UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

GUILHERME PINTO FICKEL

Video View Interpolation Using Temporally Adaptive 3D Meshes

Thesis presented in partial fulfillment
of the requirements for the degree of
Doctor of Computer Science

Advisor: Prof. Dr. Cláudio Rosito Jung

Porto Alegre
September 2015

CIP — CATALOGING-IN-PUBLICATION

Fickel, Guilherme Pinto

Video View Interpolation Using Temporally Adaptive 3D Meshes / Guilherme Pinto Fickel. – Porto Alegre: PPGC da UFRGS, 2015.

94 f.: il.

Thesis (Ph.D.) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2015. Advisor: Cláudio Rosito Jung.

1. View interpolation. 2. Stereo. 3. Disparity estimation. 4. Temporal coherence. I. Jung, Cláudio Rosito. II. Título.

"The truth may be puzzling. It may take some work to grapple with.
It may be counterintuitive. It may contradict deeply held prejudices.
It may not be consonant with what we desperately want to be true.
But our preferences do not determine what's true."

— Carl Sagan

ACKNOWLEDGEMENTS

I would like to thank everyone that contributed to the development of my thesis, but I will do so in Portuguese.

Em primeiro lugar, gostaria de agradecer ao meu orientador Cláudio Jung. Obrigado pela paciência, acessibilidade, e pela grande liberdade concedida na pesquisa. Aprendi muito ao longo dos mais de cinco anos em que fui seu orientando de mestrado e doutorado.

Ainda falando da UFRGS, um grande abraço a todos dos laboratórios 232 e 225. Os coffee breaks, churrascos, risadas e trocas de idéias tornaram estes últimos anos bem mais divertidos. E também a todos que reiniciavam o meu PC quando ele travava, geralmente o Henrique, me salvaram de algumas enrascadas.

Um grande agradecimento vai à minha mãe, que sempre me apoiou e incentivou, mesmo que nos últimos anos tenha sido “a distância”. Lembro que tu sempre diz para que eu siga o caminho que me faça mais feliz, e bem, estou sempre tentando seguir esse conselho. Meus irmãos, Roberta, Felipe, Paula e Edwin sempre me apoiaram e torceram pelo meu sucesso. Tenho muito carinho por todos, e agradeço-os por tudo.

E por fim, à Fernanda, muito obrigado por estar sempre ao meu lado. Este caminho não foi fácil, mas seu apoio e paciência(!) me ajudaram a concluir mais esta etapa.

ABSTRACT

This thesis presents a new method for video view interpolation using multiview linear camera arrays based on 2D domain triangulation. The domain of the reference image is initially partitioned into triangular regions using edge and scale information, aiming to place vertices along image edges and to increase the number of triangles in textured regions. A region-based matching algorithm is then used to find an initial disparity for each triangle, and a refinement stage is applied to change the disparity at the vertices of the triangles, generating a piecewise linear disparity map. A simple post-processing procedure is applied to connect the triangles with similar disparities, generating a full 3D mesh related to each camera (view), which are used to generate the new synthesized views along the cameras baseline.

In order to generate views with less temporal flickering artifacts, we propose a scheme to update the initial 3D mesh dynamically, by moving, deleting and inserting vertices at each frame based on optical flow. This approach allows to relate triangles of the mesh across time, and a combination of Hidden Markov Models (HMMs), applied to time-persistent triangles, with the Kalman Filter, applied to vertices, so that temporal consistency can also be obtained. With the proposed framework, view interpolation reduces to the trivial task of rendering polygonal meshes, which can be done very fast, particularly when GPUs are employed. Furthermore, the generated views are hole-free, unlike most point-based view interpolation schemes that require some kind of post-processing procedures to fill holes.

Experimental results indicate that our approach was able to generate visually coherent in-between interpolated views for challenging, real-world videos with natural lighting and camera movement. Also, quantitative evaluations using objective video quality metrics show that our interpolated video sequences are better than competitive approaches.

Keywords: View interpolation. stereo. disparity estimation. temporal coherence.

Interpolação de Vistas em Video Utilizando Malhas 3D Adaptativas

RESUMO

Esta tese apresenta um novo método para interpolação de vistas em vídeos usando câmeras ao longo de um *baseline* baseado em uma triangulação 2D. A imagem de referência é primeiramente particionada em regiões triangulares usando informação de bordas e escala, visando colocar vértices ao longo das bordas da imagem e aumentar o número de triângulos em regiões texturadas. Um algoritmo de casamento de regiões é então usado para encontrar a disparidade inicial de cada triângulo, e uma etapa de refinamento é aplicada para mudar a disparidade nos vértices dos triângulos, gerando um mapa de disparidade linear em trechos. Uma simples etapa de pós-processamento é aplicada para conectar os triângulos com disparidade semelhante, gerando uma malha 3D relacionada a cada câmera, que são usadas para gerar novas vistas sintéticas ao longo do mesmo *baseline* das câmeras.

Para gerar vistas com menos artefatos temporais (*flickering*), foi proposta uma abordagem para atualizar a malha 3D inicial dinamicamente, movendo, removendo e inserindo vértices a cada quadro baseado no fluxo óptico. Esta abordagem permite relacionar triângulos da malha ao longo do tempo, e uma combinação de Modelo Oculto de Markov, aplicado nos triângulos que persistem ao longo do tempo, com Filtro de Kalman, aplicado nos vértices, permite a geração de uma mapa de disparidade com coerência temporal. Com a abordagem proposta, o processo de gerar vistas interpoladas se reduz à trivial tarefa de renderizar uma malha poligonal, algo que pode ser feito muito rapidamente, principalmente quando placas gráficas são utilizadas. Além disso, as vistas geradas não possuem buracos, diferente de muitas técnicas de interpolação de vistas baseadas em *pixels* que requerem procedimentos de pós-processamento para preencher buracos.

Os resultados experimentais indicam que a abordagem proposta foi capaz de gerar vistas interpoladas visualmente coerentes em vídeos desafiadores, com luz natural e movimento de câmera. Além disso, uma avaliação quantitativa usando métricas de qualidade de vídeos mostrou que as sequências de video interpoladas são melhores que abordagens competitivas.

Palavras-chave: Interpolação de vistas, stereo, estimação de disparidade, coerência temporal.

LIST OF ABBREVIATIONS AND ACRONYMS

CUDA	Compute Unified Device Architecture
DIBR	Depth Image Based Rendering
FTV	Free-Viewpoint TV
GPU	Graphics Processing Unit
GPGPU	General-Purpose Computation on Graphics Processing Units
HMM	Hidden Markov Model
MPEG	Moving Picture Experts Group
MRF	Markov Random Field
MSE	Mean Squared Error
PSNR	Peak Signal-to-Noise Ratio
RAM	Random Access Memory
STRRED	Spatio-Temporal Reduced Reference Entropical Difference
WVMF	Weighted Vector Median Filter

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

Computer vision consists in mimicking the abilities of human vision by electronically perceiving and understanding images and videos. However, this is a hard and open problem within the literature. But why? In part, it is because vision is an inverse problem, in which we seek to recover some unknowns given insufficient information to fully specify the solution (SZELISKI, 2010).

Along the years, however, significant advances have been achieved in this area, allowing an efficient use of computers in several real-world applications, such as:

- **Surveillance:** monitoring and analyzing car traffic;
- **Machine inspection:** parts inspection for quality assurance;
- **Automotive safety:** detecting unexpected obstacles on the street such as peoples and animals;
- **Optical character recognition (OCR):** reading handwritten postal codes, automatic number plate recognition.
- **Film Colorization:** coloring a monochrome video by annotating some frame with a few color scribbles, and the indicated colors are automatically propagated in both space and time to produce a fully colorized image or sequence.

