot of attention lately mainly due to the growth of stereoscopic television and 2D-to-3D conversion for cinema. We introduce both kinds of techniques in this section as they are equally relevant to explain the developments that lead to stereo-video inpainting.
Generic inpainting for any user-provided mask in stereoscopic content: In the general case, unknown information can reside in both views. The user can be asked to select an object to be removed on one view only. And the mask can then be propagated to the other view via disparity mapping. User interaction may be necessary to correct artifacts due to errors in the estimated disparity. Selecting an object to be removed is also possible on the disparity or depth map directly. However, one must be cautious when doing so, as an object may appear dilated in the disparity map due to the edge-fattening problem of many stereo algorithms.

The aim of stereo inpainting is to produce a reconstructed region that preserves a natural and comfortable visual perception of depth. Stereo inpainting can exploit characteristics of stereo image pairs to its advantage. Firstly, the region to be filled in one view may be partially visible in the other due to the redundant nature of stereoscopic data. Secondly, the completion process can be guided by depth information obtained from stereo matching so as to differentiate structural elements in the scene. Note as well that consistency on both images and depth maps can provide a quality measure of the result.

Both left and right images of a stereo pair are mostly similar. Uncovered information behind unwanted foreground objects can be directly recovered in binocular half-occlusion areas that are visible in one view only. Consider the case of removal of a static artifact produced by a defect on a camera in a multi-view setup. Due to parallax, objects situated at different depth levels have an apparent motion between views. Missing data can then be recovered by mapping displacements along views. Similarly, in a stereo video the missing data in disoccluded regions are likely to be present in other frames in the shot when foreground objects are moving.

In section we have reviewed segmentation-based techniques for video inpainting. We mention in chapter how mutual occlusions and depth ordering of objects are difficult problems to solve in monoscopic content for segmentation techniques. Chang and Hsu use depth maps and multi-layer segmentation to determine the sequence of objects to fill in, so as to repair a missing area overlapped by multiple objects. The depth of each object in the scene can be directly estimated with the additional disparity information in stereo frames. Depth can also be used to segment the scene between foreground and background so as to extract structural information and restrain the search area for appropriate candidates within one depth layer.

The specific problem of disocclusions in new view synthesis: Motivated by the development of 2D-to-3D conversion in the entertainment film industry as well as the need for view interpolation for (auto-)stereoscopic displays, view synthesis techniques have become a very active field of research. Given a colour image and its corresponding depth map, new views for binocular stereopsis can be rendered with Depth-Image-Based Rendering (DIBR) techniques. Occluded background regions behind foreground objects may be revealed in the rendered images due to parallax. These disoccluded regions can then be filled in along the view synthesis workflow by means of inpainting techniques.

Fist of all, let us consider the view synthesis problem. This problem is illustrated in fig-
5.3. Inpainting for Stereoscopic Images and Videos

Reference view and associated depth map
Synthesised left and right views, and associated depth maps, with missing data in disoccluded areas
Left view (and zoom on detail) after inpainting disoccluded regions with:
(a) Criminisi et al.  
(b) Daribo and Saito

Figure 5.6: Illustration extracted from Daribo and Saito [39]. This example illustrates the problem of new view synthesis and disocclusion filling. Disoccluded areas appear near depth boundaries on the synthesised left and right views. These regions can be filled in with inpainting techniques. The output obtained with the image-based technique from Criminisi et al. [34] exhibits artifacts at the interface between the textured foreground and the uniform background. Depth-guided inpainting with Daribo and Saito [39] shows a better preservation of the depth boundaries.

Given a reference view and an associated depth map, a new view and depth map have to be generated. In the general case, a stereo-pair of frames has to be generated from a new viewpoint. Both images in the synthetic stereo pair can then contain disoccluded data near depth boundaries. This results from the displacement of foreground objects revealing some background areas that are occluded in the reference view. As the baseline between a synthetic frame and the reference frame increases, the occlusion areas widen. Depth preprocessing using filtering or morphological operations can avoid small holes after synthesis by reducing depth discontinuities. However, for larger disocclusions, preprocessing of the depth map would introduce visible geometrical distortions. This is what has motivated the development of inpainting techniques to solve the problem of occlusion generation, i.e. recovery of unknown or disoccluded background areas [156]. Figure 5.6 shows the result of two inpainting methods on the disoccluded areas.

Three-dimensional television, or 3DTV, has received a lot of attention in recent research. There is a need for creating images seen from new viewpoints, to adapt the depth content for best viewing, as well as for multi-view auto-stereoscopic displays. This has motivated the development of 3D video formats incorporating additional depth information alongside picture data, and novel view synthesis techniques. For instance, DIBR has been recognised as a promising tool to synthesise new virtual views from video-plus-depth data on-the-fly [146]. An alternative solution to view synthesis on-the-fly would be to transmit all the necessary views to the display. However, this can be infeasible for bandwidth limitation reasons. Layer depth images [138] allows to store
extra data for occluded pixels behind foreground objects. This additional occlusion data can be
generated before transmission, with inpainting techniques, and used for direct filling of unknown
disoccluded data after view synthesis. However, this means increasing the overhead complexity
of the system. Sometimes it is therefore preferable to transmit data in the form of video-plus-
deepth, and use inpainting techniques subsequently to recover the disoccluded data after novel
view synthesis. This allows for a higher flexibility in the output as the depth impression can be
adjusted and customised after transmission.

**Extending the Exemplar-based Framework with View Consistency:** Most state-of-
the-art inpainting techniques employed to modify stereoscopic content or in conjunction with
view synthesis methods extend the exemplar-based framework described in section 5.1.2
Modifications to the original framework are necessary to take into account the depth geometry of
the scene. In this section, we first review a class of techniques that enforce view consistency via
joint or distinct reconstruction of the disparity map alongside picture data on still images. Then
we detail specific adaptations of inpainting for occlusion generation in videos, and specifically
how spatiotemporally coherent occlusion data can be generated.

### 5.3.2 Image Inpainting Based on Disparity Completion

In inpainting techniques, to maintain coherence across views, disparity vectors can be exploited
to find correspondences between pixels, as motion vectors are used to maintain coherence along
frames in a video. Accurate recovery of the inter-view disparity map in the missing area is at the
core of the inpainting algorithms we review in this section. These techniques are concerned with
completion of missing data in a stereo-pair of images. We first review methods that inpaint the
colour and depth information of a stereo pair jointly [66,168], assuming that initial disparities
are known everywhere. Then we review techniques that improve the convergence of the former
algorithms by applying a two-step reconstruction to repair disparity maps before filling in the
missing colour values [1],[67],[68].

**Joint Reconstruction of Colour and Depth:** Several techniques [66,168] need full disparity
maps corresponding to the original stereo pair before any unwanted object can be selected to
be inpainted. Disparity values cannot be estimated directly in binocular half-occlusion areas in
which pixels are visible in one view only and occluded in the other. In case a depth sensor is used,
physical limitations can cause more unknown areas to appear. Disparity maps can be estimated
as well as binocular half-occlusions using standard techniques [150]. Disparity completion can
then be performed on each view separately. In Wang et al. [168], the image is first segmented
into regions of homogeneous colour via Mean Shift [29]. It is assumed that the disparity in these
regions varies smoothly and that their corresponding 3D surfaces can be modelled by a plane.
Un-occluded segments are assigned a disparity plane via RANSAC-based fitting [51]. Occluded
points are assigned nearby disparity values under the weak consistency constraint [57]. This
constraint allows inconsistent visibility relationships to be avoided, but it can be violated in real scenes, for instance due to the double nail illusion [95]. He et al. [66] note that unknown disparity values are located at depth discontinuities when the foreground partly occludes the background. Background disparity values can thus be simply propagated via diffusion to the unknown areas to obtain full disparity maps. Diffusion gives acceptable results as a depth map typically consists of homogeneous regions separated by depth boundaries.

Wang et al. [168] reconstruct both colour images and disparity maps jointly in an iterative manner. First, each view is filled with what is readily available on the other view. This mutual completion step consists in warping each view to the other using known disparity values, under the weak consistency constraint [57]. In cases where more than one point is warped to the same destination, the point with the largest disparity value is kept, as it is the closest to the viewpoint. Remaining missing pixels after mutual completion are invisible in both images. Their colour and disparity information are then synthesised simultaneously. This is done with an adapted version of the exemplar-based image inpainting technique from Criminisi et al. [34], in which depth information guides the candidate selection process. The distance between patches is extended by adding two terms. Firstly, the sum of disparity differences within a patch is used to favour candidates within the same disparity plane as the target. The second term penalises candidate points that violate the view distance constraint. This constraint penalises candidates having disparities larger than the known values around the target missing pixel, as uncovered objects can only be further away. After one pass of inpainting on both views independently, a cross-validation consistency check is used to detect incorrectly filled areas. Assuming the surfaces in the scene are approximately Lambertian\(^2\), unreliable corresponding pixels across views can be detected if their colours are too different. These areas are refined in subsequent passes. The main weak point of this technique is that the quality of the reconstruction depends on the quality of the depth maps. Therefore the quality of the output degrades when stereo matching fails, i.e. in texture-less regions and when specular reflections occur. Moreover, convergence of the process is not guaranteed, which can leave artifacts in the final output.

He et al. [66] present a depth-guided exemplar-based inpainting algorithm to process simultaneously a single colour image and its associated depth map. The distance measure is similar to Wang et al. [168], except the second additional term. It is replaced by the Euclidean distance between the coordinates of target and source patches, to encourage copying from nearby regions. However, the view distance constraint is enforced by restricting the candidate set to points that have a smaller disparity than the original value. The search is even further constrained to candidates that have a depth value in the range of the values at known neighbours around the target. An iterative procedure which considers interactions between neighbouring patches is used to ensure the coherence of the reconstruction. Similar to Wang et al. [168], the reconstructed disparity is directly copied from the source region, so gradual changes along inclined planes are difficult to preserve in the reconstructed data.

\(^2\)The apparent brightness of the surface to an observer is the same, regardless of the observer’s angle of view.
Distinct Reconstruction of Depth and then Colour: Hervieu et al. [67] ensure convergence and consistency of the reconstruction by means of a two-step process. Results of this process are illustrated in figure 5.7. First the disparity maps are inpainted using an energy minimisation process. Assuming smooth variations of depth, a diffusion mechanism propagates known disparity values while enforcing the visibility constraint. This constraint states that the disparity of an occluding pixel should be greater than the value of an occluded pixel. Then, both colour images of a stereo pair are inpainted simultaneously, so that corresponding pixels have the same intensity value. Firstly a mutual completion step similar to Wang et al. [168] is employed. Then, remaining pixels are filled in with a modified version of Criminisi et al. [34]. In this image completion technique, given missing pixels to be reconstructed in a reference view, disparity-mapped patches on the other view are compared as well as patches in the same view in the distance measure. The search for candidates is constrained to patches that are further or at the same reconstructed depth value than the centre pixel of the target patch. This avoids copying texture from foreground objects to background regions. Contrary to Wang et al. [168],
where disparity map estimation is required before the user indicates which object should be removed, inpainting of a priori missing areas is possible. Moreover, in the example displayed in figure 5.7, the inpainting results of Hervieu et al. [67] show a better view consistency than the results of Wang et al. [168].

An improvement to their previous technique is presented in Hervieu et al. [68]. During the disparity reconstruction step, plane-fitting similar to Wang et al. [168] is applied to the depth map in the reference view. Similar depth planes with different colours are then merged via Graph Cuts [20]. Then, the remaining missing disparity is filled in via a variational inpainting method on each depth region. This is followed by a disambiguation step that assigns the depth of the layer closest to the camera to the output. During the colour reconstruction step, picture information is synthesised for each depth layer separately. The search for candidates is restricted to pixels nearby and within the corresponding depth layer. Once the reference view completed, the other image is filled in by warping pixels across views with the disparity map. Binocular half-occlusion areas are then reconstructed with the previous method [67].

Abe and Shimizu [1] argue that Hervieu et al. [68] can generate visible depth and colour inconsistencies across views within a stereo pair. In order to guarantee depth consistency between the two views, mutual completion followed by the first step from Hervieu et al. [68] is applied on both disparity maps iteratively, until corresponding pixels across views have the same depth. Mutual completion is then applied to the colour images. The remaining pixels are filled in via image inpainting based on an energy minimisation framework [84]. The image inpainting technique is extended to force intensities of corresponding pixels across views to be equal, so as to guarantee colour consistency. The total energy is the sum of the energies in the left and right views. The search is constrained as well to pixels within the same depth layer, which allows to speed up the global optimisation process.

5.3.3 Video inpainting for Multi-view Displays

This section focuses on inpainting methods that are designed especially to generate occlusion data in videos for view synthesis in multi-view displays. Occlusion data can be defined as the unknown picture information revealed behind foreground objects, due to parallax, when artificially changing the viewpoint by view synthesis techniques. The unknown areas are located at the border of foreground objects and should be completed by data from the background. In this section, we first review techniques that extend exemplar-based image inpainting by exploiting information in the depth map, and generate occlusion data in a frame-by-frame process [38, 39, 108], without explicitly enforcing temporal consistency of the output. Then, we detail how temporally coherent occlusion data can be obtained [63, 98].

**Depth-based Inpainting for View Synthesis Applications:** Generation of occlusion data for a video has been achieved in some techniques by extending the exemplar-based method of Criminisi et al. [34], and applying image inpainting to each frame independently [38, 39, 108].
Results from such a technique (Daribo and Saito [39]) are displayed in figure 5.6. In Luo et al. [108], to compare patches, the difference between the average depth of valid points in the target patch and the average depth in the candidate source patch is added to the colour difference term. As this technique is concerned with reconstruction of background data, a penalty term is also added to favour candidate textures from equal or larger depth areas. The main contribution of Daribo et al. [38, 39] is to modify the priority measure from the original exemplar-based framework. They define priority as the product of three factors: the usual data and confidence terms, and a level regularity term. This extra term is defined as the inverse variance of the depth in a patch. It gives higher priority to patches lying in a consistent depth level, which favours background pixels over foreground. A term is added to the distance measure to compute the Sum of Squared Differences over depth patches. This also biases the selection of patches within the same depth level as the target. Compared to traditional image inpainting [34], these methods show better preservation of the contours of foreground objects, as can be observed in figure 5.6. However, no temporal consistency is considered, so flicker artifacts can appear in the generated occlusion data. Note that depth completion is not used by Luo et al. [108], but Daribo et al. [38] use a diffusion process to inpaint the depth map in a preprocessing step.

**Temporally Consistent Inpainting for Occlusion Generation:** Gunnewiek et al. [63] combine the output of three inpainting techniques to generate temporally consistent occlusion data at each frame of a video. This process is illustrated in figure 5.8. The first output is obtained by image inpainting within the current frame. The second one by a mutual completion technique [7] to warp binocular half-occlusion areas across views. The last one is yielded by temporal inpainting to exploit data available in the previous and next frames. The temporal inpainting algorithm uses onion peel ordering of missing pixels at the current frame. For each target pixel, non-occluded candidates in the background are searched for along time using averaged motion vectors within a tracking patch. The reconstructed colour at the target is then a mix of the colours of the candidates with weights depending on their similarity. Spatiotemporal patches from the background are compared using the Sum of Absolute Differences of colour and depth values. Finally, the reconstruction is blended with the motion-compensated output from the previously completed frame. A reliability map reflecting the quality of the best match selected for replacement at each pixel is assigned to the output of each technique. If no candidate can be found for replacement, the reliability value assigned to the pixel is null. This reliability map allows to weight the merging, increasing the influence of different results depending on where they perform the best. This allows to filter out some artifacts. The reconstruction thus obtained is temporally smooth but can appear overly blurred in areas where the reliability is low for all techniques, as can be observed in figure 5.8.

Lee and Kim [98] first reconstruct the depth map at each frame via an image completion method derived from Criminisi et al. [34]. The priority measure is the product of the usual confidence and data terms and a third term to take occlusion constraints into account. This term
5.3. Inpainting for Stereoscopic Images and Videos

Figure 5.8: Illustration extracted from Gunnewiek et al. [63]. This technique combines the output of three inpainting techniques. The mutual completion result [7] is labelled 3-view in this drawing.

is function of the average depth value within a patch and allows to assign a higher priority to areas further from the viewpoint, which correspond to the background. This order of reconstruction allows a better preservation of the boundaries of foreground objects. Once the depth maps reconstructed, affine camera motion is estimated along the sequence. Background pixels at each frame are then warped to a reference frame. Depth is used as well to combine all frames in a spatiotemporal background volume. Pixels overlapping due to depth ambiguities are submitted to a median filter. Finally the reconstruction step consists in mapping the pixels from the volume back to the missing regions. Remaining missing pixels are occluded in all frames so their values are inferred by the same image completion method that is used to process depth maps. Completed pixels are used to update the background volume so as to keep temporal consistency.
throughout. The results exhibit good temporal consistency without blurring.

5.4 Final Remarks

Many techniques have been developed for image inpainting. Reconstructed missing areas in still images need to blend seamlessly with the known surrounding data. As stated in section 5.1, it seems that exemplar-based techniques [34] have been the most successful at achieving this goal. Their main advantage compared to early techniques such as variational inpainting [14] is their capacity to preserve complex texture in large regions. The exemplar-based framework has been extended by subsequent techniques to process videos and stereoscopic data.

For video inpainting techniques, studied in section 5.2, it is crucial to preserve temporal consistency in the reconstructed area. To achieve this goal, state-of-the-art algorithms employ a combination of motion segmentation [144], motion repair [88] and object tracking [74]. In motion-based approaches, temporal smoothness is often obtained at the cost of image sharpness [23, 88, 145]. This can be acceptable as the human visual system is more sensitive to discrepancies in motion than texture [172]. But in some cases, the blur is so high that details are missing in the reconstructed area. On the other hand, some techniques use at their core a patch-based approach [111, 119]. They have a greater ability to preserve sharp details inside the reconstructed missing area, but they also have a greater difficulty to deal with complex camera motion. It seems to be difficult to combine the best of both worlds, so more research is needed in that direction. In addition, there are many remaining challenges for completion of videos, such as handling dynamic textures, or complex scenes with camera motion and moving objects.

As we have seen in section 5.3, inpainting of stereoscopic content has attracted a great deal of interest lately due to the developments of 3DTV and 2D-to-3D video conversion. Processing two views brings in new challenges because the reconstructed area in both images of a stereo pair must preserve view consistency. Asymmetry in colour and texture must be avoided as it is a source of discomfort for the viewer. So far, efforts in the literature have focused on object removal in a single pair of frames [168] or generation of occlusion data on videos for specific view synthesis applications [7, 98]. The general problem of object removal for stereoscopic videos has not been studied extensively and should be investigated more.