To fully emulate the human vision system, however, it is necessary the use of two (stereo) cameras. In the traditional stereo system, two cameras are displaced horizontally from one another, and are used to obtain two different views of the scene(JARVIS, 1983). By analyzing the two available images, the depth information relative to the cameras can be obtained, therefore allowing the extraction of 3D information of the scene. This information can be used in a variety of applications(SCHARSTEIN; SZELISKI, 2002), such as Free-Viewpoint Television (FTV), which allows the user to interactively control the viewpoint and generate new views of a the scene from any position (MORI et al., 2009).

The general approach to generate synthetic interpolated views in-between two real cameras is a simpler task if compared with FTV approaches. It consists in using depth information, usually provided by the disparity map, and distorting one or both images according to the scene geometry and the synthetic view position. The disparity map is based on the correspondence of points from one reference image to another, which in the stereo setting are the left and right cameras. Hence, one key aspect to view synthesis/interpolation is to find a good quality disparity map, since any noise in the disparities reflects as an erroneous representation of the structure of the scene.

Additionally, if rectified cameras are used, the point correspondence problem reduces to finding the disparity of each pixel/region along horizontal scan lines of the images. When dealing with stereo correspondence, most algorithms make the assumption that the pair of cameras are rectified since it diminishes both the complexity of the solution and the computational burden. (SCHARSTEIN; SZELISKI, 2002).

The quality of the estimated disparities is even more important when dealing with video view interpolation, since the presence of small artifacts that would not be very noticeable in a single image may become very salient in a video sequence, possibly causing flickering artifacts that can annoy the viewer. Those temporal artifacts greatly impact the quality of the interpolated video, and to avoid them it is necessary to use a temporally coherent disparity estimation technique. Those techniques incorporate temporal information within its computation, enforcing a smooth disparity variation along the time while allowing abrupt disparity changes in occlusion/disocclusion cases.

1.1 Problem Description

In this thesis we tackle the problem of generating temporally coherent (synthetic) interpolated views within the baseline of a linear array of rectified cameras. The usual approach is to obtain an estimation of the 3D scene, represented by the disparity map of the available cameras. Afterwards, the process of view interpolation warps one or several camera images to the synthetic view position using their respective disparity maps, and finally a post processing procedure is applied to fill the remaining holes. The traditional pipeline for view synthesis, as well as common problems that arise in real scenarios, are briefly described next.

1.1.1 3D Information From Stereo Cameras

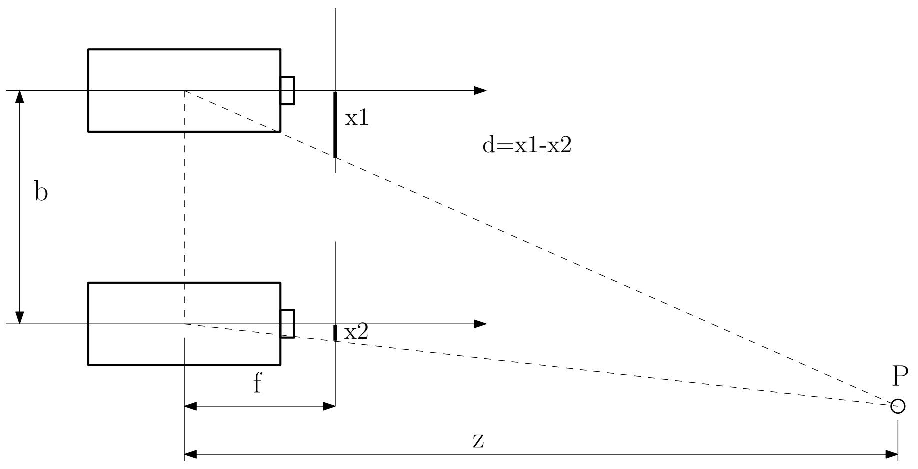
Using two rectified cameras, the disparity d consists in the horizontal position shift for corresponding points between the images, assuming a pinhole camera model. The disparity computation is important since it is possible to estimate the depth z for a given point P in the world by analyzing the disparity between its projections x_1 and x_2, the focal length f and the baseline b. As we can see in Figure 1.1, from similar triangles the depth z can be found with the following equation:

$$\frac{d}{b} = \frac{f}{z}, \tag{1.1}$$

$$z = \frac{fb}{d}, \tag{1.2}$$

with z measured in world coordinates system, f in pixels, b in world coordinates and d in pixels.

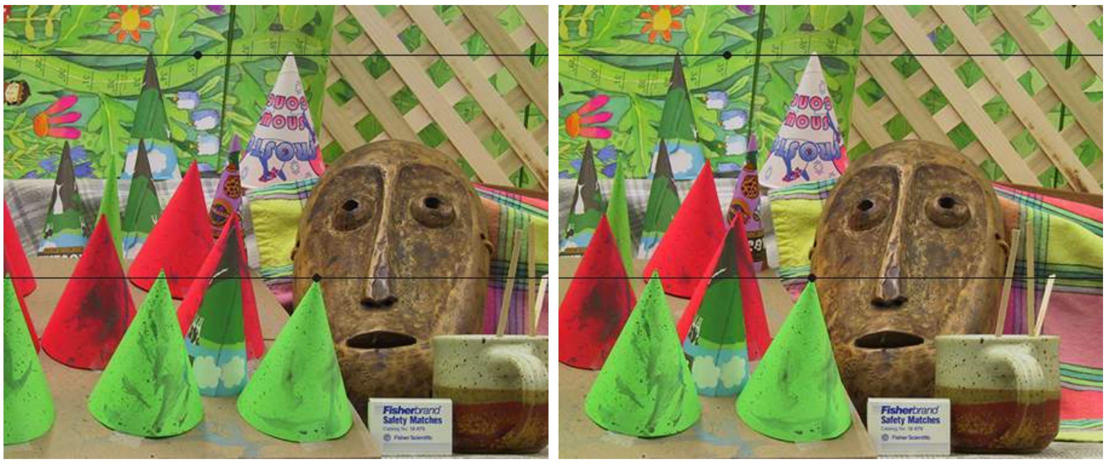
Figure 1.1 – Relationship between the baseline b, disparity d, focal length f and depth z.



However, finding the correspondence pair x_1 and x_2 is a difficult task. Even though we are dealing with rectified images and therefore the search range for x_2 is limited within a horizontal line (as it can be seen in Figure 1.2), there are several problems in the matching process such as

- **Occlusion:** one of the projections from the world point P is occluded by another object in the scene.
- **Matching ambiguity:** when there are several possible candidates for the projection x_2 in the second camera. Usually occurs in large textureless regions.
- **Sensor noise:** image noise that may prevent to find exactly the projection x_2.
- **Bad rectification:** the epipolar lines may not be horizontal, so the search line would not contain exactly the second projection of P.

Figure 1.2 – Example of rectified stereo images. Lines represent the possible search range of the marked points in black. Source: (MIDDLEBURY, 2012).



1.1.2 Basic View Interpolation Pipeline

After the disparity maps have been computed from the available cameras, the view interpolation process can perform a forward warping or backward warping of the camera images to the synthetic view position (MORI et al., 2009). The forward warping consists in using the disparity map of a given camera and warping its image to the synthetic view position. This usually generates some holes in the interpolated view that must be filled using some post-processing scheme. The backward warping estimates the disparity map in the virtual view position, i.e. it warps the disparity of the cameras to the required position. This warped disparity usually contains holes, however they are easier to fill than an image, with the usual approach being to propagate the lower disparity around the hole boundary to the hole itself.

The estimated disparity map is used to warp pixel values from the avaliable cameras to the synthetic location. To remove those holes, a variety of inpainting techniques can be used, impacting the final result (SOLH; ALREGIB, 2012; OLIVEIRA et al., 2015).

1.2 Motivation for This Work

In the typical pipeline for view synthesis, the module for estimating the disparity map is done independently from the view interpolation itself. However, the “quality” of the disparity maps clearly impacts the final view synthesis.

When computing the disparity map (without having the problem of view synthesis in mind), the Middlebury benchmark (MIDDLEBURY, 2012) is the most commonly used metric to rank and objectively compare different methods. However, as shown in (FUHR et al., 2013), such quality metric for disparity estimation presents low correlation with objective metrics used to evaluate view synthesis algorithms, such as SSIM and PSNR (Structured Similarity and Peak Signal-to-Noise Ratio, respectively). The best view interpolation results were achieved with smoother disparity maps, even though they were not well ranked in Middlebury's website. The Middlebury metric does not measure any kind of smoothness in the disparity map since it is modeled as a pixel-based boolean classification system, whose final result is a percentage of bad pixels. Even though we can have a really bad pixel and an almost correct one, they are just classified as good or bad, providing exactly the same weight in the final metric. There is also no indication of where the bad pixels are, since pixels with small disparity errors in low texture regions produce less (or none) visible artifacts in the view interpolation.

The main motivation of this thesis is to develop a solution that that tackles both disparity estimation and view synthesis as two connected problems. By using a triangular domain decomposition, a textured 3D mesh of the scene can be obtained, so that the generation of synthetic views are obtained by simply rendering the textured meshes using the GPU (through the use of OpenGL, for instance). To impose a temporally coherent disparity map we propose to adapt the initial mesh according to the scene changes, instead of generating a new one for each frame. This will allow us to enforce a smooth disparity variation on temporally persistent (triangular) regions, which generates a better quality video view interpolation.

1.3 Goals

1.3.1 Main Goals

We aim to propose a technique that tackles both the generation of a coherent video view interpolation and the estimation of disparity maps as two connected problems. Even though we do not aim to produce the best disparity map according to the widely used Middlebury's ranking, the goal is to obtain disparity maps that are better suited for the problem of view interpolation based on a linear array of two or more rectified cameras.

It is also desirable to use a representation of the scene that makes the generation of novel interpolated views easier, by avoiding common problems such as hole filling and high execution time. And since we are dealing with a linear array of multiple (2 or more) cameras, we aim to use the redundant information available by all the cameras in order to enhance the quality of both the disparity map and the interpolated views.

Finally we want to extend the technique to also generate video view interpolation. Since the use of frame-by-frame view interpolation methods usually introduce temporal artifacts due to the disparity incoherencies across consecutive frames, we will update the initial mesh along the video and enforce a temporal coherence on the disparity generation. This should produce disparity maps with a greater quality and without abrupt changes, while correctly adapting to the changes on the scene such as occlusions and disocclusions of objects.

1.3.2 Specific Goals

- to propose a representation of the scene that connects the disparity estimation with the view interpolation problems;
- to generate a smooth disparity map in homogeneous regions of the image, but allowing discontinuities between edges;
- to use the information of all the available cameras to obtain the disparity information;
- to propose a view interpolation process that uses the information of all the available cameras;
- to propose a temporal mesh update scheme that coherently adapts to the scene changes;
- to explore the temporally coherent mesh to generate video view interpolation with less flickering.

1.4 Thesis Contributions

In this thesis we propose a novel video view interpolation based on a temporally adaptive 3D mesh. The most important contributions of this work are:

- a view interpolation solution that is directly related with the disparity map estimation;
- a slow (offline) pre-processing step, which consists on the generation of textured 3D meshes, with a fast (real-time) view interpolation process;
- a temporally evolving 2D domain triangulation for video sequences that adapts to image content based on optical flow and disparity information;
- a temporally coherent disparity estimation technique based on the adaptive mesh.

1.5 Chapter Conclusions

This chapter introduced the problem that this thesis is tackling, which consists in the generation of temporally coherent view interpolation using a linear camera array. The view interpolation process is usually tackled as two different problems: i) generation of disparity map and ii) warping of the real views, with post-processing steps to remove remaining holes and artifacts.

In this thesis we propose to use an image domain triangulation to segment the image, which is the foundation of the final textured triangular mesh. This textured mesh can be easily used to generate interpolated views in real time and without holes. And finally, we present an extension for multiview sequences that adapts the 3D mesh in time based on video content, aiming to reduce the creation of temporal artifacts.

2 RELATED WORK

The main problem tackled in this thesis is the generation of temporal cohesive interpolated views based on a linear array of at least two rectified cameras. Typically, the generation of view interpolation can be split into two tasks: i) obtaining a dense disparity map, related to the 3D geometry of the scene; and ii) using rendering techniques to combine color/texture information with the disparity maps to generate the synthetic views. In this chapter, we will cover some algorithms for estimating the disparity map, both with and without temporal coherence, as well as techniques for generating the interpolated views based on multiview images and disparity maps.

2.1 Disparity Estimation

The problem of dense stereo/multiview matching has been tackled by several research groups in past years. The main goal is to increase accuracy (some widely used metrics for quantifying the quality of stereo matching algorithms are provided in the Middlebury stereo dataset Middlebury (2012), as well as a ranking of several methods) and reduce execution time, which are conflicting issues most of the time.

In order to solve this problem there are several solutions, each one with a different characteristic such as smoother disparity maps, faster computational time, etc. However they usually have the following structure (SCHARSTEIN; SZELISKI, 2002):

- **Stereo matching** in this step the algorithm, for every pixel within the reference image, computes a matching score for every pixel in the second image (either left or right) within the same horizontal line. For a 2D image this generates a 3D cost cube, spanning in the dimensions X, Y and the range of possible disparities.
- **Aggregation** to get a smoother disparity map it is common to filter the cost cube so that noisy pixels could be corrected by the information of the neighbors. This step is also important to correctly estimate the disparity within low textured regions.
- **Find the best disparity** for every pixel choose the disparity with the highest matching score.

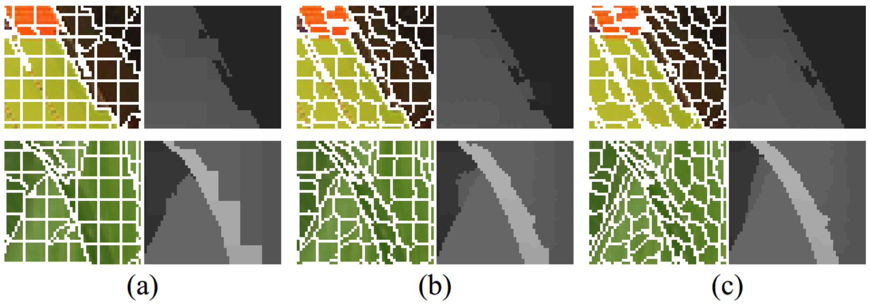
Depending on the disparity estimation technique some of these steps may not be used, or even others may be present. For example, in local algorithms the disparity computation of a given pixel depends only on the color values within a finite window, and it is made a smoothness constraint by aggregating over a support set of surrounding pixels. They usually depend more on the aggregation step to generate a good disparity map, and also may employ a final post-processing step that seeks to enhance the final disparity quality.

Global algorithms, on the other hand, make explicit smoothness assumptions and then solve an optimization problem, therefore there is no need for the aggregation step. Those techniques usually present very good results and a smoother disparity map, especially in large homogeneous areas, however the computational burden is higher. In the work of Scharstein and Szeliski (2002) it can be seen a good classification and taxonomy of disparity estimation techniques.