In the following chapter we present our work on stereo-video inpainting. Spatial coherence and image sharpness are maintained in the reconstruction by building on the exemplar-based framework presented in section 5.1.2. The technique has been designed to incorporate specific mechanisms to try and solve the problems of temporal smoothness and view consistency, discussed in sections 5.2 and 5.3 respectively, within the same framework.
In the previous chapter, we have discussed how the problem of inpainting is solved by state-of-the-art algorithms. We have reviewed solutions aimed at processing still images, videos and stereoscopic media. In this chapter, we present our research on stereo-video inpainting. The main novelty of our technique is to consider temporal smoothness and view consistency constraints simultaneously within an exemplar-based framework.

The ultimate goal is object removal for stereo-3D film post-production applications. Given user-selected regions in a stereoscopic video, the objective is to fill in this area using available picture information. As highlighted in the previous chapter, three types of constraints must be fulfilled to achieve the best reconstruction: spatial coherence, temporal smoothness and view consistency. Many existing algorithms lack temporal consistency, causing flickering and other artifacts. And view consistency is a fairly novel problem which has received attention in the literature only recently, often for the specific problem of occlusion generation, which is described in section 5.3.

This chapter explores the use of long-term picture information in conjunction with inter-view dependencies to enforce temporal smoothness and view consistency within the same framework. In section 6.1 we present the inpainting problem for stereo videos. Then, in section 6.2 we describe our exemplar-based inpainting framework. The subsequent sections explain the building blocks of our algorithm. Section 6.3 details our patch-matching strategy. Section 6.4 shows how the best candidates are selected in the source region. Finally, section 6.5 describes how picture data is reconstructed with the selected candidates.
Our modifications to the exemplar-based framework: Our stereo-video inpainting technique follows at its core an exemplar-based image inpainting technique similar to Criminisi et al. [34]. The exemplar-based framework consists in copying picture information from best matching patch candidates to fill in the missing area. It is reviewed in greater details in section 5.1.2.

According to the literature, patch-copying mechanisms similar to Criminisi et al. [34] allow for a good spatial coherence in the output. However, the original formulation has been designed to process monoscopic still images. It does not incorporate temporal or view consistency constraints. As explained in the previous chapter, these constraints must both be enforced to obtain a good reconstruction for missing data in stereoscopic videos. This chapter presents our contributions to the exemplar-based framework to process stereo videos. Our technique uses motion and disparity to search for good candidates within a constrained texture synthesis process. A temporally smooth reconstruction, which also maintains view consistency, is generated by merging the selected candidates to the surrounding data via a dedicated filtering step.

Motion-based approaches to solve the inpainting problem, such as rig removal [88], use reconstructed motion vectors to find exactly the location of the missing data in other parts of the video. Our technique uses sampling nearby missing point trajectories to extend this motion-based approach. If the missing data is found along the reconstructed trajectory, we want to select it for replacement. Otherwise, we hope to find a good candidate by exemplar-based patch matching [34]. Our method can therefore be considered as a hybrid between motion-based and exemplar-based techniques, applied to stereo footage.

6.1 Problem Statement

Our concern is the inpainting of a stereoscopic video \( V \). A pixel site \( x \in V \) is made up of four coordinates \( x = (x, y, f, v) \). The spatial coordinates of the pixel within a frame are noted \((x, y)\). And its stereo-temporal position is denoted by \((f, v)\), where \( f \) is the frame number and \( v \) is the view. We denote as \( F \) the number of frames in the video. There are two views in a stereo video, one for each eye, denoted as left \( L \) and right \( R \).

6.1.1 Overview

The task is to reconstruct the unknown picture information in an initial target region \( T_0 \), which contains the pixel sites that are initially missing. The problem is usually solved in an iterative manner. At a given iteration \( i \), the remaining unknown picture information is contained in \( T_i \). The missing area is highlighted in blue, on both views, in the example shown in figure 6.1. The reconstructed output is shown in the rightmost column of the figure. To replace the missing pixels, following the exemplar-based framework, candidates are selected from a source region \( S \), defined such that:

\[
S \subseteq V \setminus T_0
\] (6.1)
6.1. Problem Statement

Figure 6.1: Illustration of our stereo-video inpainting technique on the first stereo-pair of frames of the \textit{rose.garden.tracking} sequence. The first column from the left shows the original frames. In the central column, the area highlighted in blue indicates an artificial degradation that we wish to fill in. The rightmost column shows the output of our algorithm. Note that the degradation hides the twist in the strap for 10 consecutive frames before it is revealed.

We define a binary mask $M^i_x$ indicating known pixels at the $i$th iteration by the value one. This mask provides information on the current state of the reconstruction. The initial mask $M^0_x$ is supplied by the user. Pixels that have been inpainted in a previous step are indicated by one as well in the current mask $M^i_x$.

At each site $x$, an image or picture value $I_x$ is encoded and stored. We use the \textit{RGB} colour space, so that $I_x = [I^r_x, I^g_x, I^b_x]$. Values in each channel $r$, $g$ and $b$ are in the interval $[0, 255]$. The problem is to attribute a picture value to all missing pixels in the target region $T_0$ so that the reconstruction \textit{matches} the known surrounding information. Our matching process is detailed in section 6.3.

The missing information is reconstructed by \textit{selecting} good candidates in the source region, as explained in section 6.4. We denote by $S_x$ the selected candidate site corresponding to a missing pixel site $x$. This determines how the unknown picture value at $x$ is reconstructed by the inpainting algorithm.

In the final inpainted video, the value $I_{S_x}$ at the selected candidate is used to reconstruct the missing picture value $I_x$. More precisely, the value $I_{S_x}$ is filtered to take into account colour correction and stereo-temporal blending, as detailed in section 6.5. This yields a modified value $\hat{I}_{S_x}$, which is used to replace the missing data.
6.1.2 Notions of Pixel Neighbourhood and Data Patch

Intuitively, exemplar-based inpainting can be seen as putting together fragments of stereo video data from the source so as to yield a coherent reconstruction in the target region. Coherence is measured as how well the reconstructed data fits the surrounding known information. The surrounding information at a pixel site \( x \) is contained in its \textit{stereo-temporal neighbourhood}, denoted as follows:

\[
\mathcal{N}_x = \mathcal{N}_x^\circ \cup \mathcal{N}_x^+ \cup \mathcal{N}_x^- \cup \mathcal{N}_x'
\] (6.2)

Consider a missing pixel at a site \( x \) in the target region on the example from figure 6.1. The neighbourhood of this pixel site is represented schematically in figure 6.2. It contains sites taken from the same frame as \( x \), which are contained in \( \mathcal{N}_x^\circ \). As we are concerned with the processing of stereo videos, we extend the definition of the neighbourhood accordingly. Therefore, it also contains neighbouring sites in the next frame \( \mathcal{N}_x^+ \), in the previous frame \( \mathcal{N}_x^- \), and across view \( \mathcal{N}_x' \). Each of these four sets reside in one frame plane.

The problem is to determine whether copying known image data around a candidate pixel site \( y \), to fill in missing data around a target missing pixel location \( x \), would yield a coherent reconstruction. According to the exemplar-based framework, we can evaluate whether \( y \) is a good candidate by computing the similarity or difference between surrounding image patches around \( x \) and \( y \). We define the image patch \( I \left[ \mathcal{N}_x \right] \) as the set of all image values taken at sites in the neighbourhood \( \mathcal{N}_x \):

\[
I \left[ \mathcal{N}_x \right] = \{ I_p \mid p \in \mathcal{N}_x \} \quad (6.3)
\]

Not every pixel in the neighbourhood has the same importance for the evaluation of the distance between two image patches. First of all, it is intuitive to assume that pixels that are further away from the centre of a patch should not count as much as pixels close to the centre. We follow the approach of Bornard et al. [18] by using 2D Gaussian weights, when comparing the information between corresponding spatial patches. Given an offset \( u \), the weight \( w_u \) is defined as follows:

\[
w_u \propto \exp \left( -\frac{x_u^2 + y_u^2}{2\sigma_w^2} \right) \quad \text{where} \quad u = (x_u, y_u, \ldots)
\] (6.4)

In our experiments, we define the parameter \( \sigma_w = h/2 \) depending on the patch height \( h \). The value of \( w_u \) is normalised so that the weights sum to 1 on all possible offsets for a given spatial neighbourhood. The spatial weights are represented in the leftmost column in figure 6.3.

On figure 6.2, the height of each neighbourhood is \( h = 5 \) pixels. However, in our experiments we use \( h = 11 \) pixels. Determining the optimal size \( h \) for a patch is an open problem. If the patch is too small, the probability of incorrect matching is increased. On the other hand, if the patch is too large, it is impossible to find a good match for it. In our experiments, we set the value of \( h \) after a heuristic comparison between different values. According to the literature, the size of a patch should be adapted automatically depending on the size of the missing area [74] and characteristics of the local texture [72]. We leave this problem for future work.
6.1. Problem Statement

Forward motion vector
Disparity vector
Backward motion vector
Pixel sites in the target region
Pixel sites in the source region
Centre pixel site \( x \)

Figure 6.2: Illustration of the stereo-temporal neighbourhood system we use in our technique. This drawing represents pixel sites by colour-coded dots. The target area, which we want to reconstruct is shown in gray. Sites belonging to the neighbourhood \( N_x \) centred around a point \( x \) are coloured in blue or red if they also belong to \( N_x^0 \), in the same frame as \( x \), on the same view \((f, L)\). Some of the information is missing in \( N_x^0 \). Missing pixels are coloured in red and known pixels are shown in blue. The sites belonging to \( N_x^+ \), in the next frame on the same view \((f + 1, L)\), are found around the motion-compensated position \( x^+ \), using the forward motion vector \( U_x^+ \). They are coloured in orange. The sites belonging to \( N_x^- \), in the previous frame on the same view \((f - 1, L)\), are found around the motion-compensated position \( x^- \), using the backward motion vector \( U_x^- \). They are coloured in green. The sites belonging to \( N_x' \), in the frame on the other view \((f, R)\) are found around the disparity-compensated position \( x' \), using the inter-view disparity vector \( U_x' \). They are coloured in purple.
Spatial weights on a square $11 \times 11$ neighbourhood

Known pixels around a target pixel site

Known pixels around a source pixel site

Spatial weights restricted to the sites of common known pixels

Figure 6.3: Illustration of the principles of patch-based weighted comparison. The values of the spatial weights $w_u$ are shown in an $11 \times 11$ neighbourhood on the leftmost column. Values are normalised to $[0, 1]$ for display purposes. They are coded in grayscale, with black for zero and white for one. Consider a missing pixel $x$ and a candidate $y$. The two patches in the centre represent the available and missing data in patches around $x$ and $y$, respectively, in this order. Known pixels are indicated in green and unknown pixels in red. After alignment, unknown sites in either patch are marked by zero in the weights displayed on the rightmost column.

As can be seen in figure 6.2, the image value at $x$ is unknown, and some pixels around $x$ are also missing. Unknown pixels within the neighbourhood $N_x$ are indicated by a corresponding mask value of zero in the mask patch $M^i [N_x]$. These pixels do not contain any picture information, as illustrated in the two central columns in figure 6.3. When comparing two corresponding patches, these pixel values should be weighted out, as illustrated in the rightmost column of figure 6.3.

6.1.3 Usage of Motion and Disparity Vectors

As illustrated in figure 6.2, the pixel neighbourhood $N_x$ is made of four motion-aligned and disparity-compensated square neighbourhoods. Therefore, it is necessary to know the local disparity and motion information at $x$, denoted as $U_x$. Assuming this information is known, the forward motion vector $U_x^+$ allows to locate the centre of $N_x^+$ in the next frame, i.e. $x^+ = x + U_x^+$. Similarly, the backward motion vector allows to locate the centre of $N_x^-$ in the next frame, which we note $x^- = x + U_x^-$. Finally, the disparity vector $N_x'$ allows to locate the centre of $N_x'$ in the other view, which we note $x' = x + U_x'$.

The problem is that motion and disparity information is unknown in the missing area. We estimate this data during a preprocessing step. Firstly, we estimate forward and backward motion, as well as inter-view disparity vectors, on the known source data in the sequence. This is achieved with standard tools available in Ocula\(^1\), a suite of stereo-processing plug-ins for Nuke\(^2\). We tune the parameters of the motion and disparity estimators in Nuke, according to

\[^1\text{version 3.0} – \text{https://www.thefoundry.co.uk/products/ocula/}\]

\[^2\text{version 6.3} – \text{https://www.thefoundry.co.uk/products/nuke-product-family/}\]
6.1. Problem Statement

The scene is shown in the leftmost column. The user-defined mask is highlighted in blue around the walking man. The sphere on the left of the scene is rotating around an axis perpendicular to the ground. The man in the centre of the scene is walking to the right at a steady pace. The camera is hand-held, so some jitter motion is introduced by the operator. The central column represents colour-coded motion and disparity vectors corresponding to this scene, and obtained using Ocula. The rightmost column shows the vectors after motion interpolation in the target area. As can be seen, the motion of the walking man has been removed in the interpolated vector fields.

A more delicate step is the estimation of the motion and disparity fields inside the missing area. Our experimental approach is to propagate the information from the boundary of the target region towards its centre, frame by frame. We do so by using a still image inpainting technique similar to Bertalmio et al. [14], on every channel of the motion and disparity vectors independently. In our tests, we use the Matlab implementation by John D’Errico which is freely available[^3]. As illustrated in figure 6.4, this simple scheme gives satisfying results in most cases where motion and disparity fields are smoothly varying in space inside the missing area.

However, reconstructed vectors can be inaccurate when complex motion occurs inside the missing region. More complex methods such as Kokaram et al. [88], Wang et al. [168] or Shiratori et al. [145] should be considered for large holes containing high variations of motion or disparity, but have not been used in our experiments. As explained in sections 6.3 and 6.4, we compensate for the inaccuracy of reconstructed motion and disparity by combining the use of displacement

vectors with a patch-matching strategy in our hybrid approach.

Motion and disparity vectors are used throughout our technique, to enforce temporal and view consistency constraints. Besides data alignment to construct stereo-temporal neighbourhoods, we explain in section 6.4 how motion and disparity vectors are also used as guides for candidate search. Once they are reconstructed, we consider motion and disparity as parameters in the rest of this chapter.

6.2 A Probabilistic View of Examplar-based Inpainting

This section details and motivates the use of an exemplar-based framework for inpainting of stereo videos. We note Θ a data set containing all the parameters that control our algorithm, including reconstructed motion and disparity information $U$, the initial mask $M^0$, and various parameters such as $h$ and $\sigma_w$. We need to have access to this data to compute the various conditional density functions described in what follows.

6.2.1 Patch Matching and Sampling

Following previous works on exemplar-based inpainting [18, 45, 94], we model the picture data as a Markov Random Field. Therefore, given a pixel site $x \in V$, the image value $I_x$ depends only on the image values in the patch $I[N_x]$. It is then possible to establish a generative model that conditions the image value at each pixel. If the conditional probability density function can be modelled, it is possible to draw samples from it to replace the missing data.

The contribution of Efros and Leung [45], which has been exploited in subsequent works such as Criminisi et al. [34] is to provide a practical way to sample from the conditional distribution, using known picture data. As explained in the previous section, the aim is to find a suitable candidate site $y \in S$ to replace the missing information at a target pixel site $x \in T$. Under the Markovian assumption, the unknown image value at $I_x$ only depends on the data in the image patch $I[N_x]$. If the picture data in patch $I[N_y]$ is similar to the data in patch $I[N_x]$, the fundamental principle of exemplar-based inpainting techniques is to consider that the unknown image value $I_x$ can be replaced by the known image value $I_y$.

Given the available example picture data in the patch $I[N_x]$, the objective is to find the most likely candidate site $S_x$ in the source. This is done by comparing the example target patch to the available picture information in candidate patches $I[N_y]$. Only the known data marked by one in the mask patches $M^i[N_x]$ and $M^i[N_y]$ can effectively be exploited for patch comparison. The known picture value at $S_x$ is then used to replace the corresponding missing picture data at the site $x$. Which means that the best candidate satisfies the following equation:

$$S_x = \arg \max_{y \in S} p(y \mid I[N_x], I[N_y], M^i[N_x], M^i[N_y], \Theta)$$  \hspace{1cm} (6.5)

To compare the information between two picture patches, most inpainting methods [18, 34, 94, 144, 168, 171] define a distance measure $D[N_x, N_y]$. Distance values are defined within $[0, 1]$,
6.2. A Probabilistic View of Exemplar-based Inpainting

and a small distance signifies that the patches are similar. The ability to compare picture patches is at the basis of patch-matching strategies for inpainting. Given the distance measure $D$ and a fixed parameter $\sigma_D$, the function on the right-hand side of equation 6.5 can then be made explicit:

$$p(y \mid I[N_x], I[N_y], M^x [N_x], M^y [N_y], \Theta) \propto \exp\left(-\frac{D^2 [N_x,N_y]}{2\sigma_D^2}\right)$$ (6.6)

In our experiments, a typical value for the parameter $\sigma_D$ is 0.5. The previous two equations are often not mentioned in the inpainting literature, and the objective is usually formulated directly in terms of distance minimisation:

$$S_x = \arg\min_{y \in S} D[N_x,N_y]$$ (6.7)

6.2.2 Order of Filling for Greedy Reconstruction

Techniques such as Criminisi et al. [34] have adopted a greedy approach to solve the inpainting problem for all missing pixels. This has been discussed in the previous chapter, in section 5.1.2. An important part of such techniques is the order in which pixels in the target area are processed. To determine this order of filling, a priority measure is computed on missing pixel sites $x$ in the target region. The site with the highest priority is processed first. After the missing value at this site is filled, the priority is updated and the pixel site is removed from the target region.

In the original paper [34], priority is defined as the product of a confidence term $C(x)$ and a data term $D(x)$. Therefore, at each iteration $i$, the pixel site $x_i$ to be processed in the remaining target region $T_i$ is selected according to the following equation:

$$x_i = \arg\max_{x \in T_i} C(x) D(x)$$ (6.8)

This priority-based ordering can be re-cast in a probabilistic framework, by realising that we wish to process first missing pixels that are the most likely to be reconstructed accurately. According to the literature, these sites are found in areas close to many reliable or known pixels [18], and close to textured areas or strong edges [34,144].

The image patch distance $D$ is often used to give an indication of the reliability $R_x$ of the reconstruction [63,144,171]. By definition, the reliability of pixels in the source region is equal to one, and the value for missing pixels that have not been reconstructed yet is zero. For reconstructed pixels, we compute the reliability as $R_x = 1 - D[N_x,N_{S_x}]$. Therefore, the confidence term $C(x)$ can be considered to be proportional to the value of a probability density function $p(x \mid R[N_x], \Theta)$, given the reliability patch $R[N_x]$.

Criminisi et al. [34] use image isophotes to compute the data term. A simplification to the computation suited to process videos is used by Shih et al. [144], where detected edges are used as an indicator of the image structure within a video frame. Accordingly, we define binary image edges $E_x$ such that detected edges are indicated by one. In our experiments, we use the standard Canny edge detector, available in Matlab, during a preprocessing step. Figure 6.5 shows the
Figure 6.5: Illustration of the binary edge detection process. A stereo pair of frames with the missing area highlighted in blue is displayed on the left. The detected edges in the known area using the Canny method are shown on the right.

detected edges in the source region in a stereo pair of frames. By definition, initial edge data in the target region is set to zero. As explained in section 6.5, edge data in the missing region is then filled in at the same time as picture data is reconstructed using the value $E_{S_x}$. We can therefore consider the data term $D(x)$ to be proportional to the value of a probability density function $p(x \mid E[N_x], \Theta)$, given the edge patch $E[N_x]$.