There are several different ways to approach the problem of stereo/multiview matching. For instance, the approach presented in Zitnick et al. (2004) computes the disparity map from a set of rectified images in two stages. The first stage consists on an over-segmentation procedure of the reference image with a region growing technique, and computes the disparity for each region. In this step it is important to generate regions that do not cross disparity discontinuities. The second stage consists of an iterative refinement based on the coherence of adjacent regions, which is able to both correct most of the disparity errors of the initial estimation and generate a smooth disparity map. The work presented in Taguchi, Wilburn and Zitnick (2008) is similar to Zitnick et al. (2004), but it innovates by using a segmentation algorithm based on a fixed grid that is iteratively updated based on the feedback from the depth map, as it can be seen in Figure 2.1. This provides a depth map with well-preserved discontinuities in exchange to the more computational expansive iterative process. In fact, both approaches present good results, but are very time consuming. Their results also greatly depend on the quality of the segmentation process, which itself is a slow process especially in Taguchi's work.

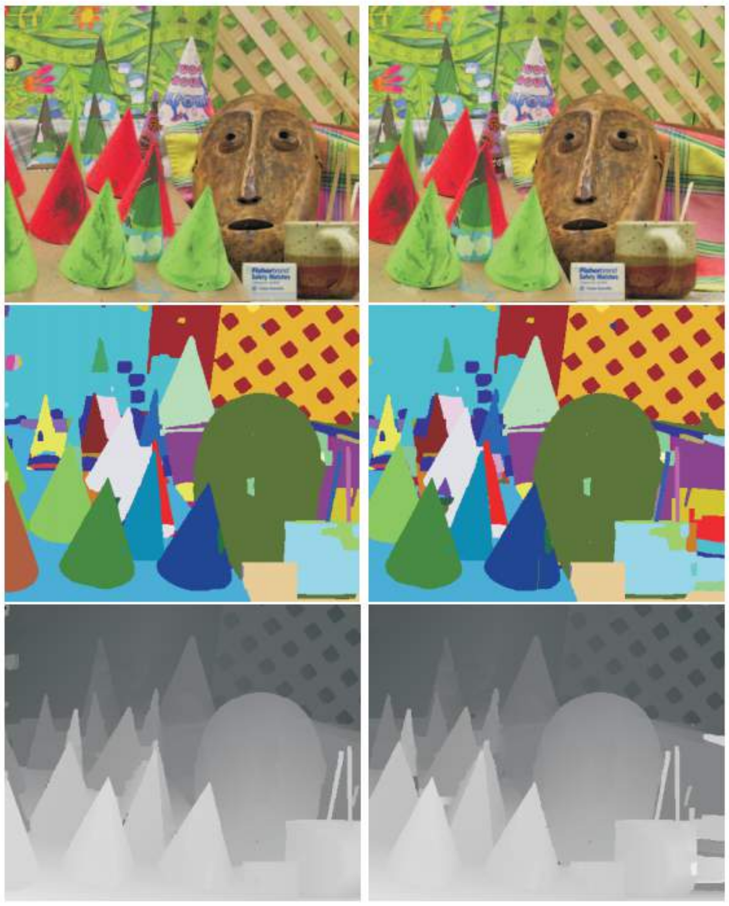
In a different global minimization approach, Bleyer et al. (2011) proposed a joint technique for both the disparity computation and object segmentation. The 3D scene is represented as a collection of coherent objects, characterized by a color model, a 3D plane that approximates the object's disparity and a 3D connectivity propriety. By assuming that 1) each object is compact in 3D, 2) every object is connected and 3) all visible parts of an object have a similar appearance, they modeled this problem as an energy minimization problem solved using a fusion move algorithm (LEMPITSKY; ROTHER; BLAKE, 2007). As we can see in Figure 2.2, the results are very good, both in the disparity computation as

Figure 2.1 – Segmentation and disparity estimation from the work of Taguchi and collaborators Taguchi, Wilburn and Zitnick (2008). (a) initial step, (b) after 2 iterations, (c) after 10 iterations. It can be seen that the segmentation was able to better represent the objects boundaries, even though some small regions present wrong disparities.



in the object segmentation. However, this approach is computationally expensive, taking over 20 minutes to compute for a standard Middlebury dataset of 375×450 pixels.

Figure 2.2 – Joint segmentation and disparity estimation results from Bleyer et al. (2011). On top row we can see the stereo images, in the middle row the obtained object segmentation and in the last row their respective disparity maps.



Some authors try to move the computational burden from the matching itself to the aggregation step. Zhang, Lu and Lafruit (2009) propose to use a simple matching technique with an adaptive support region with a cross-like shape used for both the matching and the aggregation. This structure allows a very fast GPU implementation using integral images and produces good results, but may face problems when highly textured regions are present (since the aggregation window may be small).

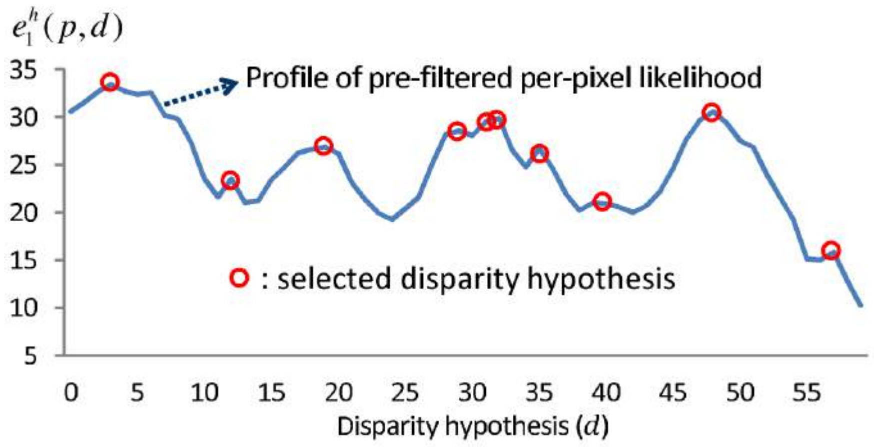
Seeking to surpass the limitations of a local aggregation window, Yang (2015) uses a tree structure derived from the stereo image pair to aggregate the costs. The nodes of this tree are all the image pixels, and the edges of the graph (tree) are the image edges between the nearest neighboring pixels. With this structure, the similarity between any two pixels can be expressed as their shortest distance on the tree. Hence, the proposed method is non-local as every node receives support from all other nodes of the tree. As expected, this non-local aggregation provides very good results with a running speed of 0.8s for a 450×375 image using a C++ implementation (provided by the authors).

A novel approach was proposed by Mozerov and Weijer (2015), in which the author combines an energy minimization method with a cost filtering approach. First, a fully connected MRF (Markov Random Field) defined on the full set of pixels is solved, and the marginal function of the solution is used as the unary potential in a locally connected MRF, which is shown to be related to a cost filtering process. Ideally, the first step should generate better results on non-occluded areas whereas the second step improves the results on occluded regions. This solution achieved good results, and is the second top-ranked stereo matching algorithm according to the Middlebury default standards as of July 2015.

Min, Lu and Do (2013) propose to speed up the disparity computation time using two approaches: 1) reduce the number of used pixels inside a given window by a given fraction and 2) reduce the number of aggregated costs by choosing only a subset of possible disparity candidates. To reduce the number of pixels in the matching process, the authors propose to use only pixels whose coordinates are multiples of a given value S; therefore the number of used pixels decreases as S increases. And to find a subset of candidate disparities to do the interpolation, it is chosen the best D_c disparities that are the local maxima, as we can see in Figure 2.3.