Determining the current pixel to process at iteration $i$ can then be expressed as solving the following equation:

$$x_i = \arg \max_{x \in T_i} p(x \mid R[N_x], \Theta) p(x \mid E[N_x], \Theta)$$

(6.9)

In section 6.3, we give details on our approach to compute the order of filling.

6.2.3 Our Extended Objective Function for Patch Matching

The objective function formulated in equation 6.5 takes into account neither temporal smoothness nor view consistency of the inpainted region. As explained in the previous chapter, in section 5.2, the need for a temporally consistent reconstruction of videos has been recognised in the literature. Likewise, view consistency constraints must be imposed when reconstructing stereoscopic material, as shown in section 5.3.

Such constraints are often imposed in the choice of the source region [144] or in the formulation of the distance measure [38]. Sometimes, it is implicitly recognised that neighbouring
6.2. A Probabilistic View of Examplar-based Inpainting

selected candidates must be used to replace corresponding neighbouring missing sites. An example of which is the use of coherence search \[4\] in Bornard et al. \[18\]. More often, consistency constraints are enforced by computing patch distances on overlapping patches within a global optimisation framework \[94,105,171\].

There is a fundamental difference between image inpainting and video inpainting: in the latter, the missing data might become available in other frames. In a stereo video there is an even higher probability that the missing data will re-occur somewhere else, possibly in the other view. In case the missing information becomes uncovered, the problem is to find where, using motion and disparity vectors, similar to rig removal approaches \[88\]. In case it cannot be found exactly, it is necessary to revert to texture synthesis, using a technique similar to Efros and Leung \[45\] and still generate a consistent reconstruction. The goal of our hybrid technique is to combine both approaches within the same framework.

Assume that the missing pixel with the highest priority \(x_i\) has been found according to equation 6.9. Firstly, we base the selection of the best candidate \(S_{x_i}\), chosen among candidates \(y \in S\), on the probability density function depending on image patches similarity, detailed in equation 6.6. We recognise the potential of further developments in equation 6.5 to reduce the number of candidates, within a greedy patch-filling process.

**Useful Information in a Patch:** The surrounding mask data \(M^i[N_{x_i}]\) can also be exploited to find the most informative candidates. For such points, non-missing pixels in \(M^0[N_Y]\) are valid candidates for replacement of corresponding missing points around \(x_i\). These candidates can be expected to have a high quantity of useful information in their vicinity. This information is deemed useful not only because it provides support for patch matching, but also because the available pixels can be copied directly to replace corresponding missing data. Quantifying this information helps selecting the best candidate in case the missing area is never totally revealed along the video.

**Source Frame Consistency:** One way of reducing the amount of potential candidates is to condition the selection of the best candidate on the prior reconstruction. Intuitively, neighbouring missing pixels would be expected to have neighbouring best candidates for replacement. Therefore, we bias the selection towards candidates having a close stereo-temporal proximity to previously selected points, selected with a high reliability. To do so, we use local information on the origin of selected candidates for replacement around the missing site \(S[N_{x_i}]\), as well as their corresponding reliabilities \(R[N_{x_i}]\).

Based on these considerations, in our technique, the best candidates \(S_{x_i}\) are searched for and selected according to the following patch-matching formulation:

\[
S_{x_i} = \arg \max_{y \in S} p(y \mid I[N_{x_i}], I[N_Y], M^i[N_{x_i}], M^i[N_Y], \Theta) p(y \mid M^i[N_{x_i}], M^0[N_Y], \Theta) p(y \mid S[N_{x_i}], R[N_{x_i}], \Theta) \quad (6.10)
\]

In section 6.3 we make explicit the two terms in the right-hand side of equation 6.9 as
well as the three terms in the right-hand side of equation 6.10. Our formulation incorporates temporal and view consistency constraints within the patch-matching procedure in a unified way. In section 6.4, we show how the source region is reduced to constrain the search and lower the computation burden of our algorithm by using the reconstructed motion and disparity vectors. Then, the values of \( I, M', E \) and \( R \) are updated, following the rules stated in section 6.5. Subsequently, the algorithm proceeds to reconstruct the next missing pixel at \( x_{i+1} \) in the updated target region \( T_{i+1} = T_i \setminus \{x_i\} \). The inpainting technique stops when there is no remaining unknown pixel in the target region.

### 6.3 Our Stereo-Video Patch Matching Strategy

In section 6.2, we have presented our exemplar-based inpainting framework. This section details the core of our stereo-video inpainting algorithm. We reconstruct the missing data with a greedy approach. Missing pixels are processed sequentially, according to a priority-based ordering, shown in equation 6.9. Then, candidates are searched for and selected in the source region to replace the unknown picture data. The best candidate satisfies equation 6.10. In this section, we make explicit the terms in the two equations mentioned above. The computation of these terms uses patch-based estimates and parameters contained in \( \Theta \).

#### 6.3.1 Computing the Order of Filling

As explained in section 6.2.2, we process missing pixels sequentially, according to their priority or order of filling. We must define the two factors in the right-hand side of equation 6.9 to make explicit the ordering used in our algorithm. The first one is called confidence term and the second one data term.

The confidence term is based on the reliability of the reconstructed data around a missing pixel \( x \). In a stereoscopic video, missing pixels close to the source region are likely to be uncovered in the neighbouring frames along time or across view. They are also surrounded by pixels with a high reliability, as the reliability is one for source pixels. We wish to process such pixels first. Given a fixed parameter \( \sigma_R \), the confidence term is then computed as follows:

\[
p(x \mid R[N_x], \Theta) \propto \exp \left( -\frac{(1 - \mu^R_x)^2}{2\sigma^2_R} \right)
\]  

(6.11)

The mean reliability \( \mu^R_x \) is closer to one for pixels surrounded by many known neighbouring pixels. We weight the influence of neighbouring points with the spatial weight \( w_u \) introduced in equation 6.4. Accounting for motion and disparity, the computation is as follows:

\[
\mu^R_x = \frac{1}{4} \left( \sum_{u \mid x^+ + u \in N^+_x} w_u R_{x^+ + u} + \sum_{u \mid x^- + u \in N^-_x} w_u R_{x^- + u} + \sum_{u \mid x' + u \in N'_x} w_u R_{x' + u} + \sum_{u \mid x' + u \in N'_x} w_u R_{x' + u} \right)
\]  

(6.12)
6.3. Our Stereo-Video Patch Matching Strategy

Figure 6.6: Illustration of the order of filling, computed according to the equations described in section 6.3.1. A stereo pair of frames from the sequence illustrated in figure 6.5 is displayed. On each view, the target area is located at the centre of the green rectangle. The priority of the missing pixel sites in the target area is represented in shades of gray, with black for the value zero and white for one. Brighter pixels are located around the border of the missing region, and close to strong edges. They are processed first. Darker pixels towards the centre of the target region are processed in subsequent iterations.

The data term indicates that missing sites $x$ near edges or textured areas possess more useful picture information, compared to sites in texture-less areas. Indeed, patch matching techniques based on gradient information use edges to align corresponding patches. Patches containing highly-textured image data can be aligned more easily. Therefore, given a fixed parameter $\sigma_E$, the data term is computed as follows:

$$p(x \mid E[N_x], \Theta) \propto \exp \left(-\frac{(1 - \mu^E_x)^2}{2\sigma_E^2}\right)$$

(6.13)

The mean edge value $\mu^E_x$ is closer to one for points surrounded by many detected edges. Note that we only use edge data in the same frame as $x$, because we are mainly interested in the texture information in this frame to reconstruct the missing data. The computation is as follows:

$$\mu^E_x = \sum_{u \mid x+u \in N^c_x} w_u E_{x+u}$$

(6.14)

In our experiments we set the parameters as $\sigma_R = 0.7$ and $\sigma_E = 0.5$ to give more weight to the data term. Figure 6.6 illustrates the value of the priority of missing pixels on a stereo-pair of frames. It can be seen that points close to the border of the missing region and close to strong edges have a higher value. Therefore, they are processed first by our algorithm. As explained in section 6.5.3, after the missing information at the current missing pixel $x_i$ has been replaced by our inpainting technique, the order of filling is updated. Now we have detailed how to select the target missing pixel to be processed at each iteration $i$, we turn to the objective function to optimise for candidate selection in the source. This objective function can be seen in equation 6.10.
6.3.2 Computing a Distance Measure for Image Patches

The first term in the right-hand side of equation (6.10) is defined in equation (6.6). It gives an indication on how well the surrounding image data $I_{N_x}$, around the selected site $x_i$ in the target region, matches the image data $I_{N_y}$, around a candidate $y$ in the source. We define the distance function $D_{N_x, N_y}$ used in our algorithm, to compare stereo-video patches of image data, as follows:

$$D_{N_x, N_y} = \frac{1}{\alpha^o + \alpha^+ + \alpha^- + \alpha'} \left( \alpha^o D_{N^o_{x_i}, N^o_y} + \alpha^+ D_{N^+_{x_i}, N^+_y} + \alpha^- D_{N^-_{x_i}, N^-_y} + \alpha' D_{N'_x, N'_y} \right)$$

(6.15)

Where the weights are parameters to balance the influence of each term. Typical values in our experiments are $\alpha^o = 1$, $\alpha^+ = 0.3$, $\alpha^- = 0.3$ and $\alpha' = 0.5$.

In what follows, we explain the computation of $D$ for square image patches taken in the same frame as their centre $I_{N^o_x}$. The definition is similar for the other patches in corresponding frames along time $I_{N^+_x}$, $I_{N^-_x}$, and across views $I_{N^o_y}$. Consider the patch $I_{N^o_{x_i}}$ around the missing site $x_i$, and the corresponding patch $I_{N^o_y}$ around the candidate site $y$, taken in the same frames as their respective centres.

The Sum of Squared Differences (SSD) has been widely used in the inpainting literature to compare image patches [18,34,39,94,119,144,168]. A contribution of our work is to use instead a distance measure derived from the structural similarity, presented in Wang et al. [170]. Structural similarity, or SSIM, is designed to put the emphasis on image structure when comparing patches. It is claimed to be closer to the human sense of visual quality than simpler measures such as the SSD. We then define our distance measure as the structural dissimilarity:

$$D_{N^o_{x_i}, N^o_y} = 1 - \text{SSIM}_{N^o_{x_i}, N^o_y}$$

(6.16)

The original formulation of the SSIM applies to grayscale image data. For data in the RGB colour space, we compute the distance on each channel independently, and then aggregate the results to generate the final value. Following the work of Galar et al. [55], we choose the dual of the geometric mean of the distance values at each channel, for its good performance in the task of stereo matching. This formulation boils down to defining the SSIM in equation (6.16) as the geometric mean of the structural similarity, on all three colour channels $c \in \{r, g, b\}$. We then process colour image patches as follows:

$$\text{SSIM}_{N^o_{x_i}, N^o_y} = \left( \prod_{c \in \{r, g, b\}} \text{SSIM}^c_{N^o_{x_i}, N^o_y} \right)^{\frac{1}{3}}$$

(6.17)

We can now detail the computation of the structural similarity, applied to each colour chan-
maximum operator to discard possible negative values. The weighted cross-correlation, as supplied by the user $M$

The second term in the right-hand side of equation 6.10 depends on data in the initial mask $B$. This means that the vectors can be inaccurate. Occlusions due to parallax or moving objects can also cause correspondences not to be appropriate, in the neighbourhood $N_{x_i}$, between sites in the same frame as $x$ and sites in other frames along time and across view. To account for these inaccuracies, we design a weight $\lambda_u$ to lower the contribution of mismatching pixels. Consider for example the data in the next frame, for which the weight is defined as follows:

$$\lambda_u^+ \propto \exp \left( -\frac{\left(I_{x_i,+u}^G - I_{x_i,+u}^B\right)^2}{2\sigma^2_{\lambda}} \right)$$

Where we use grayscale image values $I^G_{x_i} = 0.299 I_{x_i} + 0.587 I_{y_i} + 0.114 I_{z_i}$. In our experiments we set the parameter $\sigma_{\lambda}$ manually. Typically, we have $\sigma_{\lambda} = 10$. Uniform weighting is also possible by letting $\sigma_{\lambda} \to \infty$. We also define $\lambda_u^- \propto 1$ as the notion of mismatch does not exist in this case.

### 6.3.3 Exploiting the Amount of Useful Information in a Patch

The second term in the right-hand side of equation (6.10) depends on data in the initial mask supplied by the user $M^0$, and data in the mask that indicates the current state of the recon-

$$\text{SSIM}^e \left[ N_{x_i}^c, N_{y_i}^c \right] = \max \left( \frac{2\mu^L_{x_i} \mu^L_{y_i} + C_1}{\left(\mu^L_{x_i}\right)^2 + \left(\mu^L_{y_i}\right)^2 + C_1}, \frac{2\sigma^L_{x_i,y} + C_2}{\left(\sigma^L_{x_i,y}\right)^2 + \left(\sigma^L_{y_i}\right)^2 + C_2}, 0 \right)$$

Where $C_1$ and $C_2$ are small values introduced to improve numerical stability. We employ the maximum operator to discard possible negative values. The weighted cross-correlation, as well as the means and standard deviations are computed on each colour channel of image patches, centred at the missing site $x_i$ and candidate site $y_i$. We adapt the computation to our framework as follows:

$$\mu^L_{x_i} = \frac{1}{\kappa} \sum_{u | x_i,+u \in N_{x_i}^c} \lambda^0_u w_u M_{x_i,+u}^1 M_{y_i,+u}^1 I_{x_i,+u}^c$$

$$\left(\sigma^L_{x_i} \right)^2 = \frac{1}{\kappa} \sum_{u | x_i,+u \in N_{x_i}^c} \lambda^0_u w_u M_{x_i,+u}^1 M_{y_i,+u}^1 \left( I_{x_i,+u}^c - \mu^L_{x_i} \right)^2$$

$$\sigma^L_{x_i,y} = \frac{1}{\kappa} \sum_{u | x_i,+u \in N_{x_i}^c} \lambda^0_u w_u M_{x_i,+u}^1 M_{y_i,+u}^1 \left( I_{x_i,+u}^c - \mu^L_{x_i} \right) \left( I_{y_i,+u}^c - \mu^L_{y_i} \right)$$

where $\kappa = \sum_{u | x_i,+u \in N_{x_i}^c} \lambda^0_u w_u M_{x_i,+u}^1 M_{y_i,+u}^1$

The computation of $\mu^L_{x_i}$ and $(\sigma^L_{x_i})^2$ can be deduced from the first two equations above by replacing $x_i$ by $y_i$. The term $\kappa$ is a normalising value. The mask $M^1$ is used to weight out unknown data during comparison, as explained in figure 6.3. The next paragraph explains the role of the weight $\lambda_u$ in the previous equations.

Remember that we infer the disparity and motion information at the missing locations during the preprocessing step detailed in section 6.1.3. This means that the vectors can be inaccurate. Occlusions due to parallax or moving objects can also cause correspondences not to be appropriate, in the neighbourhood $N_{x_i}$, between sites in the same frame as $x$ and sites in other frames along time and across view. To account for these inaccuracies, we design a weight $\lambda_u$ to lower the contribution of mismatching pixels. Consider for example the data in the next frame, for which the weight is defined as follows:
struction \( M^i \). In the former, only pixels in the source are marked by one. In the latter, locations of pixels that have been inpainted previously are also marked by one.

Remember that we wish to reconstruct the missing image data at \( x_i \), using the available data at \( y \). If neighbouring pixels around \( y \) are also known, they are likely to provide good supporting information for patch comparison with the distance \( D \) defined previously. Following the coherence search principle \([4]\) used by Bornard et al. \([18]\), they are also likely to be good candidates to replace corresponding pixels around \( x_i \). We quantify the probability of having useful information around the candidate site \( y \) as follows:

\[
p(y \mid M^i[N_{x_i}], M^0[N_y], \Theta) \propto \exp\left(-\frac{(1 - \mu_{x_i,y}^M)^2}{2\sigma_M^2}\right)
\]

In our tests, we set the scale parameter to \( \sigma_M = 0.7 \). The mean mask value \( \mu_{x_i,y}^M \) is closer to one for points surrounded by many known points. We also give extra weight to pixels in the neighbourhood of \( y \), that could be used to replace corresponding pixels in the neighbourhood of \( x_i \), that are actually missing. For these points, the mask value \( M^i \) is equal to zero. Given a parameter \( \alpha_M \in [0, 1] \), we quantify this in the following formulation:

\[
\mu_{x_i,y}^M = (1 - \alpha_M^M) \sum_{u \in N_y^\circ} w_u M_{y+u}^0 + \alpha_M \sum_{u \in N_{x_i}^\circ} w_u (1 - M_{x_i+u}^i) M_{y+u}^0
\]

The first term from the left quantifies the amount of information available around \( y \) and the second term quantifies the amount of available information around \( y \) that is missing in corresponding locations around \( x_i \). In our experiments, we set the parameter such that \( \alpha_M = 0.9 \) to increase the influence of new data available around the candidate \( y \). Data in the spatial neighbourhood \( N_y^\circ \) only is used during computation because in our image reconstruction step we only consider this data to fill in missing pixels in \( N_{x_i}^\circ \). Remaining unknown pixels in the stereo-temporal neighbourhood \( N_{x_i} \) are inpainted during a subsequent iteration.

### 6.3.4 Exploiting Prior Information for Source Frame Consistency

The third term in the right-hand side of equation \[6.10\] makes use of prior information from previous iterations of the algorithm. It involves the location \( S[N_{x_i}] \) and the reliability \( R[N_{x_i}] \) of selected candidates for pixels that have already been reconstructed, around the current target missing site to process \( x_i \). We use this information to impose a bias towards selecting candidates from the same frames. We decompose this term along time and across views using the formula of total probability. We model the total distribution as a mixture of Gaussian distributions:

\[
p(y \mid S[N_{x_i}], R[N_{x_i}], \Theta) = \sum_{f \in \{1, \ldots, F\}} \sum_{v \in \{L, R\}} p(y \mid f, v, \Theta) p(f, v \mid S[N_{x_i}], R[N_{x_i}], \Theta)
\]
is defined as a 2D Gaussian kernel:

$$p(y | f, v, \Theta) \propto \exp\left(-\frac{(f_y - f)^2}{2\sigma_f^2}\right) \exp\left(-\frac{(v_y - v)^2}{2\sigma_v^2}\right)$$

where $y = (\cdot, \cdot, f_y, v_y)$ \hspace{1cm} (6.27)

The parameters $\sigma_f$ and $\sigma_v$ are selected manually in our tests to control the bandwidth of the Gaussian kernel. Typically $\sigma_f = 5$ and $\sigma_v = 1$. The lower value for $\sigma_v$ makes it less likely to select a candidate $y$ from a different view to the origin of previously reconstructed points.