As expected, as S gets bigger than one, the quality of the results starts to slowly decay while the computation time is greatly reduced. However, the parameter D_c had a different impact on the results, since in several experiments the best results were achieved with low values for D_c. So it is clear that unnecessary candidates with low confidence may contaminate the aggregation process.

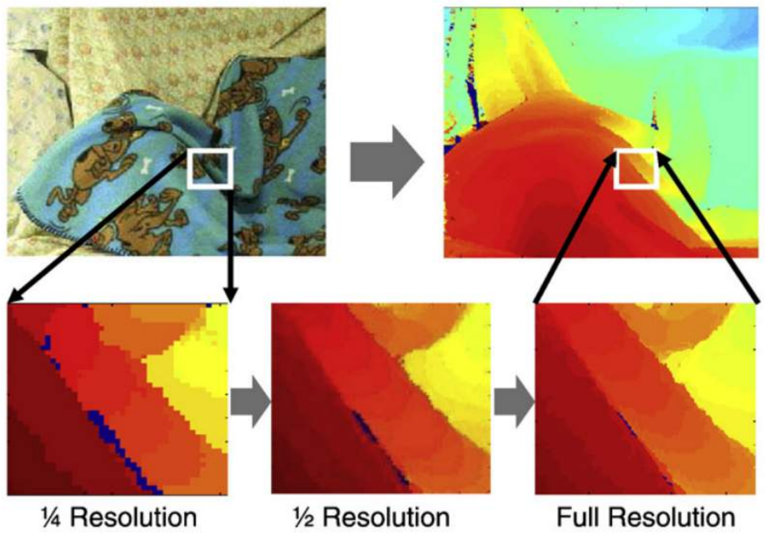
Figure 2.3 – Disparity candidate selection from Min, Lu and Do (2013) with $D_c = 10$, i.e. 10 candidates.



In recent years, several stereo matching methods are using the GPGPU technology to achieve real-time speeds. However, due to the limitations of the GPU hardware and software architecture, most methods usually are pixel-based, with fixed support matching window and with a robust, easily parallelizable matching and aggregation steps. Those approaches are the most common since the GPU performance is directly related to the degree of parallelism of the algorithm. Usually solutions that are fast in GPU are very slow on CPU, and the contrary may also be true. For instance, the approach of Richardt et al. (2010) consists of a GPU implementation of a simple matching technique and a modified aggregation step from Yoon and Kweon (2006). By combining both the luminance and hue information from HSV color space with a very fast implementation of the bilateral filter used in the aggregation step, this solution was able to generate very good results with real time speeds (approximately 14 fps for the Cones dataset (MIDDLEBURY, 2012)). Similarly, Yang (2013) proposes a hardware-efficient bilateral filtering for the aggregation step of stereo matching. Even though bilateral filtering has been proved to be effective, the execution time tends to be high even with the current GPU implementations. With the proposed hardware-efficient bilateral filtering, they were able to achieve good results within the Middlebury ranking system, while achieving a speed of 67 frames per second.

Zhao and Taubin (2011) proposed a GPU approach based on a progressive multi-resolution pipeline which includes background modeling and dense matching with adaptive windows, as it can be seen in Figure 2.4. For applications in which only moving objects are of interest, their approach effectively reduces the overall computation cost quite significantly, and preserves the high definition details. The work of Rhemann et al. (2011) also is centered in a cost-filtering framework, which can also be used in the aggregation step in the stereo correspondence problem. It is shown that, even with a simple matching technique, good results can be achieved with real time speeds (23 fps in average for the Middlebury dataset). In Mei et al. (2011), the authors extend the idea of the cross-shaped windows presented in Zhang, Lu and Lafruit (2009), by including a texture metric and a GPU implementation. As of july 2015, this is the sixth top-ranked stereo matching algorithm according to the Middlebury default standards.

Figure 2.4 – Coarse to fine matching from the work of Zhao and Taubin (2011).

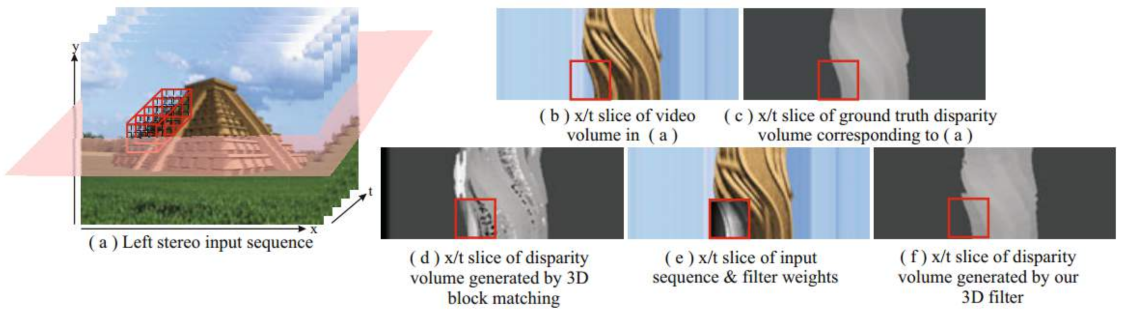


A different approach was proposed by Sun et al. (2014). Instead of focusing on the aggregation step, they find a sparse set of matching points within the stereo images and build a sparse cost cube. The disparities are then propagated in this cost cube according to an edge-aware filtering. They proposed an O(1) geodesic filter, which is shown to be both fast on a GPU implementation (9ms for a standard Middlebury test) and achieved good results for the edge-aware disparity propagation.

2.2 Temporal Cohesive Disparity Estimation

A natural approach to generate temporally cohesive disparity estimation is simply to filter the cost space (the aggregation step, according to the well-known taxonomy proposed by Scharstein and Szeliski (2002)) along time. This was the main idea of Hosni et al. (2012), whose aim was to update a previous work (RHEMANN et al., 2011) to propose a new cost-filtering framework along time. Their idea is based on the assumption that the disparity of a given pixel should be constant along a small space and time window, as it can be seen in Figure 2.5

Figure 2.5 – Spatio-temporal stereo matching from Hosni et al. (2012). For a sequence of frames(a) the cost volume is smoothed using a 3D box filter illustrated in red. The resulting disparity map in (d) does not preserve disparity discontinuities. The proposed approach weights the pixels inside the 3D box filter (e) to achieve a result (f) that is aligned with the space-time object boundaries.


y
x
t
(a) Left stereo input sequence
(b) x/t slice of video volume in (a)
(c) x/t slice of ground truth disparity volume corresponding to (a)
(d) x/t slice of disparity volume generated by 3D block matching
(e) x/t slice of input sequence & filter weights
(f) x/t slice of disparity volume generated by our 3D filter

To calculate the aggregation weights of the pixel near (spatially and temporally) the reference pixel, a guided filter that uses both the spatial and temporal proximity to the reference pixel as well the color similarity is used. This approach allows an efficient GPU implementation, achieving good results with a real-time execution time.

Bleyer and Gelautz (2009) propose to generate a temporally smooth disparity map by filtering a set of disparities within a given time window. A disparity $d_{p,t}$ of pixel p in the frame t is temporally smoothed by computing the median of the following array of disparities D:

$$D = \{d_{p,i} | t - \sigma \leq i \leq t + \sigma\}, \tag{2.1}$$

where σ is a parameter that defines the smoothing strength, i.e. the temporal window size.