The second term in the sum, $p(f, v | S[N_{x_i}], R[N_{x_i}], \Theta)$ serves as a weight for the 2D Gaussian function defined previously. A higher weight is assigned to frames $(f, v)$ that appear more frequently in the patch $S[N_{x_i}]$. We also use the reliability information in the patch $R[N_{x_i}]$ to lower the weight of frames yielding a lower reliability. Given a parameter $\sigma_{S,R} = 0.7$, the weights are defined as follows:

$$p(f, v | S[N_{x_i}], R[N_{x_i}], \Theta) \propto \exp\left(-\frac{(1 - \mu_{S,R}^i)^2}{2\sigma_{S,R}^2}\right)$$ \hspace{1cm} (6.28)

In the equation above, the mean value $\mu_{S,R}^i$ is closer to one if selected candidates have been drawn from frame $(f, v)$ with a high reliability, in the stereo-temporal neighbourhood of $x_i$. Given a binary operator $[a == b]$, which is equal to one if $a = b$, the mean value is computed as follows:

$$\mu_{S,R}^i = \frac{1}{4}\left(\sum_{u|x_i^+ + u \in N_{x_i}^+} w_u R_{x_i^+ + u}\left[S_{x_i^+ + u} == (\cdot, \cdot, f, v)\right] + \sum_{u|x_i^- + u \in N_{x_i}^-} w_u R_{x_i^- + u}\left[S_{x_i^- + u} == (\cdot, \cdot, f, v)\right]
+ \sum_{u|x_i' + u \in N_{x_i}'} w_u R_{x_i' + u}\left[S_{x_i' + u} == (\cdot, \cdot, f, v)\right] + \sum_{u|x_i'' + u \in N_{x_i}''} w_u R_{x_i'' + u}\left[S_{x_i'' + u} == (\cdot, \cdot, f, v)\right]\right)$$ \hspace{1cm} (6.29)

We can now compute equation 6.10 for any candidate $y$ in the source area. However, searching for candidates in the whole source region is computationally inefficient. In the next section, we detail how the best candidates are searched for and selected via patch tracking and local optimisation. Patch tracking uses the reconstructed motion and disparity vectors. We also detail our coherence sewing mechanism that further reduces the computational burden of candidate search.

### 6.4 Candidate Search using both Motion and Disparity

After selecting the current target missing pixel site $x_i$, by solving equation 6.9, our algorithm searches for the best candidate $S_{x_i}$ to replace the missing data by solving equation 6.10. Searching for candidates on the whole source region within a stereoscopic video is inefficient. The
search area can be greatly reduced by exploiting possible data redundancy and taking into account stereo-temporal consistency. For each missing site $x_i$, we search for candidates in a subset of the source $\delta S_{x_i}$, which we call the candidate set. We have:

$$\delta S_{x_i} \subseteq S$$  \hspace{1cm} (6.30)

We perform a two-step search of the candidate set. The first step consists in searching for the best candidate along time and across view on the trajectory of the current missing pixel. The second step is a local optimisation procedure via gradient descent. We use patch tracking as a principled way to orient the search towards the most likely candidates. This reduces the amount of data in the candidate set to a thin band of pixels, along a missing pixel trajectory, in both views. In our tests, there is about 100 to 400 candidates in $\delta S_{x_i}$, depending on the length of the video.

### 6.4.1 Patch Tracking using Reconstructed Motion and Disparity Vectors

Initially, in exemplar-based image inpainting techniques such as Criminisi et al. [34], the candidate set is a large enough square area around the current missing pixel. Reconstructing a video frame by frame with such a technique produces temporal inconsistencies, such as flickering artifacts. A basic extension from exemplar-based image to video inpainting, proposed by Bornard et al. [18], uses motion-compensated previous and next frames to expand the candidate set. This is not sufficient in many cases where the missing information is occluded during several frames before being uncovered. Techniques such as Jia et al. [74] further expand the candidate set in a video by using patch tracking. This allows exploitation of long-term information, in frames that can be far apart from the current target frame, along the trajectory of the current target missing pixel. In our method, we also exploit long-term picture information along pixel trajectories.

In a stereo-pair of frames, both left and right views are almost identical. Inter-view correspondences can be found between all known pixels that are not in binocular half-occlusion areas. The main idea of exemplar-based inpainting techniques for stereoscopic material [7, 168] is to use the information available on the other view to fill in the current one. We also use that information in our method by considering the trajectory of the stereo-counterpart of the current missing pixel. Figure 6.7 summarises our approach for candidate search via patch tracking in a stereo video. We exploit the reconstructed motion and disparity information $U$ to guide the search for candidates, along time and across view.

In our previous publications on stereo-video inpainting [126, 128], we explain how the candidate set $\delta S_{x_i}$ can be defined as a union of a few sub-images, selected by using reconstructed motion and disparity vectors as guides to find the missing information. In this section we adopt a more general formulation which ties together our solution to the patch matching problem presented in section 6.3 and the search for candidates. Our main assumption is that the best candidates to solve equation 6.10 would lie along or nearby the trajectory of a missing pixel. This assumption allows us to perform the search for the best candidate in two steps. The first
Figure 6.7: Illustration of our patch tracking technique. In this example, a moving object is partially covered by the missing area on both views. After motion and disparity repair, the recovered vectors are used to estimate the trajectory of the missing pixel on both views. A matching score based on the patch-matching formula in equation [6.10] is estimated along the trajectory. The best match yielding the maximum score is selected, in this example it is on frame 5 on the right view. In other words, the frame $(5, R)$ corresponds to the best candidate to replace the unknown image data at the current missing site. Motion and disparity vectors are composed along time and across view using bilinear interpolation. This creates stereo-tracks for all missing pixels. The missing information is more likely to reside near the track. However, to consider possible drifting, we also use a refinement step at the best selected frame to align the recovered data more closely to the known surroundings of the missing area. The refinement step is described in section 6.4.2.
step consists in tracking the current patch on the trajectory of the current missing pixel along
time and across view as shown in figure 6.7. This tracking step allows to select the best candidate
in the frame \((f_T^i, v_T^i)\). The second step is a local optimisation procedure via gradient descent,
to align the patch in the best selected frame to the picture content in the current frame.

Computation of the trajectory is based on the reconstruction of motion and disparity vectors
inside the missing region, which is done as a preprocessing step explained in section 6.1.3. For
each target missing site, the disparity vector is used to align the point to its stereo-counterpart
and motion vectors are composed along time. We use bilinear interpolation for motion and
disparity composition to reduce drifting errors. We perform forward and backward tracking for
both left and right view streams as shown in figure 6.7.

Along the trajectory, the matching score is computed at each frame to find the best candidate
according to the right-hand side of equation 6.10. However, to speed up the computation, the first
two terms in the score are computed only on the most likely frames, given the prior information
collected in the patches \(S[N_{xi}]\) and \(R[N_{xi}]\). The whole score is then computed for candidates \(y\)
which satisfy:

\[
p(y | S[N_{xi}], R[N_{xi}], \Theta) > \theta_T \tag{6.31}
\]

In our experiments, we typically set \(\theta_T\) to a small value, e.g. \(\theta_T = 10^{-5}\), to avoid yielding
a sub-optimal solution. After this tracking step, we obtain a first estimate of the location of
the best candidate along the trajectory of the missing pixel. We note this candidate \(S_{x_i}^T = (x_T^i, y_T^i, f_T^i, v_T^i)\).

For tracking, our technique uses a simple first order integration with bilinear interpolation
of the local motion and disparity vectors. A more computationally intensive, but more accurate
tracking method for dense correspondences based on Optical Flow is presented in Crivelli et
al. [35,36]. Their method combines motion fields computed with different time intervals between
two frames to estimate the optimal path between two corresponding pixels. Optimal combination
of short-term and long-term matching is shown to reduce drifting and allows to deal efficiently
with temporary occlusions. We have not experimented with this scheme in our tests as it has
been published after the completion of our work.

In the next section, we present a local refinement step which uses previously computed
correspondences across a longer-term interval to mitigate drifting errors.

### 6.4.2 Local Refinement via Gradient Descent

Tracking from the reconstructed motion and disparity vectors may be inaccurate. More pre-
cisely, the tracking error or drifting increases with the temporal distance to the target missing
site. For robustness to drifting, we perform a local optimisation step to refine the spatial co-
ordinates \((x_T^i, y_T^i)\) of the candidate \(S_{x_i}^T\) obtained after patch tracking. During this step, we
seek to maximise the first factor in the right-hand side of equation 6.10 only, as the frame location
\((f_T^i, v_T^i)\) remains fixed. We can then focus on solving equation 6.7 to find the best match
with the lowest distance value to the target missing site. The final optimised selected candidate location is noted $\hat{S}_{x_i}$.

Firstly, we compute a possible estimated initialisation for the refined candidate location, noted $S^E_{x_i}$. We compute it as a weighted average of the locations of candidates previously selected from the frame $(f^T_i, v^T_i)$, to inpaint picture data around $x_i$. Given these locations in the data patch $S[N_{x_i}]$, we obtain:

$$S^E_{x_i} = \frac{m^0 S^0_{x_i} + m^+ S^+_{x_i} + m^- S^-_{x_i} + m' S'_T_{x_i}}{m^0 + m^+ + m^- + m'}$$  \hspace{1cm} (6.32)$$

The site locations $S^0_{x_i}$, $S^+_{x_i}$, $S^-_{x_i}$ and $S'_T_{x_i}$ are computed over the patches $S[N_{x_i}]$, $S[N^+_{x_i}]$, $S[N^-_{x_i}]$ and $S[N'_T_{x_i}]$ respectively. Each weight $m^0$, $m^+$, $m^-$ and $m'$ is equal to one if there is at least one valid point $S_{x_i+u} = (\cdot, \cdot, f^T_i, v^T_i)$ in the respective patch, and equal to zero otherwise. We use the reliability information in the patch $R[N_{x_i}]$ to lower the influence of points yielding a poor reliability. The computation is similar for each term. For instance, the estimated candidate $S^0_{x_i}$ is computed as follows:

$$S^0_{x_i} = \frac{\sum_{u|x_i+u \in N^+_{x_i}} w_u R_{x_i+u} \left[ S_{x_i+u} = (\cdot, \cdot, f^T_i, v^T_i) \right] S_{x_i+u}}{\sum_{u|x_i+u \in N^+_{x_i}} w_u R_{x_i+u}}$$ \hspace{1cm} (6.33)$$

To refine the selected position, we then use a weighted gradient descent process. Firstly, the current position is initialised as $S^0_{x_i}$. If the weights defined above are all null, $m^0 = m^+ = m^- = m' = 0$, the computation above is invalid and $S^0_{x_i} = S'_T_{x_i}$. Otherwise, we consider $S^E_{x_i}$ as a possible initialisation, and assign it to $S^0_{x_i}$ if it lowers the distance to the target missing site:

$$S^0_{x_i} = \begin{cases} S^E_{x_i} & \text{if } D \left[ N_{x_i}, N^E_{x_i} \right] < D \left[ N_{x_i}, N'_T_{x_i} \right], \\ S'_T_{x_i} & \text{otherwise.} \end{cases} \hspace{1cm} (6.34)$$

We then run $N$ iterations of the gradient descent algorithm. This procedure aligns the image data in a square patch around the refined candidate position to the picture data in a square patch around the missing site $x_i$. At the $n^{th}$ iteration, the refined candidate location is noted $S^n_{x_i}$. The algorithm only considers patches in the same frame as their centre, that is $I[N^0_{x_i}]$ around the target missing location, and $I[N^0_{S^T_{x_i}}]$ around the refined candidate position. Gradient descent is stopped in case a maximum number of iterations is reached, typically $N = 3$ in our tests, or if there is not enough available information around the refined candidate location. The later case can be expressed as follows:

$$\sum_{u|x_i+u \in N^0_{x_i}} w_u M^i_{x_i+u} M^0_{S^T_{x_i}+u} < \theta_R \sum_{u|x_i+u \in N^0_{x_i}} w_u M^i_{x_i+u}$$  \hspace{1cm} (6.35)$$

In our tests we set $\theta_R = 0.6$. This parameter controls when a low amount of known information around the candidate position would stop the gradient descent. If this happens, we set the estimated refined candidate location $\hat{S}_{x_i}$ to $S^0_{x_i}$ and trigger a fallback procedure detailed below.
After $N$ iterations we make sure that gradient descent actually lowers the distance to the target missing site and set the best value to be the estimated refined candidate location $\hat{S}_{x_i}$:

$$
\hat{S}_{x_i} \leftarrow \begin{cases} 
S^N_{x_i} & \text{if } D \left[ N_{x_i}, N_{S^N_{x_i}} \right] < D \left[ N_{x_i}, N_{S^0_{x_i}} \right], \\
S^0_{x_i} & \text{otherwise.}
\end{cases} 
$$

(6.36)

If the gradient descent procedure is stopped before $N$ iterations, if the final refined estimate $S^N_{x_i}$ does not lower the distance to the target missing site, or if it is not in the source, a fallback procedure is triggered. This procedure consists in searching for the best candidate via exhaustive search in a small square region around $\hat{S}_{x_i}$. This fallback mechanism is triggered when tracking ends up in a missing region, for instance when the missing pixel is never revealed in the stereo video, or when the gradient descent fails to converge to a suitable solution. We consider candidates $\hat{S}_{x_i} + \delta$ with $\delta = (\delta_x, \delta_y, 0, 0)$. The offsets $\delta_x$ and $\delta_y$ are typically in $\{-1, 0, 1\}$. However, if there are no source points in this search area, the offsets are increased gradually until there is at least one valid point. We then update the refined candidate if the distance to the target missing site is lowered:

$$
\hat{S}_{x_i} \leftarrow \hat{S}_{x_i} + \delta \text{ if } D \left[ N_{x_i}, N_{\hat{S}_{x_i} + \delta} \right] < D \left[ N_{x_i}, N_{\hat{S}_{x_i}} \right]
$$

(6.37)

### 6.4.3 Skip the Search via Coherent Patch Sewing

Our patch matching strategy is similar to the template matching idea from Shih et al. [144]. The missing data in the centre of a patch is inpainted, and the remaining surrounding information ensures that the structure of the stereo video is preserved in the reconstructed area. During candidate search, we explicitly seek points with more known surrounding data via the term defined in equation 6.24. Once the best estimated candidate $S_{x_i}$ has been found and refined, our goal is to exploit the available information around it by copying as much pixels as possible to the missing area surrounding $x_i$.

In Criminisi et al. [34], whole patches are copied directly to replace the missing image information, which can cause blocky artifacts in the reconstruction. In techniques such as Jia et al. [74], Graph Cuts is used to merge parts of a copied patch to the surrounding known data, which can increase the computational load. In Bornard et al. [18], a technique called coherence search [4] is used to allow whole blocks of known data to be copied from the source, by considering the direct neighbours of the selected candidate to replace corresponding pixels around the target missing site. This greatly reduces the burden of per-pixel replacement by skipping steps in the candidate search. We extend this technique to process a larger neighbourhood of pixels, with a more restrictive criterion for candidate selection, to avoid reconstruction errors.

First of all, the image information at $x_i$ is reconstructed using the image information at $S_{x_i}$, following the procedure detailed in section 6.5. Then our coherent patch sewing mechanism tries to inpaint missing data in a small neighbourhood around $x_i$, and in the same frame $(f_i, v_i)$. The aim of this mechanism is to skip the computationally intensive candidate search steps
6.4. Candidate Search using both Motion and Disparity

described previously in this section. Given offset values \( \delta = (\delta_x, \delta_y, 0, 0) \), each shifted known pixel at \( S_{x_i} + \delta \in N_{S_{x_i}}^\circ \cap S \) is considered to replace the corresponding shifted unknown pixel at \( x_i + \delta \in N_{x_i}^\circ \cap T_i \). Only missing pixels connected to at least two known pixels are considered, to avoid inpainting isolated points in the target area.

In our tests, if a valid shifted candidate is available in a small region around \( S_{x_i} \), we copy directly the value to the corresponding points within a small region around \( x_i \), for which \( (\delta_x, \delta_y) \in \{-1, 0, 1\}^2 \). For these points, we then have \( S_{x_i + \delta} \leftarrow S_{x_i} + \delta \). For larger offsets, i.e. \( (\delta_x, \delta_y) \in \{-5, \ldots, 5\}^2 \) and \( \delta_x^2 + \delta_y^2 > 2 \), we replace the image value at the missing site by the corresponding shifted candidate only if a distance-based condition is verified. More precisely, we consider the image data at \( S_{x_i} + \delta \) to be a good replacement for the unknown value at \( x_i + \delta \) if the distance to the shifted target missing site is below a threshold, defined in the following equation:

\[
S_{x_i + \delta} \leftarrow S_{x_i} + \delta \quad \text{if } \quad D \left[ N_{x_i + \delta}, N_{S_{x_i}} \right] \leq \frac{D \left[ N_{x_i}, N_{S_{x_i}} \right] + \mu_i^D (f_i, v_i)}{2} \tag{6.38}
\]

Shifted target missing points around \( x_i \) are processed by order of decreasing priority. In the equation above, the mean distance \( \mu_i^D (f_i, v_i) \) is an indication of the quality of the reconstruction so far, in the frame \( (f_i, v_i) \). It is computed as a weighted sum of the distances of previous matches found by candidate search to reconstruct the \( n_i \) previously missing pixels in the frame \( (f_i, v_i) \). The initial value \( \mu_0^D \) is set to zero for every frame. Subsequently, the values are updated as follows:

\[
\mu_{i+1}^D (f_i, v_i) \leftarrow \begin{cases} 
\gamma_i \mu_i^D (f_i, v_i) + (1 - \gamma_i) D \left[ N_{x_i}, N_{S_{x_i}} \right] & \text{if } x_i = (\ldots, f_i, v_i) \\
\mu_i^D (f_i, v_i) & \text{otherwise.}
\end{cases} \tag{6.39}
\]

with \( \gamma_i = \begin{cases} 
0 & \text{if } n_i \leq 1, \\
\frac{n_i - 1}{n_i} & \text{if } 2 \leq n_i < n_{\text{max}}, \\
\frac{n_{\text{max}} - 1}{n_{\text{max}}} & \text{otherwise.}
\end{cases} \tag{6.40}
\]

Where \( n_i \) is the number of pixels that have been through candidate search via patch tracking in frame \( (f_i, v_i) \) previously. In our tests we fix the parameter \( n_{\text{max}} = 20 \). The number is chosen heuristically because it provides a good dynamic adaptation of the value of the mean distance to the local properties of the region being reconstructed in the majority of our tests.