However this simple approach fails when there is movement in the scene along time, caused either by moving objects or camera. To tackle this problem the authors compute the optical flow using a fast implementation of Horn and Schunck (1981) algorithm. With this information, the array of disparities D (from Equation 2.1) are computed tracing the pixel p along the time using the optical flow vectors. The obtained results are good, but clearly it highly depends on the quality of the optical flow. Also, the results within the borders of the objects are not consistent due to the limitations of the optical flow in tracking those pixels.

A different cost calculation is proposed by Sizintsev and Wildes (2009). It conceptualizes the stereo correspondence in terms of image spacetime, which encompasses both spatial and temporal characteristics of local pattern structure. The local orientation captures the spatial pattern of observed surfaces, whereas orientations that extend into the temporal dimension capture dynamic aspects, such as motion.

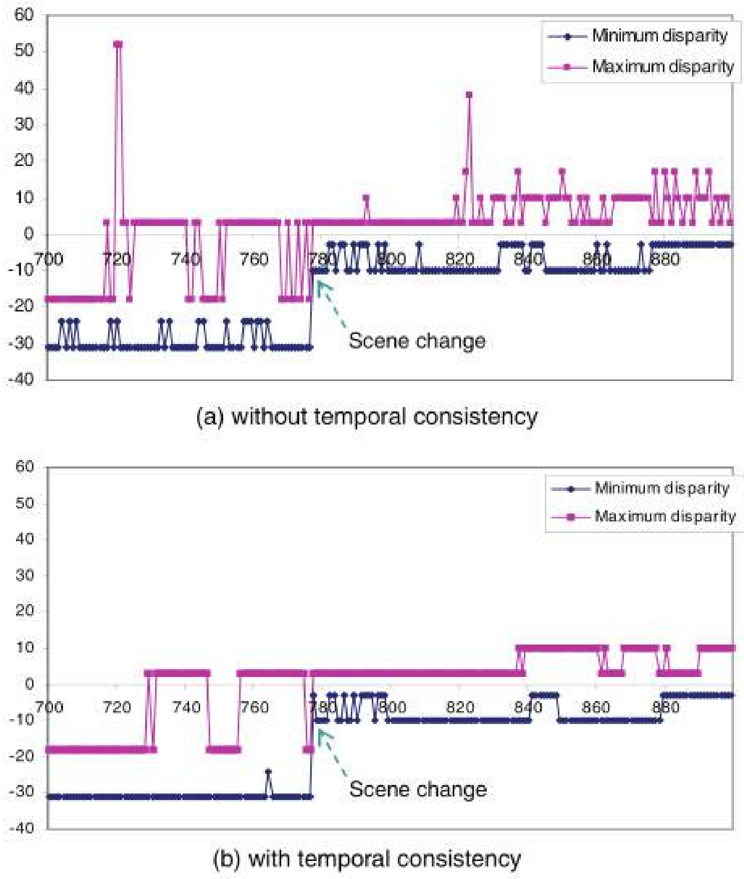
To generate a representation of orientation time, the images are filtered with a series of oriented filters, and the individual energy measures are recast in terms of the proposed spatiotemporal quadric (called ste$\underline{q}$uel), which captures local orientation as well the variance of spacetime about orientation. By the use of the proposed stereo matching using the stequel representation, the authors were able to achieve good results even in challenging video sequences.

Aiming to improve on the stequel representation, Sizintsev and Wildes (2014) proposed a slightly different approach. The representation in terms of the stequel limits the ability to characterize the presence of multiple orientations at a single point since they are all are collapsed to a single quadric. This may hinder the disparity estimation, especially in near surface discontinuities. In contrast, their approach doesn't need to construct an intermediate stequel representation and provides a more direct disparity estimation approach. The results were somewhat better than the previous approach, however they also included the possibility of estimating multilayer disparities in the presence of (semi)transparent and specularly reflecting surfaces.

In order to avoid the problems of local minima and unstable results of disparity estimated along time, Min et al. (2010) impose temporal coherence by limiting the range of possible disparities for each pixel. Based on a disparity histogram that is generated with the sparse feature matching algorithm SURF, temporal consistency is then enforced by calculating a weighted sum of temporally-neighboring histograms, where the weights are determined by the similarity of depth distribution among frames. As we can see in Figure 2.6, the algorithm was able to restrict the disparity search range around a confidence region, avoiding large disparities changes between consecutive frames.

In Lang et al. (2012) the authors proposed a generic framework to impose temporal coherence to a large class of problems, such as optical flow, colorization, scribble propagation, and disparity estimation. They propose to separate the data term from the regularization term and solve them in series. By creating a sparse set of solutions that minimizes the data term locally, they use an edge-aware filtering to create a dense first estimative of the answer. For disparity esstimation based on a pair of stereo images, they

Figure 2.6 – Disparity range search example from the approach of Min et al. (2010). (a) Search range without temporal coherence. (b) Search range with temporal coherence.



first calculate the disparity of a small subset of pixels using SIFT, and afterwards they use an edge-aware filter to create a dense estimative. To estimate the final result the algorithm performs a series of N 1D filterings, alternating between horizontal and vertical orientations until convergence.

When stereo video sequences are used, the alternate 1D filtering scheme is also applied along time. For that purpose, the algorithm estimates the *paths* for each pixel, which are computed by following the optical flow vectors at each frame. By having also the paths for each pixel, the iterative filtering occurs both in horizontal, vertical and temporal dimensions. The results for the disparity estimation were very good, especially within the disparity discontinuities.

Another technique that uses optical flow to impose temporal coherence in the disparity estimation is proposed by Hung, Xu and Jia (2013). They first compute a first estimative for the disparity and optical flow for a given number of frames, and afterwards they create a motion trajectory for each pixel. This approach is more robust then the paths used in Lang et al. (2012) since they use sub-pixel accuracy, and it is able to correctly track points within disparity discontinuities by using information about image edges. Finally, depth and scene flow are refined based on global temporal constraints. One drawback of this technique is the heavy computational burden, requiring 1 minute to generate the depth and optical flow for a 640×480 image using a 24 cores CPU.

2.3 View Interpolation

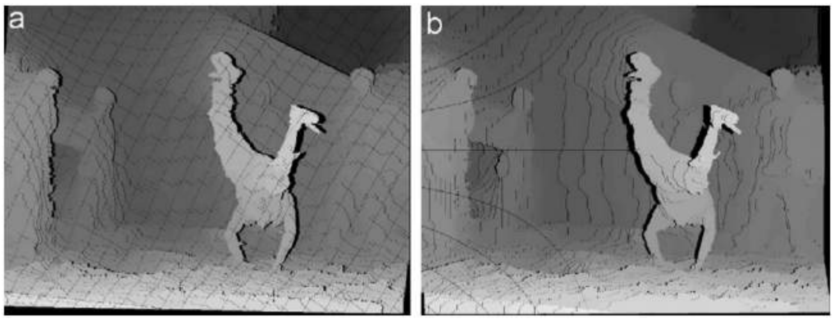
The view interpolation problem can be tackled in several different ways. In Levoy and Hanrahan (1996) an efficient way to represent the light field called light slab was proposed. Using this representation, the interpolated view can be generated by just sampling the images of the given light field. This technique is quite fast and and simple to implement, however the quality of their results depends directly on the density of the light field used.

An interesting approach using view morphing was proposed by Seitz and Dyer (1996). Firstly, the two images are rectified in order to avoid distortions in the morphing stage, and then a dense point matching within both images is computed. The view morphing is then generated according to the projection matrix of the virtual view, which is calculated as a translation between the projection matrix of both real cameras. The authors also propose a more generic morphing that can be used to interpolate within images of different 3D projective transformations of the same object, however it requires some user interaction. Their technique presents similar limitations of common techniques for view interpolation, however. The interpolated view is obtained by projecting the input images, the projections of two points within the same pixel is resolved using the disparity which need to be computed, and it is also necessary to fill the remaining holes.

Other approaches directly use the 3D scene information, such as depth maps. The ones that combine this information with the real views to generate synthetic views are called depth image-based rendering (DIBR) techniques, and face several challenges. The direct warping of pixels to the desired synthetic viewpoint may generate gaps or conflicting information from different images. In particular, occlusions are a common source of problems, since they tend to corrupt the estimate of the disparity maps and also leave gaps in the interpolated views. But even though the disparity estimation is an important step for view interpolation, they are usually tackled as two disjoint problems. The view interpolation step consists in combining the images from the actual cameras with the obtained disparity map(s) to generate a new realistic view of the scene.

Mori et al. (2009) proposed a view synthesis approach based on depth warping. In their approach, the depth values from different real views are warped to the virtual viewpoint, holes are filled with median filters, and smoothing is performed with a bilateral filter. The results of the projected disparity maps without any post processing to remove the holes can be seen in Figure 2.7. These two depth maps are projected to each real camera, and the virtual view is rendered by blending the two neighboring images (alpha blending is used at most pixels, but either the right or left images may be used alone at occlusion pixels). Their results have good visual quality, and the implementation of their approach is used as the reference software for MPEG standardization experiments for Free-viewpoint TV (MPEGFTV). However, the disambiguation when two 3D points project to the same pixel is based on the closest distance, so that a wrong disparity estimation leading to a small depth may corrupt the warping process.

Figure 2.7 – Projected disparity maps from the left and right cameras from the algorithm of Mori et al. (2009). (a) Projection from the left side and (b) projection from the right side.



Ndjiki-Nya et al. (2010) focus on the problem of filling gaps with visually coherent texture. In their approach, the authors initially find the depth map incoherencies, and assume that they belong to the background. Then, the Laplace equation is used for an initial estimate of the pixel values on uncovered regions, and patch based texture synthesis is used to complete the process. Despite the good results shown by the authors, wrong depth estimates may lead to significant degradation of the rendering results. Müller et al. (2009) proposed a scheme to deal with inconsistencies of the disparity map in different views, known to produce visual artifacts. In their approach, reliable areas are projected first, and unreliable areas (detected along depth discontinuities) are split into foreground and background data. Foreground areas are projected first and merged with the reliable data, and then background data is projected. Clearly, a key issue in Müller et al. (2009) is the reliability in which foreground and background are discriminated. In Tian, Vetro and

Brand (2011), a trellis based view synthesis framework is proposed to deal with the view synthesis problem. In their approach, the depth of each pixel is chosen from a candidate list, aiming to maximize the estimate of the synthesis quality. The results presented in Tian, Vetro and Brand (2011) indeed showed reduction of image artifacts, but the problem of filling holes independently for each generated view remains.

The number of gaps in the interpolated views can be reduced or eliminated if 3D meshes are used. For instance, Zitnick et al. (2004) built a 3D mesh from the obtained disparity map, and rendering is implemented on a GPU. Although the results look good, the method is complex and requires a considerable amount of pre- and post-processing operations.

Pujades and Devernay (2014) compares direct (such as DIBR techniques) and energy-based methods for the interpolation process. The results showed that this variational approach presented little to none improvement over standard techniques on lambertian scenes, despite being more computationally expensive. However, for non-lambertian scenes the improvement was noticeable, especially in rendering the structure of the image.

There are also some approaches for obtaining interpolating meshes directly from oriented point clouds (BERNARDINI et al., 1999), but they are sensitive to noise. Another family of algorithms for 3D reconstruction based on point clouds explore implicit function fitting (KAZHDAN; BOLITHO; HOPPE, 2006; CALAKLI; TAUBIN, 2011), which are more resilient to noise, but with considerable computational complexity.

2.4 Chapter Conclusions

This chapter presented a revision of existing techniques related to the main focus of this thesis. For the disparity estimation problem there are basically two different approaches: global optimization solutions, such as Yang (2012), Bleyer et al. (2011), Chen et al. (2013) and local algorithms, such as Min, Lu and Do (2013), Taguchi, Wilburn and Zitnick (2008), Zhang, Lu and Lafruit (2009). In particular, some of them allow highly parallel formulations suited for GPGPU implementation (RICHARDT et al., 2010; YANG, 2013). Even though global optimization algorithms usually presented the best results (in terms of the quantitative metrics for the Middlebury benchmark), recent local minimization algorithms have achieved similar results, and usually with a lower execution time (SUN et al., 2014).

To create temporally coherent disparity maps, one approach is to also calculate the optical flow and refine one with the information from the other. This can be done with a computationally costly minimization of both (disparity and optical flow) at the same time (HUGUET; DEVERNAY, 2007), or by minimizing them separately (HUNG; XU; JIA, 2013; LANG et al., 2012). A simpler possibility is to either modify the matching cost to include temporal information instead of only doing it in a frame-by-frame manner (SIZINTSEV; WILDES, 2009), or even to filter the matching costs along both space (X and Y orientations) and time (HOSNI et al., 2012; BLEYER; GELAUTZ, 2009). Although those solutions are typically faster and simpler, the use of optical flow allows more accurate tracking of pixels in time, providing a much more robust and direct method to impose temporal cohesion.

For the view interpolation problem, it could be observed that even though the disparity estimation is of great importance in the generation of a good quality synthetic view, stereo matching and view interpolation are usually treated as two disjoint problems. The most common approach is to simply warp the available images guided by the disparity maps and afterwards fill the remaining holes (MORI et al., 2009; NDJIKI-NYA et al., 2010; MüLLER et al., 2009). It is also possible to use a triangular mesh to represent the scene, and the generation of the interpolated view becomes a trivial renderization of the mesh (ZITNICK et al., 2004). However, to directly obtain an interpolated mesh from the point cloud is a hard task (KAZHDAN; BOLITHO; HOPPE, 2006; CALAKLI; TAUBIN, 2011).

Next, we present our approaches for disparity estimation and view interpolation suited for linear camera arrays.

3 GENERATION OF THE 3D MODEL

Let us consider a linear array of $N_{\mathcal{C}}$ cameras C_1,...,$C_{N_{\mathcal{C}}}$ with the same intrinsic parameters and same camera rotation parameters, and translated along the axis of the array (so that all captured images should be rectified). Also, let b_i denote the baseline of camera i with respect to the first camera C_1, for $i = 2, ..., N_{\mathcal{C}}$. Note that if $N_{\mathcal{C}} = 2$, the problem is reduced to the classical stereo matching problem of rectified images.

Our approach consists of initially generating a 3D model of the scene using as reference each of the $N_{\mathcal{C}}$ cameras, and then synthesizing new views using a virtual camera at any position along the array. To obtain a 3D model using a given camera, we first generate a 2D triangular tessellation of the reference image, and then employ a region-based approach to compute the matching cost at each possible disparity for each triangular region. A non-local aggregation step that uses the information of all the triangles is used, generating a piecewise constant disparity map. Then, a refinement procedure is applied to smooth the disparity map, generating a piecewise linear depth map. Finally, a full 3D triangular mesh is generated by connecting vertices with similar disparity values, and the texture of occluded triangles are retrieved from neighboring cameras. These steps are described next.

3.1 Triangular Tessellation

The triangular tessellation step aims to partition the reference image into several triangular regions, so that each region should represent a small portion of a single object in the scene that is likely to have a uniform disparity. Those triangles should be larger in homogeneous areas to avoid ambiguities in the disparity estimation process, while preserving the disparity discontinuities and correctly representing finer details.