This thresholding scheme avoids overshooting by copying too much data at a time. The mean distance \( \mu_i^D (f_i, v_i) \) adapts to the statistics of the current area being reconstructed. The thresholding scheme also considers how good a replacement \( S_{x_i} \) is. If its distance value to the target missing site \( D \left[ N_{x_i}, N_{S_{x_i}} \right] \) is high, the mean distance lowers the threshold so that less pixels can be copied. On the contrary, if the distance value to the target missing site is low, then it is a good candidate and the threshold can be put higher by the mean distance to copy more pixels around it.
6.5 Image Reconstruction

Before replacing the missing image value, we correct its colour to avoid artifacts due to colour discrepancy across views, or in frames that are further apart in time. We also attenuate flicker artifacts via stereo-temporal blending. These mechanisms increase smoothness and coherence of the reconstruction along time and across views as a post-matching step. Finally, the order of filling or priority and the various variables used by our algorithm are updated.

6.5.1 Estimating the Colour Correction Coefficient

We have tested our technique on stereoscopic videos taken from the Sigmedia stereo-video database [31]. We have observed on raw data from this data set that colours between left and right views can differ significantly. This phenomenon, which can be seen in the leftmost column of figure 6.8, can be caused by differences in the balance setup of the cameras, for instance. Colour imbalance in a stereo-pair of frames can hamper binocular merging by the human brain and needs to be corrected. We address this issue during preprocessing. The central column in the figure shows the same stereo-pair of frames after colour balancing. The colour distribution on the right view has been corrected to match more closely the left view.

Coming back to the reconstruction of a target missing pixel $x_i$, consider the best candidate $S_{x_i}$. It can be selected on a frame which is far away in time, or in a different view, compared to the frame in which $x_i$ resides. There can be significant changes in illumination between these frames, due to the evolution of the conditions of capture at the time of shooting. There can also be a visible residual colour imbalance across views, even after the colour equalisation step discussed in the previous paragraph. Visible artifacts due to colour discrepancy could occur in the reconstructed area, if changes in illumination and residual colour imbalance are not accounted for. Some of these artifacts can be observed on the reconstruction in the rightmost column of figure 6.8.

To homogenise the reconstruction, we correct the colour of candidate pixels. More precisely, the intensity value at each candidate pixel in the source is adjusted before being copied at the corresponding missing location. The effect of our colour correction mechanism can be observed on an example in figure 6.8. Consider two pixel sites $x$ and $y$ in the video $V$. We use a one-tap linear predictive model to account for intensity changes:

$$I_x \approx a(x, y)I_y$$

(6.41)

The colour correction coefficient $a(x, y)$ is applied to each channel in the RGB colour space. We solve this system via a weighted least-squares estimator, over common available information in the neighbourhoods of each point. We identify image intensity as the grayscale image value. We have:

$$a(x, y) = \frac{\sum_{u|u+x \in \mathcal{N}_x} w_u I_{x+u} M_y^x + u M_y^i M_{x+u} I_{x+u}^G}{\sum_{u|u+y \in \mathcal{N}_y} w_u I_{y+u} M_y^x + u M_y^i [I_{y+u}^G]^2}$$

(6.42)
6.5. Image Reconstruction

Figure 6.8: Illustration of colour correction on the 20th frame of the rose_garden_tracking sequence. The leftmost column shows the original stereo-pair of frames before colour balancing. During preprocessing, in this example, we use Pitié and Kokaram's colour mapping [122] process, implemented as a plug-in for Nuke, to equalise the colour range across views, as shown in the central column. The missing region is also highlighted in blue. The rightmost column compares the output in case no additional colour correction is used during image reconstruction and in case our colour correction mechanism is used. Even after colour balancing during preprocessing, there is a visible colour mismatch if the data is copied directly from the source. This is due to variations in the lighting conditions along time as well as residual colour imbalance across views.

If saturation is reached in one colour channel, i.e. in case there is a channel \( c \in \{r, g, b\} \) for which \( a(x, y)I_y^c > 255 \), the correction value is set to the maximum value that does not cause any saturation.

6.5.2 Colour Correction and Stereo-Temporal Blending

If distortions are introduced in the video, we want this to happen in a smooth and coherent manner. To achieve this, we use a stereo-temporal filtering step. It consists in blending the best result obtained by candidate search \( I_{Sx} \) with the stereo-temporal data available in the direct vicinity of the missing pixel \( x \). The neighbouring sites around \( x \) are computed by using the
motion and disparity vectors, as explained in section 6.1.3. We obtain the following image value after colour correction and image blending:

$$\hat{I}_{Sx} = \frac{\beta^{o}a(x, S_{x})I_{Sx} + \beta^{+}M_{x}^{+}I_{x}^{+} + \beta^{-}M_{x}^{-}I_{x}^{-} + \beta'M_{x'}^{+}a(x', x')I_{x'}^{+} - \beta'M_{x'}^{-}I_{x'}^{-} + \beta'}{\beta^{o} + \beta^{+}M_{x}^{+} + \beta^{-}M_{x}^{-} + \beta'M_{x'}^{+} + \beta'}$$ (6.43)

The mask values $M^{i}$ allow to weight out the contribution of unknown pixels in the sum. We colour-correct the stereo-counterpart value at $x'$ in case there are remaining colour imbalances between views. However, we do not colour-correct temporal correspondences as illumination changes are unlikely to happen from one frame to the next. In our experiments, typical values for the blending parameters are $\beta^{o} = 1$ and $\beta^{+} = \beta^{-} = \beta' = 0.5$. These parameters control the strength of the filter. A higher value for $\beta^{+}$, $\beta^{-}$ or $\beta'$ means a smoother output but also introduces more blur.

### 6.5.3 Update Rules

After computing the corrected and filtered candidate image value $\hat{I}_{Sx}$, the unknown image data at the missing pixel $x_{i}$ can be filled. We also update the value of the mask, edge and reliability as follows:

$$I_{x_{i}} \leftarrow \hat{I}_{Sx_{i}}$$ (6.44)

$$M^{i}_{x_{i}} \leftarrow 1$$ (6.45)

$$E_{x_{i}} \leftarrow E_{Sx_{i}}$$ (6.46)

$$R_{x_{i}} \leftarrow 1 - D \left[ N_{x_{i}}, N_{Sx_{i}} \right]$$ (6.47)

The priority or order of filling is then updated for unknown pixels in the neighbourhood around $x_{i}$, which can be in the same frame $(f_{i}, v_{i})$ only, or also along time and across view, depending on the selected mode of completion. Our algorithm processes each pixel in the missing area in an iterative manner. However, there are several possibilities regarding the stereo-temporal progression of the completion. Firstly, one can order all pixels in the target region according to equation 6.9. However, it can necessitate more memory during processing because missing pixels can be selected anywhere in the video. Frame-by-frame processing has the advantage of allowing for previews of completed frames as the algorithm progresses. Therefore, the second completion mode we consider orders pixels in the target region using equation 6.9, but within the stereo-pair of frames at a fixed time instant. The remaining stereo-pairs of frames are then reconstructed sequentially along time.

In our experiments we choose to use a frame-by-frame mode of completion, similar to our previous publications about stereo-video inpainting [126–128], in which all frames are processed sequentially along time and across views. In this case, the priority is updated within the frame to be processed only. The algorithm starts with the first frame in the left view $(1, L)$. Once this frame is inpainted, the corresponding frame on the right view $(1, R)$ is processed. Then
the algorithm moves on to process the next frame along time, first on the same view \((2, R)\), then on the other view \((2, L)\). The remaining frames are processed accordingly, in the following order \((3, L), (3, R), (4, R), (4, L), \ldots, (F, L \text{ or } R)\). The last frame is either \((F, L)\) or \((F, R)\) depending on the number of frames in the sequence. This ordering mode propagates the reconstructed information from the starting frame to the end. It has the advantage of generating full completed frames early during the completion which can provide the user with feedback on how well the technique performs.

6.6 Summary of the Algorithm

1. Estimate motion and disparity in the missing region as explained in section 6.1.3.
2. Estimate the initial order of filling in the missing region as explained in section 6.3.1.
3. For each frame in the stereo video:
   (a) Select the missing pixel with the highest priority according to equation 6.9.
   (b) Compute the patch-matching formula described in sections 6.3.2, 6.3.3, 6.3.4 to search for the best candidate in the source for replacement along time and across view as explained in section 6.4.1. The best candidate is selected according to equation 6.10. Refine its position following section 6.4.2.
   (c) For the target missing pixel and data in its surroundings which are eligible for coherent patch sewing, as described in section 6.4.3, replace the missing image data after colour correction and stereo-temporal blending and update the priority and other variables as shown in section 6.5. The value of the inpainted pixel is computed according to equation 6.43.
   (d) Repeat steps 3a to 3c until all missing pixels are inpainted in the current frame.

6.7 Final Remarks

In this chapter, we have presented a new algorithm for stereo-video inpainting. Built around an exemplar-based framework, our technique includes dedicated mechanisms to enforce stereo-temporal consistency in the output. A sequence is first preprocessed to recover the motion and disparity vectors within the missing area. The target area, which can be in one or both views of a stereoscopic video, is then inpainted in a sequential greedy fashion.

The objective function used in previous exemplar-based techniques is extended in three ways. Firstly, we use a patch distance function based on the structural similarity instead of the sum of squared differences. This distance function puts the emphasis on matching image patches with a similar structure, via statistical estimates. Secondarily, we compute a term to help select the most informative candidate, based on the amount of newly revealed information in its vicinity.
This is particularly useful in case the missing data is never totally revealed. Thirdly, the history of the reconstruction is used to favour the selection of candidates close to points that have been previously selected. In previous works, this last point has been done implicitly or within global optimisation frameworks.

Searching for the best candidate in the known source region in a stereo video is a computationally intensive process. We reduce the computation burden in two ways. Firstly, we use reconstructed motion and disparity vectors to guide the search along time and across views. The best candidate is first found along the trajectory of a missing pixel, and then a local refinement step is used to rectify its position and correct tracking errors. Secondly, instead of per-pixel replacement, we use a coherence sewing mechanism that considers shifted candidate points, around the best selected candidate, to reconstruct corresponding shifted target missing points.

Image reconstruction then consists in replacing the unknown picture information at the target missing site with the known picture information at the selected best candidate. Direct replacement yields a spatially sharp reconstruction, but it can produce temporal artifacts such as flickering. It can also exhibit colour mismatching, if there is some colour imbalance across views. We modify the pixel replacement procedure in two ways. Firstly, colour information is corrected to blend it with the surrounding information in the current frame being reconstructed. Secondly, a low-pass filter is used to blend the image information at the selected candidate with available picture data across views and along time. This stereo-temporal information is fetched with the motion and disparity vectors in the next and previous frames, as well as in the other view. These mechanisms increase the temporal smoothness of the output, as well as its view consistency.

In the next chapter, we present experimental results using our stereo-video inpainting technique on various challenging stereoscopic videos. We provide comparison with state-of-the-art video inpainting techniques, and we discuss the shortcomings and benefits of our approach.
This chapter presents experimental results on our stereo-video inpainting technique. We test our technique on various videos taken from the Sigmedia stereo-video database presented in Corrigan et al. \cite{Corrigan2010}. These videos exhibit complex camera motion, can contain moving objects and various artifacts. The area to be inpainted can be in one view only or in both views.

We compare our method to existing techniques from the literature. Particularly, we show results from the Nuke implementation of the rig removal technique from Kokaram et al. \cite{Kokaram2008}. This technique is based on accurate recovery of the motion field in the missing area within a Bayesian framework. We also show results from the state-of-the-art video inpainting technique presented recently by Newson et al. \cite{Newson2010, Newson2011}. This technique uses a modified PatchMatch \cite{Baker2011} algorithm to assemble fragments of video cubes within a global optimisation framework.

How to evaluate and compare the results of inpainting techniques is an open problem. In most real-world applications, such as rig or artifact removal, there is no ground-truth data available to assess the reconstruction. In this case, we perform a qualitative analysis to compare our technique to the two state-of-the-art methods mentioned above. We focus our attention on spatiotemporal coherence of the reconstruction, and on preservation of view consistency.

In addition to object-removal tests, we design some tests with an artificial degradation to be repaired. The aim of these tests is to enable quantitative comparison of the output of our technique and other methods to a reference video. In these tests, the missing information is predominantly static so that it can be recovered accurately in other frames along the sequence. The stereo videos we use for our tests are summarised in figure 7.1.
<table>
<thead>
<tr>
<th>Sequence name</th>
<th>Duration (in frames)</th>
<th>Frame size (in pixels)</th>
<th>Total number of missing pixels</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rose_garden</td>
<td>51</td>
<td>720x378</td>
<td>$\sim 8.7 \times 10^4$</td>
<td>The mask is located on the back of a walking woman. The twist in the strap is occluded during the first 10 frames.</td>
</tr>
<tr>
<td>red_barrier</td>
<td>71</td>
<td>960x720</td>
<td>$\sim 4.6 \times 10^6$</td>
<td>The mask is static. It occludes parts of the red barrier and the wall. The video contains complex camera motion, parallax, and changes of scale.</td>
</tr>
<tr>
<td>water_drop</td>
<td>150</td>
<td>960x540</td>
<td>$\sim 1.6 \times 10^6$</td>
<td>The mask is located around a water drop that falls on left camera of the stereo rig, at the fifth frame. The video contains non-planar camera motion and changes of scale.</td>
</tr>
<tr>
<td>walking_man</td>
<td>100</td>
<td>960x540</td>
<td>$\sim 4.5 \times 10^6$</td>
<td>The mask is located around a walking man and his reflection. The camera motion is predominantly static, with some jitter. A moving person at the very right of the scene is occluded during a few frames.</td>
</tr>
<tr>
<td>walking_lady</td>
<td>100</td>
<td>960x720</td>
<td>$\sim 7.4 \times 10^6$</td>
<td>The mask is drawn around a walking lady, on both views. The camera motion is predominantly static and the lady is moving slowly. She crosses a couple walking in the opposite direction.</td>
</tr>
<tr>
<td>green_screen</td>
<td>100</td>
<td>960x540</td>
<td>$\sim 4.0 \times 10^5$</td>
<td>The mask is located on the head of the talking actor, on both views, from frame 70 to frame 90.</td>
</tr>
<tr>
<td>pub_talk</td>
<td>100</td>
<td>960x540</td>
<td>$\sim 2.5 \times 10^5$</td>
<td>The mask is located on the head of the talking man, on both views, from frame 32 to frame 60.</td>
</tr>
</tbody>
</table>

Figure 7.1: Summary of the stereo videos, extracted from the Sigmedia database [31], that we use to perform inpainting experiments. The aim is to fill in the missing data in the masked area, highlighted in blue. In the first two sequences from the top, the mask indicates an artificial degradation, so that ground-truth data is available for objective comparison. In the next three tests, a video object has to be removed from the scene, and no ground-truth data is available. In the last two sequences, the face of a person is occluded for a few frames by the mask. In this case, the reconstruction should not be expected to match the ground-truth video exactly.
7.1 Our Experimental Setup

To help the reader understand the context in which we have performed our tests, Figure 7.1 shows a summary of the stereo videos on which we test our technique. This section presents the database, the inpainting experiments, the evaluation methods, and the other techniques we use for comparison as well as our testing environment. In section 7.2, we then present a comparison to state-of-the-art techniques and discuss the results of our tests.

7.1.1 A Small Database of Stereoscopic Videos

Generic stereo-video inpainting has not been studied in the literature previously, so there is no available database of stereoscopic videos to test our technique. We have used the Sigmedia stereo-video database [31] to create inpainting tests. Our tests consists of 7 stereo videos capturing various indoor and outdoor situations. We choose videos with a varying degree of difficulty due to the size of the missing area and the motion content in the shot. A summary of the short clips extracted from the original sequences is shown in figure 7.1.

Along with the video data, we create binary masks indicating the area to be inpainted. Except for the walking man and walking lady sequences, all masks are created with semi-automatic rotoscoping tools available in Nuke, a popular compositing software for movie post-production. The position and shape of the masks are adjusted manually on key frames, and in-between values of control points are interpolated automatically along time. Moreover, Ocula, which is a set of Nuke plug-ins for stereo-video processing, allows to project the masks from one view to the other automatically, using disparity information. The masks are illustrated as blue highlights in the third column from the left in figure 7.1. All masks are defined on every frame except for the water drop, green screen and pub talk sequences.

To create the masks for the walking man and walking lady sequences, we have experimented with automatic rough matte propagation using our motion-based sparse stereo-video segmentation technique presented in chapter 3. Firstly, the scene is segmented automatically. Then, the segmentation output is refined to isolate a target video object to be removed from the sequence. These first two steps are performed according to chapter 3. After the segmentation steps, affine motion models are computed from frame to frame using tracks belonging to the selected target object. Finally, the motion models are used to automatically propagate a bounding box drawn by the user on one key frame. In practice, propagation from several key frames may be necessary to cover the object fully along the video.

As mentioned in section 6.5.1, colour balancing is a necessary preprocessing step to our inpainting technique. Figure 7.2 illustrates how the colour matcher tool in Ocula can be used to equalise the colour between views in two examples. The videos shown in figure 7.1 are all preprocessed with such a technique. The next section details the inpainting tests we perform on

---

1 version 6.3 - https://www.thefoundry.co.uk/products/nuke-product-family/
2 version 3.0 - https://www.thefoundry.co.uk/products/ocula/
Figure 7.2: Illustration of the colour equalisation step using Ocula. In the leftmost example, the left view is considered too bright. Colour transfer is done from the right view to the left view. In the rightmost example, the right view is considered too dark. Colour transfer is done from the left view to the right view. The output stereo pairs have a more balanced colour distribution.

7.1.2 Overview of the Inpainting Tests

For the rose garden and red barrier sequences presented in figure 7.1, an artificial degradation has to be removed from the shot. In these experiments, the aim is to reconstruct the missing data within the masked area as closely as possible to the original ground-truth sequence. We have designed these tests to be able to quantify the performance of our technique and compare it to other methods. We employ the objective image quality metrics described in the next section to compare the output of our tests to the reference sequence. However, bear in mind that quantitative evaluation must always be confirmed by a qualitative analysis.

For the water drop, walking man and walking lady sequences presented in figure 7.1, an unwanted object has to be removed from the scene. For these tests, the quality of the output has to be assessed subjectively as no ground-truth data is available. In the walking man and walking lady sequences, the masks are located around people walking in different environments. The output then consists of the background of the scene without the selected person. In these cases, we seek to create a clean plate sequence from a video shot. In the water drop sequence, a rain drop falls on the left camera of a stereo rig, corrupting the left view of the shot. We draw a mask around the corrupted image area and the output consists of the restored view. In this case, the aim is to remove an artifact on the sequence. This example illustrates a video restoration application for inpainting.

Finally, for the green screen and pub talk sequences presented in figure 7.1 the mask covers the face of a person in the scene for a limited number of frames. The missing data can
contain unique facial expressions that are not repeated elsewhere in the scene. An acceptable
reconstruction should be close to the original sequence in terms of quality metric, but it is
unlikely that it would be identical. Therefore, analysis of the results of these tests can use a
mixture of objective and subjective arguments.

7.1.3 Quality Metrics for Stereoscopic Videos

Objective quality assessment for monoscopic videos is a challenging topic, which has been studied
extensively in the literature. The latest developments make use of motion compensation to take
into account temporal evolution of videos \[137,169\]. Simpler methods use statistics from image-
based quality scores computed at every frame. We opted for the second approach by studying
the temporal variations of the \textit{SSIM} \[170\] score computed at each frame between the output of
our tests and a reference sequence. Moreover, there are two video streams in stereo videos, and
they should be analysed together. This can be done by observing the evolution of the scores on
each view separately. We also use an extension of the \textit{SSIM} measure to stereo images \[27\] for a
joint treatment of both views. In section 7.2, the computed scores are displayed in graphs and
summarised by standard box plots.

First of all, we employ the original \textit{SSIM} \[170\] measure as a reference to assess the quality
of the output in our video reconstruction tests. We use the \texttt{Matlab} implementation provided
by the authors. For each frame, on each view separately, we compute \textit{SSIM} indexes between
the inpainted result and the original frame. The indexes are computed over image blocks of
size $11 \times 11$ pixels. In the original implementation, the final score to compare two frames is
computed as the average value over all blocks. However, as differences between the original
image and the inpainted version arise exclusively in the reconstructed area, we only use the
indexes on blocks in the masked region. The final quality score we compute between two images
is thus the mean of the \textit{SSIM} indexes in the reconstructed area, which we note MSSIM in the
following sections.

We also employ the adaptation of the \textit{SSIM} measure to a stereo-pair of frames presented
by Chen et al. \[27\]. This technique has been developed to assess the quality of stereoscopic
images that have been affected by possibly asymmetric distortions. The technique consists in
computing the quality score over an intermediate image. The intermediate image serves as an
approximation of the cyclopean view formed within an observer’s mind when viewing a stereo-
pair of images. It is synthesised as a weighted combination of the left view and the disparity-
compensated right view. The weights take into account binocular rivalry, which depends on
visual stimulus strength. The weights are then computed from the normalised magnitudes of
Gabor filter bank responses. We use the \texttt{Matlab} implementation provided by the authors. We
modify the technique to use the \textit{SSIM} measure in the masked area in the intermediate image as

\[3\]https://ece.uwaterloo.ca/~z70wang/research/ssim/
\[4\]http://live.ece.utexas.edu/research/quality/
explained in the previous paragraph. The masked area is computed as the union of the mask on the left view and the disparity-compensated mask from the right view. The mean of the indexes in the masked area is denoted as $\text{MSSIM}_3D$ in the following sections.

Chen et al. [27] evaluate the relevance of their quality metric for stereo images on a series of directed human studies. These tests estimate how well the metric is correlated to human perception. The results of these tests show that, in general, the perceived quality of stereoscopic images cannot be accurately characterised by simple averaging of 2D quality measures across the left and right images. In the case of asymmetric distortions in particular, the authors show that their technique yields a significantly higher correlation with the human subjective judgements. A possible cause for this discrepancy is binocular rivalry, which is a perceptual effect that occurs when the two eyes view mismatched images at the same retinal locations. Binocular rivalry triggers a sense of failed stereo-fusion or an alternation of predominance of the left and right images. Stimuli that are higher in contrast, or have more visible contours tend to dominate the rivalry. Such stimuli seem to give a befitting description to the type of artifacts often observed in inpainted images, and which are usually caused by broken object structures and visible seams. This remark provides an additional motivation for the use of this technique to evaluate our results.

### 7.1.4 Implementation and Testing Environment

We compare our method to two state-of-the-art techniques. Firstly, we use the rig removal method from Kokaram et al. [88]. This technique is based on accurate motion field reconstruction within a Bayesian framework. Using the reconstructed motion, available data is pulled from other locations in the video to fill in the missing area. Secondarily, we employ the more recent patch-based video inpainting technique presented by Newson et al. [111,112]. This technique is based on a patch-matching approach for constrained texture synthesis. Temporal smoothness constrains are enforced within a global optimisation framework. Our technique can be seen as a hybrid between these two methods as it uses motion and disparity as guides for a constrained texture synthesis method, within an iterative exemplar-based framework.

For the method of Kokaram et al. [88], we run our tests with the implementation of that technique available in Nuke. For the method from Newson et al. [111,112], the inpainted videos have been kindly provided by the authors. We test our technique on a Matlab implementation of the stereo-video inpainting algorithm described in chapter 6. Our code uses Olivier Woodford’s mexed implementation of the 2D interpolation function for images, which is faster than the native Matlab function. The machines on which we run our experiments are PCs containing 16GB of RAM and eight Intel i7 CPUs operating at 3.4GHz. The amount of RAM needed to process one video varies from one to six gigabytes in our tests.

\[http://www.robots.ox.ac.uk/~ojw/software.htm\]
7.2 Analysis of the Results

In this section, we compare the results of our stereo-video inpainting method to the two state-of-the-art video inpainting techniques mentioned in section 7.1.4. Whereas our method processes both views in a stereo videos jointly, the reference video inpainting techniques are applied to each view separately. We present results for each sequence described in figure 7.1. Three stereo-frames from each sequence are shown to illustrate our results. The frames are cropped around the reconstructed region for readability reasons. When available, quantitative evaluation with the metrics detailed in section 7.1.3 is also presented. To validate our observations, the reader can watch the videos online at http://www.sigmedia.tv/Misc/ResultsThesisFelix.

7.2.1 Results on the Rose_garden Sequence

The rose_garden sequence is presented in figure 7.3. The missing data is situated in the back of the walking woman. The mask hides a twist in the strap during the first ten frames before drifting and revealing it. This test provides a simple example where motion and disparity information can be recovered easily within the masked area. Providing that motion information is recovered accurately, it is possible to pull the picture information from other frames to recover the original data almost perfectly.

Figure 7.3 shows the results of our method and the other two state-of-the art techniques. For all methods tested, no major breakdown can be observed in this example. Some noticeable errors can be seen for the method of Newson et al. [111, 112]. Particularly, the structure of the twist is broken in frame 6, the colour of the strap is slightly too dark on frame 27, and the dark coat area is too smooth on frame 45. However, the evolution of the output of Newson et al. [111,112] is temporally smooth.

Figure 7.4 shows the evolution of the MSSIM scores on this sequence, for the three techniques analysed in this study. Overall, Kokaram et al. [88] achieves the best results in this example, closely followed by our technique. The output of the technique presented by Newson et al. [111, 112] yields a slightly lower score, especially in the beginning of the sequence, while the mask is on the twist in the strap. The evolution of the MSSIM3D score is presented in figure 7.5 and it shows the same pattern.

7.2.2 Results on the Red_barrier Sequence

The red_barrier sequence is presented in figure 7.6. The missing data is situated on a complex background containing a red barrier, a stone wall and a further plane behind an opening in the wall. The mask is static and the missing data is revealed progressively along time as the camera moves. The camera motion is complex and contains some jitter. The mask is rather large and crosses various depth layers, which is challenging to motion and disparity reconstruction approaches.
Figure 7.3: Inpainting results on three frames from the rose garden sequence. The mask indicating the region to fill in is shown as a blue highlight in the first row from the top. Comparison between our technique and the two state-of-the-art methods presented in section 7.1.4 is displayed in the next three rows.

Figure 7.4: Evolution of the MSSIM score for the rose garden sequence. The temporal evolution of the score on each view is shown on the left, and the box plot representation of the scores on both views is shown on the right. The method from Kokaram et al. [88] yields the best results on this example.
7.2. Analysis of the Results

Figure 7.5: Evolution of the MSSIM3D score for the rose_garden sequence. The temporal evolution of the score is shown on the left, and the box plot representation is shown on the right. The method of Kokaram et al. [88] yields the best results on this example.

Figure 7.6 shows the results of our method and the other two state-of-the art techniques. All methods achieve a similar reconstruction in this example. The texture of the wall is well reconstructed by all techniques, as can be seen on frame 6. However, some noticeable artifacts can be observed on frames 24 and 50.

In frame 24, observe the top of the barrier and the far plane behind it, showing the wheel of a car and parts of a window. In the outputs of all three technique, a slight misalignment can be seen on the top of the barrier. Whereas our technique and Newson et al. [111, 112] preserve the sharpness of this object, a blending operation occurs in the result from Kokaram et al. [88]. The motion due to parallax causes the reconstruction in the far plane to be slightly inaccurate for Kokaram et al. [88] and Newson et al. [111, 112], whereas our technique achieves the best reconstruction of that element of the image. This is probably due to the fact that no stereo information is exploited by the other two methods.

In frame 50, observe the distortion of the top right of the reconstructed barrier. Whereas our method generates a consistent error on both views, there is a discrepancy between the left and right views in the output from the other two techniques. This shows the necessity of a joint processing of both views.

Figure 7.7 shows the evolution of the MSSIM scores on this sequence, for the three techniques compared in this study. Overall, our technique achieves the best results in this example, followed by Kokaram et al. [88] and Newson et al. [111, 112]. The evolution of the MSSIM3D score is presented in figure 7.8 and it shows a similar pattern. As can be seen around frame 24 on the graphs, when there is a discrepancy between the MSSIM scores across views, the MSSIM3D score seems to emphasise that difference.
Figure 7.6: Inpainting results on three frames from the red barrier sequence. The mask indicating the region to fill in is shown as a blue highlight in the first row from the top. Comparison between our technique and the two state-of-the-art methods presented in section 7.1.4 is displayed in the next three rows.

Figure 7.7: Evolution of the MSSIM score for the red barrier sequence. The temporal evolution of the score on each view is shown on the left, and the box plot representation of the scores on both views is shown on the right. Our method yields the best results on this example.
7.2. Analysis of the Results

7.2.3 Results on the Water\_drop Sequence

The water\_drop sequence is presented in figure 7.9. In this example, no ground-truth data is available. The task is to remove a water drop that falls on the left camera of a stereo rig. The missing data is situated on a complex background, containing a group of statues, a water jet, and buildings further away. The mask is static, located on the water drop, which falls on the fifth frame. As the artifact is only situated on the left view, the missing data can be retrieved by our technique across views, whereas the other two video inpainting techniques studied here can only rely on data revealed due to camera motion along time on the left view.

Figure 7.9 shows the results of our method and the other two state-of-the-art techniques. As the reconstructed area is situated on the left view only, we show the uncorrupted right view on the first row and the next rows show the mask and the reconstructed results on the left view. There is no available ground truth on this sequence, however, a qualitative analysis shows that our method outperforms the other two techniques on this example.

By observing the third row in figure 7.9 it can be seen that our technique is able to reconstruct the missing area accurately. The most noticeable error is a slight difference in colour around the legs of the upper statue. The output from Kokaram et al. shows noticeable spatial distortions, which become worse as time progresses, as can be seen on frames 80 and 140. We infer that this is due to the long temporal extent of the missing region, which causes the motion repair mechanism to fail. The result from Newson et al. yields an erroneous reconstruction which is temporally consistent, or evolving smoothly. It seems that the texture from the building on the left has been found to match accurately the surroundings of the missing region by this method. This algorithm seems to be trapped in a local minimum in this case.

The use of motion information helps our technique, as the correct texture is available in the
Figure 7.9: Inpainting results on three frames from the *water drop* sequence. The first row from the top shows the uncorrupted right view. The mask around the water drop on the left view is shown as a blue highlight in the second row. The aim of the experiment is to remove this artifact. Comparison between our technique and the two state-of-the-art methods presented in section 7.1.4 is displayed in the next three rows.

beginning of the sequence.

Figure 7.10 shows an anaglyph representation of the inpainting results detailed above, on frame 80. By using red/cyan glasses, it can be seen that a comfortable sensation of depth is rendered in our result, whereas visual discomfort due to inconsistencies across views is noticeable in the outputs of the other two techniques. The mismatch creates binocular rivalry. This example shows that our algorithm benefits from the information at the right view to maintain a coherent reconstruction during the sequence, whereas the other two methods show a degradation over
7.2. Analysis of the Results

Figure 7.10: Anaglyph representation of the inpainting results on frame 80 from the water drop sequence, as presented in figure 7.9. We invite the reader to use standard red/cyan glasses to view the 3D effect on the results. It can be observed that our technique preserves the stereo-consistency whereas viewing the outputs from the other two techniques provokes discomfort.

time and spatial and stereo inconsistencies.

7.2.4 Results on the Walking man Sequence

The walking man sequence is presented in figure 7.11. In this sequence, the task is to reconstruct the background in place of the walking man. The mask is obtained in a semi-automatic manner by propagating a rough matte drawn by the user on key frames. After user-assisted sparse stereo video segmentation, using the method described in chapter 3, we compute affine motion models for the cluster corresponding to the walking man. We draw a rectangular shape around the man and his reflection on both views, in one frame. This shape is then warped from frame to frame to generate the final mask.

Figure 7.11 shows the results of our method and the other two state-of-the art techniques. The background in this sequence is rather simple, except for a moving person, which is occluded by the walking man towards the end of the shot. On the first two frames in the figure, it can be seen that all three techniques achieve a similar quality of reconstruction. On the left view of frame 18, one can notice a slight misalignment at the top of the stone bench on the result of
Figure 7.11: Inpainting results on three frames from the *walking man* sequence. The mask indicating the region to fill in is shown as a blue highlight in the first row from the top. Comparison between our technique and the two state-of-the-art methods presented in section 7.1.4 is displayed in the next three rows.

Figure 7.12: Detail of our inpainting results on frame 18 from the *walking man* sequence. Without colour correction, noticeable artifacts are visible in the reconstruction, they are particularly visible in the right view in this example.
Newson et al. [111, 112]. On frame 49, on the results of Kokaram et al. [88], one can notice a slight colour imbalance producing a visible seam at the boundary of the reconstructed region, near the edge of the library wall.

However, a significantly different behaviour can be observed on the left view of frame 79, where there is an occluded moving person in the background. Blending can be noticed in the results of Kokaram et al. [88], as the local motion in the missing region is not recovered accurately enough. The person is missing in the output from Newson et al. [111, 112]. On the other hand, our method yields a sharp reconstruction of the person, even though motion reconstruction is inaccurate. This illustrates the benefits of combining an exemplar-based technique with motion reconstruction. By viewing the video, one can see that there is a visible temporal transition when the person is revealed again, in our results, but also in the outputs of the other two methods. This shows that reconstructing a background region containing moving objects is still a challenge for inpainting methods.

There is a residual colour imbalance across views in this sequence, even after the colour equalisation step we perform during preprocessing. Unlike video inpainting methods utilising one view at a time, our technique combines data from both views to reconstruct the missing region. We illustrate in figure 7.12 the contribution of our colour correction mechanism presented in the previous chapter, in section 6.5.1. Without it, artifacts in the reconstructed area due to the colour imbalance across views are much more noticeable.

### 7.2.5 Results on the Walking lady Sequence

The walking lady sequence is presented in figure 7.13. In this sequence, we want to reconstruct the background in place of the walking lady. The mask is obtained in a semi-automatic manner, using the same procedure as for the walking man sequence. Figure 7.13 shows the results of our method and the other two state-of-the art techniques. There is almost no camera motion in this sequence, but the mask is rather large. The main challenge is caused by the couple walking from right to left, and which is at least partially covered by the mask during almost 40 frames. All three methods fail to output a correct reconstruction in this example.

The output from the technique by Newson et al. [111, 112] contains errors in the reconstructed static background. Notably, the base of the black post is missing on the left view in frame 20, and the white sign is missing in the right view in frame 95. In frame 67, it can be seen that the couple is partially reconstructed in the right view, whereas it is absent on the left view. We infer that the algorithm used by this technique converges to an incorrect solution because of the size of the mask and the amplitude of the motion of the occluded couple.

The result from Kokaram et al. [88] shows several problems. In frames 20 and 95, a colour discrepancy can be observed between the reconstructed area and its surroundings. In frame 95, the region marked in red cannot be reconstructed by the technique. This happens when the missing region is never revealed in the shot. It can also happen if the missing data is revealed
only for a few frames which are temporally far. The former case shows the limit of a purely motion-based approach. The latter could be due to inaccuracies in the reconstructed motion field, or due to the use of only a subset of source frames for reconstruction. Blending artifacts can also be observed on frames 20 and 67, due to an inaccurate reconstruction of the motion of the couple.

Our method yields a correct reconstruction in the beginning of the shot as shown in frame 20, and in the end, as shown in frame 95. As we use an exemplar-based framework, a coherent reconstruction is attained in areas that are hardly ever revealed, and where a purely motion-based approach such as Kokaram et al. [88] can fail, as can be seen on frame 95. However, the motion of the couple cannot be reconstructed accurately by our method, and although a sharp and stereoscopically coherent reconstruction is generated at frame 67, it contains artifacts. Viewing the video shows that the walking motion of the couple is not reconstructed properly at all.

This example shows that reconstructing a background region containing moving objects is still a challenge for inpainting methods, even more so when there is a high motion amplitude.
7.2. Analysis of the Results

Figure 7.14: Anaglyph representation of the inpainting results on frame 67 from the walking_lady sequence, as presented in figure 7.13. We invite the reader to use standard red/cyan glasses to view the 3D effect on the results. It can be observed that our technique preserves the stereo-consistency in the reconstructed area, even if this reconstruction is erroneous. The output from Kokaram et al. [88] is also consistent across views in this example. However, the output from Newson et al. [111, 112] cannot be exploited for stereo imagery.

and if the mask has a large extent in time and space. However, as we are interested in the reconstruction of stereoscopic videos, we need to ensure that when our technique fails it does so in a coherent manner across views. Figure 7.14 shows an anaglyph representation of the inpainting results detailed above, on frame 67. By using red/cyan glasses, it can be seen that our technique preserves the stereo-consistency in the reconstructed area, although this reconstruction is erroneous. The output from Kokaram et al. [88] is also consistent across views in this example. However, the output from Newson et al. [111, 112] cannot be exploited for stereo imagery due to significant binocular rivalry.
Figure 7.15: Inpainting results on three frames from the green_screen sequence. The mask indicating the region to reconstruct is shown as a blue highlight in the first row from the top. The original sequence is displayed on the second row. Comparison between our technique and the Nuke implementation of the rig removal method presented by Kokaram et al. is shown in the next two rows.

7.2.6 Results on the Green_screen and Pub_talk Sequences

We analyse the results of the tests on the green_screen and pub_talk sequences in the same section, as both videos present similar challenges. The green_screen sequence is presented in figure 7.15 and the pub_talk sequence is presented in figure 7.17. In both cases, the task is to recover the missing data in a masked area situated on the face of a person. In each video, the camera is static and the head of the person is moving only slightly. Some unique facial expressions may occur during the time the mask covers the face, so that the reconstruction should not be expected to match the ground-truth video exactly. For these tests we only compare our method to the technique by Kokaram et al. as we do not have the data for the other technique.

Results on the green_screen sequence can be seen in figure 7.15. Our method yields a plausible reconstruction in this case, although many noticeable differences can be observed compared to the original sequence. Particularly, the subtle motion of the lips and the eyes of the talking man are not recovered by our method. However, the reconstructed area is consistent with its surroundings and preserves the details of the texture in the original sequence. In the
7.2. Analysis of the Results

Figure 7.16: Evolution of the MSSIM3D score for the green screen sequence. The temporal evolution of the score is shown on the left, and the box plot representation is shown on the right.

output from Kokaram et al. [88], it seems that a mixture between two images is used to fill in the missing region. We infer that it is made of data fetched in the video before the mask appears and after it disappears. The discrepancy between backward and forward reconstruction is visible here as the motion of the facial features of the man is not recovered accurately enough. Quantitative evaluation of the results with the MSSIM3D score is shown in figure 7.16. Our method yields a higher score overall. However, towards the last masked frame, the transition to the actual data in the sequence seems to occur faster in the output from Kokaram et al. [88], as this technique yields a better score.

Results on the pub talk sequence can be seen in figure 7.17. The outputs of both methods for frame 35 are similar. This frame is close to the first masked frame and uncovered data before this frame can be used for reconstruction. In frame 38, our method is better at preserving the texture of the original sequence whereas in the output from Kokaram et al. [88], some artifacts are visible due to blending. Observe also that the talking man is blinking on the original sequence, but this detail is impossible to recover accurately in this case. Between frame 38 and frame 48, the man turns his head. In frame 48, our technique yields a distorted result as the aspect of certain facial features of the man is propagated from previous frames, whereas others follow more closely the motion of the head. However, the output from Kokaram et al. [88] shows a better adaptation to the motion of the head, although severe blending artifacts can be noticed. Observing the MSSIM3D graph on figure 7.16 it can be seen that after frame 53, the score obtained by the technique of Kokaram et al. [88] becomes higher than the score obtained by our technique. This confirms that a better transition is achieved by this technique between the inpainted data and the original data.

Overall, in both examples, the reconstructed texture with our method is accurate at each frames, but too static when watching the video. This is partly due to our frame-by-frame
Figure 7.17: Inpainting results on three frames from the pub_talk sequence. The mask indicating the region to reconstruct is shown as a blue highlight in the first row from the top. The original sequence is displayed on the second row. Comparison between our technique and the Nuke implementation of the rig removal method presented by Kokaram et al. \cite{88} is shown in the next two rows.

Figure 7.18: Evolution of the MSSIM3D score for the pub_talk sequence. The temporal evolution of the score is shown on the left, and the box plot representation is shown on the right.
7.3. Discussion

Table 7.1: Comparison of the processing times of the tests studied in this section.

<table>
<thead>
<tr>
<th>Method</th>
<th>rose_garden</th>
<th>red_barrier</th>
<th>water_drop</th>
<th>walking_man</th>
<th>walking_lady</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our method</td>
<td>1 hr 28 min</td>
<td>66 hr 50 min</td>
<td>26 hr 50 min</td>
<td>78 hr 30 min</td>
<td>138 hr</td>
</tr>
<tr>
<td>Kokaram et al. [88]</td>
<td>10 min</td>
<td>40 min</td>
<td>45 min</td>
<td>40 min</td>
<td>1 hr 10 min</td>
</tr>
<tr>
<td>Newson et al. [111, 112]</td>
<td>1 hr 7 min</td>
<td>6 hr 18 min</td>
<td>7 hr</td>
<td>2 hr 50 min</td>
<td>12 hr</td>
</tr>
</tbody>
</table>

texture synthesis approach for reconstruction. The approach from Kokaram et al. [88] struggles to reconstruct accurately the motion field within the missing area and a smooth solution is obtained by blending the data from several locations in the video. However, with our method, the choice of selecting the best candidate allows for a better preservation of the original texture in the inpainted region.

7.3 Discussion

Processing Speed: Table 7.1 shows the processing time of our technique compared to the other two state-of-the-art techniques presented in section 7.1.4, for the first five tests detailed previously in this chapter. Total time is obtained for the other two methods by adding the processing time for each view. As can be seen, the Nuke implementation of the method from Kokaram et al. [88] is the fastest. This is not surprising as the implementation has been optimised for performance in Nuke, because it is designed for professional use in post-production houses. The method from Newson et al. [111, 112] is up to one order of magnitude faster than our method. One of the goals of Newson et al. [111] is to speed up the algorithm presented by Wexler et al. [172], the success of this approach is highlighted in our comparison. Conversely, speed has not been a major focus during the design of our algorithm. In our tests, the use of coherent patch sewing as described in section 6.4.3 yields a speed-up factor of three on average. However, our technique is still very slow when compared to the other two methods. The time to process a video with our method can vary from a few hours to a few days depending on the number of pixels in the missing region, as can be seen in table 7.1. Processing time could potentially be greatly reduced by porting our Matlab prototype to C++ for instance.

Candidate Selection: Reduction of the number of candidates for comparison during the search of the source is one way of increasing the speed of our technique. It can be done by increasing the threshold in equation 6.31. However, one must be cautious not to increase the value too much as this could yield a suboptimal solution if the best candidate is not included in the candidate set. More generally, we assume that constraining the search, to favour selecting neighbouring candidates to fill in neighbouring missing pixels helps improving the coherence of the reconstruction. This is the reason for the bias used during patch matching as detailed in section 6.3.4.
Stereo-Temporal Smoothness: In practice, imposing a bias during candidate selection alone could still result in temporal artifacts such as flickering in our results. This is also partly due to the frame-based constrained texture synthesis algorithm which is at the core of our technique: optimisation of candidate locations as well as reconstruction of missing pixels are performed within one frame at a time. This means that the reconstruction is optimised locally for each frame and variations in alignment from one frame to the next can still occur. The addition of the stereo-temporal blending scheme presented in section 6.5.2 greatly increases temporal smoothness in our results. This filtering step ensures that the solution from the previous frame is propagated to the next frame, while incorporating progressively new information found via patch tracking and matching.

Pixel and Frame Ordering: We use an iterative approach to solve the inpainting problem. The pixel ordering within each frame follows section 6.3.1. However, we have considered several possibilities regarding the ordering of reconstruction of each frame as explained in section 6.5.3. A global ordering of all pixels is first considered. In our tests, this global ordering could result in propagation of errors or temporal inconsistencies more easily. Joint processing of stereo-pairs of frames does not bring a substantial improvement in our tests compared to our default frame-by-frame ordering. Our approach processes each frame in a sequential manner. This sequential ordering has several benefits. First of all, fully completed frames are generated early during computation, whereas with a global ordering one has to wait until the end of the completion to see whether the reconstruction is satisfactory. Moreover, with a careful parallel implementation, one can further reduce computation time by starting the reconstruction at different frames at the same time. This second point is left for future developments. Finally, the sequential ordering allows to exploit fully the benefits of the stereo-temporal blending mechanism detailed in section 6.5.2, as all pixels reconstructed in a previous frame are known and usable for blending.

Application to monoscopic inpainting: In this chapter, we have only presented our results on stereoscopic videos. It would be interesting to assess the performance of our algorithm on monoscopic inpainting problems, where no stereo information is available. To illustrate this problem, we have experimented on the duo sequence extracted from Granados et al. [60]. We invite the reader to watch the results at http://www.sigmedia.tv/Misc/ResultsThesisFelix. The main challenges in this video are caused by pedestrian motions. Firstly, there is the motion of several pedestrians crossing the path of the two persons being removed from the shot, and secondarily, there is also apparent motion caused by reflections in the background. On the other hand, the masks are very tightly fitted to the outline of the two persons to inpaint. This is an obvious advantage for many inpainting techniques, as the quantity of pixels to be recovered is kept to a minimum. However, for motion-based approaches, as Bugeau et al. [23] remark, it is necessary to dilate such masks so as to remove the influence of residual motion from the objects to inpaint. Even after dilation of the masks, our results show some residual artifacts that can be
due to motion. We show an improvement if the motion vectors are set to zero. Setting the motion vectors to zero is a valid approach for this inpainting problem as there is no camera motion and the area to recover is situated on the background. At the end of the day, this example shows that our technique can potentially be applied to monoscopic inpainting. However, the quality of the results can probably be improved by tuning the parameters to compensate for the lack of stereo information and to discard invalid motion estimates.

7.4 Final Remarks

Our results show that the novel stereo-video inpainting technique we describe in chapter 6 is able to fill in missing data in stereo sequences while maintaining temporal coherence and view consistency in the output. In section 7.1, we have presented the small database of stereo videos on which we test our technique. Some tests correspond to real-world situations where no ground-truth data is available and for which a subjective evaluation is the only possibility. We have also designed tests with artificial degradations to fill in, for which the ground truth is available. These tests allow a quantitative analysis, which we perform with the standard SSIM [170] measure as well as an extension of that measure to stereo-pairs of frames [27]. We have showed in section 7.2 that our joint processing of both left and right views allows a better stereo-consistency in the output compared to independent processing of each view in state-of-the-art techniques [88,111,112]. In our comparative study, we have also highlighted the benefits of our hybrid approach, which combines the use of reconstructed motion and disparity within an exemplar-based framework.
This thesis presents contributions to two fundamental aspects of stereo-video post-production, namely segmentation and inpainting. Our research has been focused on extending existing state-of-the-art approaches, designed for monoscopic content, to process stereoscopic videos. We have highlighted in the analysis of our results that the use of long-term motion information in conjunction with inter-view disparity is a key to enable qualitative improvements in further development of stereo-video segmentation and inpainting algorithms.

8.1 Towards User-assisted Dense Stereo-Video Segmentation

In chapter 4 we have presented the results of our user-assisted sparse segmentation technique for stereoscopic videos, which is detailed in chapter 3. The main contributions of our work is to cluster tracks on both views in stereo videos jointly, while increasing long-term consistency of the clusters. Moreover, our method does not necessitate to solve automatically the hard problem of finding the number of objects in a scene but relies on user interaction at the shot level to refine the segmentation.

**Affinity Between Any Pair of Tracks:** Exploiting the affinity between overlapping tracks has been recognised in previous works [22,53,54] to allow segmentation techniques to be robust to partial occlusions of objects. Our technique exploits the affinity between any pair of tracks in the video to form clusters of tracks. Our novel affinity measure is used to compare both overlapping and disjoint tracks, in both views jointly. Exploiting disjoint tracks allows to recover...
the connection between tracks after short-term full occlusions. Our affinity measure is also able to compare tracks located in different views via stereo-mapping of the trajectories using disparity vectors, which is a key to enable our technique to process both views jointly.

**User Interaction in the Sparse Domain:** User interaction is necessary to bridge the semantic gap between the output segments from existing segmentation approaches, and the desired output needed for a given application. This is the motivation for the development of supervised approaches to video segmentation [5, 11]. Existing approaches often require the user to temper with dense masks on key frames along the sequence. In the worst case, the user has to modify the results on each frame, which can be time consuming, and therefore costly in an industrial post-production context. Our approach enables user interaction on all frames in the shot at once, by modifying the results in the sparse domain, in which a video is represented by feature point tracks.

**Towards a Sparse-to-Dense Approach:** The main shortcoming in our approach is that it lacks an extension to the dense domain. Sparse segmentation can be used as a stepping stone to dense segmentation, as highlighted by the recent developments of sparse-to-dense methods [11, 101, 116]. The major difficulty of such approaches is to obtain a comprehensive coverage of the objects in a scene: if key points are not tracked on small or fast objects, it is going to affect the dense output. Our method could be extended in a similar fashion, and the user-assisted refinement step at the sparse level could be used to modify the output prior to computation of the dense segmentation, which is more time-consuming. The idea of keeping the user involved during computation, albeit with a minimal amount of interaction required is certainly a desired quality for practical applications of stereo-video segmentation to post-production.

### 8.2 Towards Segmentation-based Inpainting for Stereo Videos

In chapter 7 we have presented the results of our exemplar-based stereo-video inpainting technique, which is detailed in chapter 6. The main contribution of our work is to exploit long-term motion and disparity information within a constrained texture synthesis framework [34, 45] to process stereo videos. The use of motion and disparity allows to guide the sampling process at the core of the exemplar-based framework. Our results show that our method yields a coherent reconstruction along time and across views.

**Hybrid Approach using Patches and Motion/Disparity:** Our technique combines aspects of patch-based video inpainting approaches [111, 144] to elements of motion-based approaches [23, 88]. Within an iterative patch-matching framework, reconstructed motion and disparity vectors are used as guides, via patch tracking, to constrain the search for the best candidate for replacement of missing data. Unlike techniques based solely on the use of motion
to pull available data to the missing area, our technique uses patch matching to allow corrections from inaccuracies during the tracking process. We also use motion and disparity information to enforce smoothness of the reconstruction along time and across views via a filtering process. The stereo-temporal blending used to filter the reconstruction also takes into account the problem of colour correction. Colour correction should be used in stereo videos to prevent artifacts in case there is a colour mismatch between the left and right views.

Towards a Segmentation-based Approach: Our technique uses a simple image-based interpolation method to infer the missing motion and disparity vector values. The quality of our results could thus degrade if the quality of the reconstructed motion and disparity vectors is too low. In our comparison with existing state-of-the-art video inpainting approaches \cite{88, 111, 112}, we highlight the difficulty for our method and existing techniques to reconstruct missing area containing moving objects with complex motions. By using either a single motion model for the whole video \cite{111, 112}, or non-parametric Optical Flow \cite{88}, it is difficult to process the cases where several possible motions occur in the missing area. Motion segmentation has been recognised as a possible solution to this problem. Previous video inpainting techniques based on segmentation \cite{119, 144, 185} generally inpaint the missing area independently on each motion layer. The layers are then composited together, which can create unrealistic results. The main drawback of these methods, is that they need a precise dense segmentation, which is a very hard problem to solve in practice without time-consuming user intervention. The segmentation and inpainting problems become even harder for multiple objects occluding each other, under free-moving camera motion. It could be beneficial to combine sparse segmentation with inpainting. A possibility would be to use several motion models, corresponding to the various segments in the scene, during candidate search. The different candidates using different motion models could then be used to decide which object is reconstructed. However, tracking with multiple motion models is a difficult problem, even more so when feature points could be occluded by the mask to inpaint. The missing data could be interpolated along time, however there is no trivial solution to this problem.

The Need for Speed: Our stereo-video inpainting algorithm has a relatively simple and generic framework. However, its processing speed is far below the industry standard rig removal \cite{88} implemented in Nuke. Our approach is academic, and we recognise that in real-life scenarios, motion-based techniques can achieve very good performances. A competitive industrial approach using our inpainting algorithm would first need to optimise the speed of the technique. First of all, instead of a Matlab implementation, one would need to use C++ to optimise the performance of our algorithm. Secondly, the speed of our method can be increased by reducing the number of candidates considered during sampling. However, this solution could yield a suboptimal output.
**Towards a Global Approach:** The main drawback of our pixel ordering approach is that it is increasingly difficult to reconstruct the data in the centre of the missing region as the size of the region increases. Artifacts can thus be propagated from frame to frame until a transition occurs with the actual data at this location if it becomes revealed. This problem could be overcome to a certain extent by using a multi-scale implementation of our technique, or even better by performing the reconstruction within a global optimisation framework. These improvements require substantial modifications to our implementation and are left for future developments. Our technique needs to use blending to increase the temporal smoothness of the reconstruction. A global optimisation approach could help improve the temporal consistency of the results.

### 8.3 Final Words

Our segmentation technique is intended to help bridge the semantic gap in state-of-the-art stereo-video segmentation. It is sufficiently quick to allow interactive user-assisted refinement and to be useful for propagation of rough mattes. Further developments could build on our technique to enforce temporal and view consistency in dense segmentation for stereo videos.

Our stereo-video inpainting method intends to bridge the gap between patch-based and motion/disparity-based approaches. There are still a number of problems to solve before it could be used in a day-to-day post-production environment. The main focus of further developments would be to improve computational speed, so as to be able to take advantage of the qualitative improvements of our technique in an industrial context.
Appendix: Video Stabilisation

Digital video stabilisation is useful to improve the viewing experience of shaky videos and eases further processing such as segmentation, encoding and restoration. In this appendix, we present an adaptive, low-latency video stabilisation technique targeting real-time applications for home video and broadcast content viewed on a mobile platform or TV.

Our technique stabilises videos by shifting each frame inversely to the measured 2D translational jitter, which is computed coherently along the estimated dominant motion layer. This allows our technique to stabilise a wide variety of sequences, including the difficult case of zooming. Our smoothing filter preserves intentional motion and keeps a low latency throughout the processing.

We present an extension of our technique to remove rolling shutter artifacts and include a logo detection process to stabilise TV footage such as news or sports, while preserving static areas. We compare our approach to existing methods and show its potential on several sequences captured casually with smartphone cameras and TV content. The work described here has been performed during an internship at Sony’s Stuttgart Technology Centre.

A.1 Related Work

In the past decades, usage of video acquisition devices has dramatically increased. Nowadays anybody can easily record high quality video sequences on mobile devices. But home videos often remain hard to watch due to undirected camera motion and image shakiness. Indeed, amateur videographers seldom use mechanical stabilisation apparatus (e.g. tripod, dolly or Steadicam) to
maintain their lightweight handheld device steady. Home videos that record precious memories cannot be summarily discarded. Improving that content by enhancing its camerawork has thus been a motivation for digital video stabilisation \cite{109,154,180}. Although optical stabilisation can be implemented on video cameras, software approaches offer more flexibility and are less expensive. The latter usually consist of motion estimation, motion compensation and image rendering \cite{110}. A review of each step follows.

A.1.1 The Three Steps of Video Stabilisation

Inter-frame motion is more often represented in the 2D image plane than in 3D space \cite{104} that necessitates complex structure-from-motion. The most common 2D motion model used are the translational \cite{153,154} and affine \cite{180} models. Global motion parameters are usually computed from local motion cues obtained by Feature Tracking \cite{61,180}, Block Matching \cite{154,181} or Optical Flow \cite{6}. Robust parameter fitting is necessary to account for outliers and model shortcomings \cite{154,180}, although video analysis can be bypassed by exploiting built-in motion sensors on smartphones \cite{64}.

Motion parameters are then used to compute a transformation that compensates for unwanted motion so that the remaining intentional motion in the video is *directed* and smooth. In some techniques, it corresponds to low-pass filtered input motion, using Gaussian \cite{109}, IIR \cite{154} or Kalman \cite{180} filters. In other techniques it is fitted directly to the noisy camera path, using smooth parabolic or linear models \cite{61,104}. Adaptive filtering \cite{154} is necessary to preserve deliberate, long-term global camera movements.

The last step, image rendering, reveals unknown areas at the frame borders that should be taken into account. The missing data can be minimised upstream \cite{104}, cropped out \cite{61,153}, or filled by mosaicking or motion inpainting \cite{109}. Distortions can also be explicitly minimised \cite{104}. After stabilisation, remaining motion blur, unrelated to the rendered camera motion, can be removed to further improve the sequence \cite{109}.

A.1.2 Rolling Shutter Removal Techniques

Rolling shutter is a row-by-row acquisition mode for inexpensive CMOS cameras causing additional wobble distortions in shaky videos. These artifacts are treated as structured noise in Liu et al. \cite{104}. To correct these artifacts, the video can be transformed as if all rows were imaged at the same time by computing motion parameters that vary according to image row \cite{6,61,64}. Global motion is estimated from several correspondences within each frame, with additional regularisation \cite{6}, interpolation \cite{64}, or both \cite{61}. Each scan-line can then be aligned so that the resulting global motion is the smooth intended motion.
A.1.3 Logo Detection Methods

All video stabilisation techniques described heretofore displace On-Screen Display (OSD) along with scene pixels when compensating for jitter on TV content. We propose to integrate logo detection \[118, 165\] in our framework to detect and maintain unchanged static OSD, i.e. logos. A logo is assumed to be static and persistent, thus its contour can be detected by computing gradients on a time-averaged frame. Morphological operations then follow to extract the logo area. Geometric or temporal constraints can also be enforced to reduce false detections \[118\]. Animated or semi-transparent or logos \[165\] are harder to detect as a greater amount of change in the background is necessary.

A.2 Video Stabilisation for Home Video and TV Broadcast

We present a video stabilisation system with integrated logo detection and rolling shutter correction. In section A.2.1 we detail our novel translational dominant motion estimation that enables treatment of a wide variety of camera motions, including zooming which remains difficult using prior techniques \[153, 154\]. In section A.2.2 we present our adaptive IIR low-pass filter retaining intended motion while removing high-frequency jitter by combining motion classification and on-line drift correction. The filter parameters can be tailored to the desired system latency, making it suitable for real-time applications. In section A.2.3 we extend our technique to attenuate rolling shutter artifacts via motion interpolation. The method does not need calibration to estimate active time of the sensor \[6\] but works on a smaller class of videos. In section A.2.4 we add logo detection to stabilise TV content while leaving static OSD unchanged. To the best of our knowledge, logo detection had never been integrated with video stabilisation before. In section A.3 we present our results and a comparison with Youtube’s stabiliser \[61\], the Deshaker\[1\] plug-in for Virtualdub and MotionDSP’s vReveal\[2\].

A.2.1 Dominant Motion Estimation

Our system builds on existing Local Motion Estimation (LME) to compute inter-frame motion. We use 3D hierarchical Block Matching with integer pixel precision, similar to Tanakian et al. \[154\]. Other techniques can be employed although quantisation may be needed to keep computation tractable for subpixel valued vectors such as Optical Flow field. The block-based backward motion field, from time \(t\) to time \(t - 1\), is noted BMV\(_t\) in what follows.

Motion estimation for video stabilisation should allow estimation of the jitter that stems from camera unsteadiness. But camera motion is more constraining to estimate than dominant motion, which can be a combination of both camera and apparent object behaviour. Our dominant motion estimator iterates over computation of Dominant Motion Vector (DMV) and

\[http://www.guthspot.se/video/deshaker.htm\]
\[http://www.vreveal.com/\]
Appendix: Video Stabilisation

Figure A.1: Left: motion histograms (with the DMV indicated in red) are displayed along with the DML outliers as red pixels superimposed on frame intensity. Moving objects are consistently removed even when they become large. Right: motion smoothing and drift correction.

Dominant Motion Layer (DML) to remove outliers such as moving objects and texture-less areas, for which LMEs usually fail. Our technique is illustrated in figure A.1, on the left.

We use 2D translational DMV as in similar previous methods \[153, 154\], for computational simplicity and to forbid frame distortion. Global motion is estimated via maximum of motion histogram in Tanakian et al. \[154\]. We compute the vector corresponding to the centre of mass of the histogram instead, with subpixel accuracy (see figure A.1, left) and verified experimentally that our approach allows stabilisation of a broader variety of camera movement, especially zooming, for which translational models either fail or require a complex by-path \[153\]. The DML is represented as a matrix \( DML_t \) which elements are valued from 0 for outliers to 1 for inliers, allowing for partial membership.

We compute the backward DMV as a weighted sum of local backward motion vectors in \( BMV_t \):

\[
DMV(t) = \sum_{v \in BMV_t} H(v) v
\]  
(A.1)

Where \( H \) is the motion histogram, obtained by accumulating weights \( W_t \) in one bin for each motion vector \( v = (dx, dy) \) and normalised to sum to 1. We use a Gaussian spatial prior \( W_g \) to give more importance to vectors close to the centre of the frame as they are more likely to belong to the DML. Element-wise multiplication yields the total weights \( W_t = W_g \cdot DML_t \).

Elements of \( DML_0 \) are initialised as 1. At the current frame \( F_t \), vectors in low-textured areas are weighted out by \( W_e \), thresholded norm of intensity gradients, yielding a first estimate of the DML:

\[
DML^0_t = DML_{t-1} \cdot W_e
\]  
(A.2)

Then for \( n = 1 \) to \( N_{\text{iter}} \) we successively estimate the DMV according to equation (A.1) and update the DML as:

\[
DML^n_t = (\beta DML^n_{t-1} + (1 - \beta) \Theta^n_0) \cdot W_e
\]  
(A.3)

The update step consists in blending the previous DML estimate with the current detected
motion inliers (with $\beta = 0.5$), a binary matrix such that:

$$
\Theta_n^v(i,j) = 1 \text{ if } \| \text{BMV}_t(i,j) - \text{DMV}(t) \|_2^2 < \theta_n^v
$$

(A.4)

$$
\theta_n^v = 1 + \sum_{v \in \text{BMV}_t} H(v) \| v - \text{DMV}(t) \|_2^2
$$

(A.5)

This thresholding operation weights out vectors that do not participate to the DMV computation while enforcing robustness to inconsistencies in edge detection and inaccurate inliers. In our experiments we found that $N_{\text{iter}} = 1$ is enough to remove most of the influence of moving objects, while not pruning too many vectors in degenerate motion cases such as zooming. In figure A.1, left, it can be seen that the influence of the tram is removed in the computation of the DMV. To completely remove the influence of points that are the more likely to be outliers, we set DML values below 0.25 to 0. Similarly, to fully account for points that belong to the DML, we set values above 0.75 to 1.

The goal of DML estimation is twofold: pruning outliers and enforcing temporal consistency of the DMV. When a moving object comes closer to the camera, its motion can be estimated as the dominant one with some techniques [153, 154], thereby possibly causing jitter to be wrongly estimated, which can generate artifacts, e.g. sudden jumps of the image, in the stabilised output.

### A.2.2 Adaptive Motion Compensation

We then decompose the DMV into intended (i) and unwanted (u) motion:

$$
\text{DMV}(t) = \text{DMV}_i(t) + \text{DMV}_u(t)
$$

(A.6)

The unwanted motion is used afterwards to estimate the amount of correction needed to stabilise each frame. The intended motion, assumed to be directed and smooth, is estimated by temporal low-pass filtering of the DMV. We use a two-step filter, made up of averaging and IIR filter:

$$
\text{DMV}_i(t) = \alpha(t) \text{DMV}_i(t-1) + (1 - \alpha(t)) \overline{\text{DMV}}(t)
$$

(A.7)

$$
\overline{\text{DMV}}(t) = \frac{1}{p+k+1} \sum_{s=t-p}^{t+k} \text{DMV}(s)
$$

(A.8)

The averaging step gives $\overline{\text{DMV}}(t)$. For real-time applications a causal filter is needed, so $k$ can be set to 0. We found that using only the next frame improves the stability of the system while keeping a low latency, so we set $k$ and $p$ to 1.

To follow more closely intentional motion, we adapt the IIR filter parameter $\alpha$ depending on motion statistics [154]: intentional and unwanted motion energies. The intentional motion energy is defined as follows:

$$
E_i(t) = \frac{1}{N} \sum_{s=t+T-N+1}^{t+T} \text{DMV}(s)
$$

(A.9)
Where we set $T = 1$ and $N = 7$, using less samples at the start of the video. For a causal filter, $T = 0$, at the cost of higher delay in adaptation. The unwanted motion energy is defined similarly:

$$E_u(t) = \frac{1}{N - 1} \sum_{s=t+T-N+2}^{t+T} |DMV(s) - DMV(s-1)|$$  \hspace{1cm} (A.10)

We use motion classification as in Yeh et al. [181] where the motion regime $R_t$ can be either Temporally Correlated if both $E_u(t)/E_i(t) < K_1$ and $E_i(t) > K_2$ or Random-Like otherwise. We set $K_1 = 0.5$ and $K_2 = 3$. At valid regime transitions, i.e. $R_{t-1} \neq R_t$ and $R_{t+1} = R_t$, we lower the value of $\alpha$ from $\alpha_{\text{max}} = 0.9$ to $\alpha_{\text{min}} = 0.5$ in two frames, with intermediate value $\frac{\alpha_{\text{max}} + \alpha_{\text{min}}}{2}$, and then increase it back to $\alpha_{\text{max}}$ likewise (ignoring regime transitions occurring while $\alpha$ is modified). The idea is to follow intended motion changes while maintaining a smooth motion otherwise. Motion classification and adaptation of $\alpha$ are done independently on X and Y dimensions.

The current amount of jitter is:

$$\Delta_u(t) = \sum_{s=0}^{t} DMV_u(s)$$  \hspace{1cm} (A.11)

As can be seen in figure A.1 (top right), the smooth intentional dominant trajectory $\Delta_i(t) = \sum_{s=0}^{t} DMV_i(s)$ can drift from the gross displacement $\Delta(t) = \sum_{s=0}^{t} DMV(s)$ due to motion vector integration in equation A.7. To avoid this, we monitor the amount of drift between the two curves:

$$\Delta_d = \Delta_u(t - 1) + DMV(t) - DMV_i(t)$$  \hspace{1cm} (A.12)

If $|\Delta_d| > \theta_d$ then we modify the intended motion vector as follows:

$$DMV_i(t) = \Delta_u(t - 1) + DMV(t) - sgn(\Delta_d) \theta_d$$  \hspace{1cm} (A.13)

And update the unwanted motion accordingly. Impacting drift correction on the intended motion allows for smooth correction, as the rectification is taken into account by the low-pass filter (see figure A.1 bottom right). Drift correction is performed independently on X and Y dimensions, with $\theta_d$ set to 5% of the frame width and height respectively.

To render a sequence that preserves intended motion while removing the jitter, each frame $F_t$ is shifted by $-\Delta_u(t)$ using bilinear interpolation. For computational efficiency reasons, and similar to previous methods [61, 153] we crop out the unknown pixel information revealed at the borders according to the maximum compensation $\theta_d$, which can be tuned by the user to balance the desired amount of smoothing and image loss. In the following section we propose an adaptation of our system to correct rolling shutter distortions.

### A.2.3 Rolling Shutter Correction

To attenuate rolling shutter wobbles, we extend our technique by interpolating samples of the DMV for each scan-line of frame $F_t$ of height $h$. We divide BMV$_t$, of height $h_{\text{MV}}$, into $N_{\text{slices}} = 15$
slices, of height \( h_{\text{slices}} = h_{\text{MV}} / N_{\text{slices}} \), on which we apply the method described in section A.2.1 to obtain a sample \( \text{DMV}_{\text{slices}}(k) \) at each slice. Figure A.2 illustrates this procedure. Slices have an overlap of \( h_{\text{slices}} / 2 \) for smoothness. To retain global outliers information we average the slice-based DMLs with a full-frame DML. Edge information is computed for the whole frame, but we use slice-based spatial Gaussian priors. We also found useful to prune outliers more aggressively, so we set \( N_{\text{iter}} = 2 \).

The DMV at each scan-line \( s \) at time \( t \) is computed as a mixture of sliced-based DMVs.

We follow the idea of Grundmann et al. [61] to use Gaussian weights for interpolation, for its flexibility and robustness:

$$\text{DMV}(s, t) = \frac{1}{K_s} \sum_{k \in S} \lambda_k \exp \left( -\frac{(s - c_k)^2}{2\sigma^2} \right) \text{DMV}_{\text{slices}}(k)$$  \hspace{1cm} (A.14)

Where \( K_s \) is a normalising factor, \( \sigma = \frac{3h}{N_{\text{slices}}} \), \( c_k \) and \( \lambda_k \) are the middle row and the sum of DML values, normalised to add to 1, in the \( k \)th slice, respectively. And \( S \) is the set of valid slices, for which \( \lambda_k \geq \frac{1}{\sqrt{2N_{\text{slices}}}} \). As in Baker et al. [6], the motion of each row is aligned to the intentional motion \( \text{DMV}_i \) at the mid-frame scan-line. Temporal smoothing then follows section A.2.2.

### A.2.4 Logo Detection

To stabilise shaky TV videos while preserving static OSD we incorporate a logo detection mechanism to our system. Our logo detection system is illustrated in figure A.3. Similar to [118] we detect persistent edges on a time-averaged frame:

$$F_t = \begin{cases} \gamma(t) F_{t-1} + (1 - \gamma(t)) F_t & \text{if } \| \text{DMV}(t) \|_2^2 > \theta_m, \\ \gamma(t) F_{t-1} & \text{otherwise.} \end{cases}$$ \hspace{1cm} (A.15)

Where \( \gamma \) is defined as in [118]:

$$\gamma(t) = \begin{cases} \frac{t-1}{t} & \text{if } t < t_{\text{start}}, \\ \frac{t_{\text{start}} - 1}{t_{\text{start}}} & \text{otherwise.} \end{cases}$$ \hspace{1cm} (A.16)
Our novelty is motion detection with a threshold $\theta_m = 10$ so as to accumulate frames that provide sufficient background difference to re-enforce static edges only, thus avoiding false detections. Stabilisation starts at frame $t_{\text{start}} = 11$ to allow time for logo detection to converge.

We combine binary edge maps obtained on each colour channel of $F_t$ with logical OR as the use of colours instead of intensity only improves logo contour detection [165]. To extract the logo area we apply dilation to merge groups of pixels followed by binary hole filling. Then small or thin objects are pruned. We define such an object to be made of less than 1000 pixels, or contained in a rectangle bounding box with width or height below 30 pixels. We also remove objects which centroids lie in a region of low motion magnitude:

$$M_t(x) = 1 \text{ if } ||BMV_t(x)||_2^2 < \theta_c \text{ and } 0 \text{ otherwise.}$$

$$\overline{M}_{t_{\text{start}} - 1}$$ is initialised as $M_{t_{\text{start}} - 1}$. We deem an object static if the sum the values of $\overline{M}_t$ in a $3 \times 3$ patch around its centroid is below $\theta_c = 2$. Combining motion cues with image features can help prevent false detections (e.g. when there is no logo in the video) and possibly cover a larger range of OSD. Once pruning is done, we perform erosion to obtain the current logo map. We use morphological operators implemented in Matlab with disks of radius 10 and 8 for dilation and erosion respectively.

The logo map can be unstable for semi-transparent logos over low contrast background, and false detections can still arise even after pruning spurious objects, so we check whether detecting a logo improves stabilisation of the current frame in a post-matching process. Once a logo map is estimated, a stabilised frame is rendered by globally correcting for unwanted jitter as in section A.2.2 while keeping the detected static area unchanged. We render the stabilised frame in three cases: using the current logo mask, using the previous logo mask and using a
no-logo mask. Then image area at each logo object in the previous logo map (or in the no-logo map) and in the current logo map (or in the no-logo map) are compared in the current and previous frames. The current logo map is updated depending on which configuration yields the minimum Mean Absolute Difference, viz. logo from previous, current or no-logo mask. Figure figure A.3 illustrates the post-matching process. Uncovered areas around logos are filled in by copying non-logo data from the previous frame aligned to the current one with DMV, and from the current frame.

A.3 Experimental Results

We have tested our technique on a variety of real sequences, acquired with different smartphone video cameras and containing various camera motions: static, pan, tilt, zoom, dolly. We have also experimented with broadcast footage of sport events containing logos. To assess the gain in smoothness in our results we compare Interframe Transformation Fidelity (ITF) [110] before and after stabilisation. We also measure the Missing Area (MA) necessary to fill if cropping is not used, as a percentage of the total frame area. The videos and our assessment can be viewed at http://www.sigmedia.tv/Misc/VS2012.

Video 1 contains large moving objects that can generate errors in motion estimation in techniques where moving objects are supposed to be small [153] (see the vReveal output). Our technique is able to estimate the DMV of the background (see the DML video) and allows for accurate stabilisation. Video 2 contains zooming, which yields a multimodal histogram that fools motion estimation techniques such as Tanakian et al. [154] where one mode is selected as the DMV in a temporally inconsistent manner, which can generate jitter in the output. In our technique, the DMV is a combination of modes, centred at zero, with variations generated by the jitter that can thus be smoothed out. A costly 2D optimisation step is used by Tajbakhsh [153] to obtain usable translational parameters under zoom. Video 3 contains panning and shows the ability of our technique to follow intended camera motion so as to generate as little missing area as possible. This is important to minimise errors in image filling during future developments.

Rolling shutter wobble is well attenuated by our approach in videos 4 and 5, but severe distortions in video 6 are not corrected as well as in Grundmann et al. [61] by our simple approach. Our scan-line alignment procedure is not suited for sequences with motion along the optical axis (or zooming), in which case fallback would be necessary (wobble is hardly noticeable under zooming). Videos 7 and 8 contain logos that are maintained unchanged by our technique while prior methods move them along with the picture content (see the competitors outputs). Even with our crude hole filling mechanism the sequences are more pleasant to watch. False detection can occur on sequences without logos, which should be avoided in future developments.

In general the Youtube Stabiliser performs the best, even though a few errors in warping that distort the output can be noticed (see videos 1 and 5) and static segments can look unnaturally immobile. Deshaker performs well although efficient rolling shutter correction needs camera
parameters which can be hard to obtain. The vReveal stabiliser has problems with large moving objects and cannot correct rolling shutter wobbles. None of the competitors maintain logos stable in broadcast sequences. Unlike those competitors, our technique works on-line so we think it achieves satisfying stabilisation.

A.4 Final Remarks

We have presented a video stabilisation system that is flexible enough for real-time applications. Our novel translational dominant motion estimator shows that the motion model does not necessarily need to represent the scene’s exact global motion. As long as the estimated model is time-consistent, unwanted jitter can be identified based on the dominant motion layer. The jitter can then be removed by frame shifting, yielding a stabilised video. We have shown that our technique can be combined with rolling shutter correction by using motion interpolation to further stabilise wobbly videos. Integrated logo detection allows our technique to successfully stabilise TV content while preserving logos. For full-frame stabilisation and to improve hole filling behind logos it is necessary to consider video completion techniques such as the one mentioned in Matsushita et al. [109].


[119] K. Patwardhan, G. Sapiro, and M. Bertalmio. Video inpainting under constrained camera motion. IEEE Transactions on Image Processing, 16(2):545–553, 2007. [73] [75] [78] [79] [90] [104] [145]


[185] Y. Zhang, J. Xiao, and M. Shah. Motion layer based object removal in videos. In 7th IEEE Workshops on Application of Computer Vision, volume 1, pages 516–521, 2005. [69] [73] [74] [75] [145]