The first step is to find an adequate set of vertices for the triangular tessellation. In areas where depth is homogeneous, the density of points should be small, whereas close to depth discontinuities the number of vertices should be higher to correctly represent the objects boundaries. Since the depth discontinuities are usually associated with image edges, it is desirable to prioritize the placement of vertices along image edges.

To initialize the vertices, we compute an edge map of the reference image, and vertices are uniformly spread along the edges, with a minimum distance of P_{min}.According to our experimental results, we found that $P_{min} = 9$ generated vertices sufficiently close to represent smaller details of the scene while avoiding small triangles, which tends to present higher ambiguities in the matching process. There are several existing methods for detecting edges in monochromatic or color images, which would impact the placement of the initial vertices. Although any edge detector could be used in this step, we chose the Canny operator (CANNY, 1986), which is a classical edge detector based on oriented gradients and non-maxima suppression. For the sake of comparison, we also used a more recent approach (DOLLÁR; ZITNICK, 2013) and the final results were similar. Hence, due to the higher run time of Dollár and Zitnick (2013) with only marginal improvement on the interpolated view, we choose to use the Canny detector. Further comparisons can be seen on Section 3.6.

To populate the remaining of the image with vertices, we use a technique similar to the particle generation procedure proposed in Sand and Teller (2008). This is accomplished by calculating a scale map of the image that provides, for any given pixel, the local image complexity. With this information, the goal is to distribute the remaining vertices according to the local homogeneity of the image, increasing the density in complex (textured) regions.

The scale map is calculated by filtering the image with Gaussian kernels, which scales are given by $\sigma(j) = 1.9^j$ for $0 \leq j \leq 5$. Then, for each pixel, we find the range of scales in which a pixel value does not change substantially. If the pixel color remains similar across several scales, it represents a pixel with a low visual complexity around it. For a given pixel (x, y), we compute

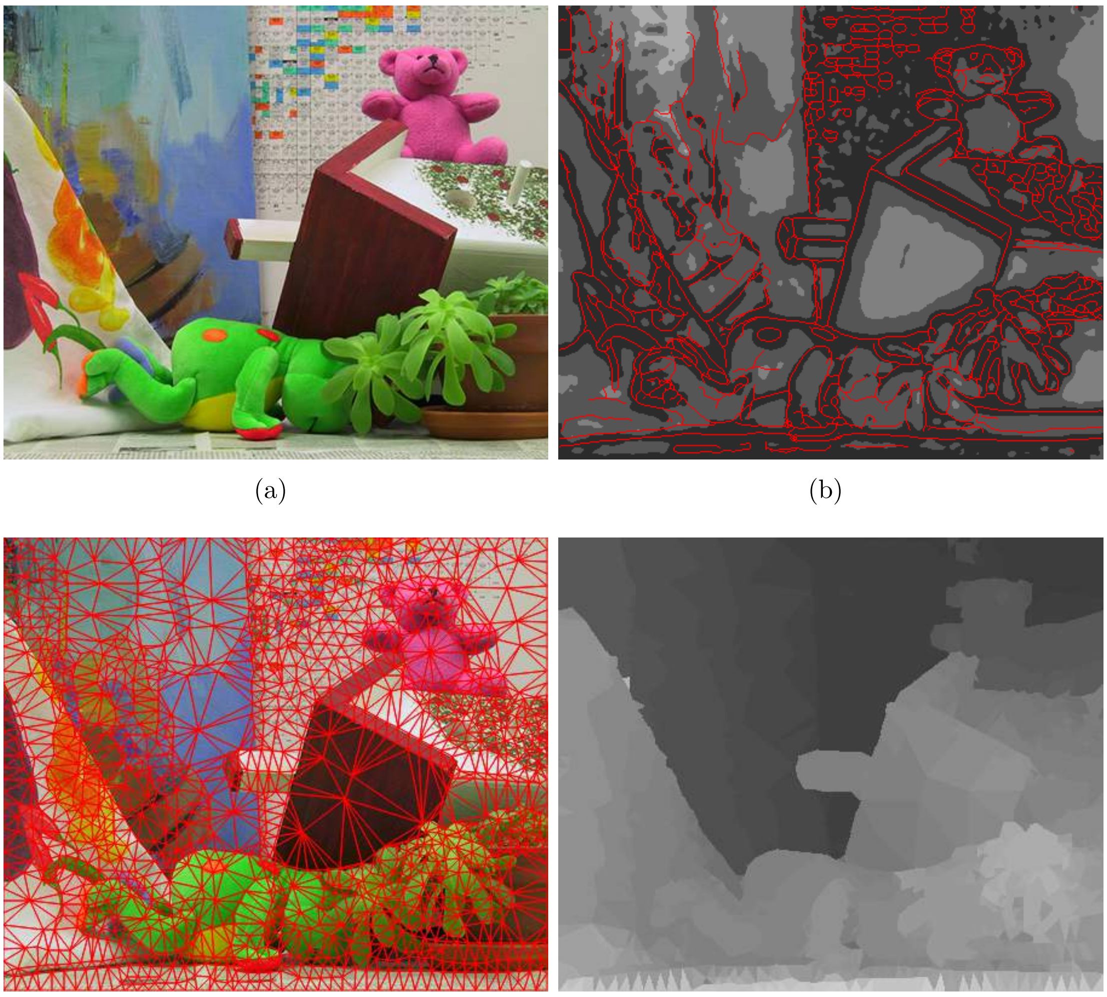
$$k(x,y) = \underset{j}{\operatorname{argmax}} \left\{ j \middle| \ \|I_l(x,y) - I_1(x,y)\|_2 < \delta_s, \forall l = 1, ..., j \right\}, \tag{3.1}$$

where δ_s is a threshold (we used $\delta_s = 10$, as in Sand and Teller (2008)), $\|\cdot\|_2$ denotes the L^2 vector norm, and $I_l(x, y)$ is a 3D vector with the RGB values at scale l. In other words, $k(x, y)$ is the largest scale for which the color of pixel (x, y) remains roughly constant after successive smoothing with Gaussian kernels.

Finally, $k(x, y)$ is smoothed with another Gaussian kernel with standard deviation $\sigma_s = 2$, producing a final blurred scale index map $\hat{k}(x, y)$ that is used as a minimum distance threshold. More precisely, we perform a raster scan in the image, and add another vertex at a given position (x, y) if its distance to the closest existing vertex is greater than $1.5^{\hat{k}(x,y)+g}$, with $g = 6$ controlling the density of vertices. To ensure that the final triangulation will encompass all the image, vertices within a distance P_{min} are spread along the boundaries of the reference image. With this set of vertices, the final triangulation is obtained by the Delaunay triangulation.

An illustration of the tessellation step is presented in Figure 3.1. The reference image is shown in Figure 3.1(a), and the scale map with superimposed edges is shown in Figure 3.1(b). The final tessellation is shown in Figure 3.1(c), and it can be observed that regions with higher textural information present more and smaller triangles (e.g. the periodic table).

Figure 3.1 – Triangulation steps. a) reference image (Teddy from Middlebury dataset) b) scale map (grayscale) superposed with the edges map (red) c) final triangulation in red d) initial disparity estimation.



(a) (b)

(c) (d)

3.2 Initial Disparity Estimation

After the image tessellation step, each triangular region is expected to present a roughly uniform disparity value. The initial disparity estimation consists of assigning a single disparity value to each triangular region, using an approach similar to Zitnick et al. (2004), Fickel et al. (2012). Since we are dealing with a set of rectified cameras along the same baseline, the matching process reduces to finding the maximum of a similarity matching function computed for a range of possible disparity values along a horizontal scan line.

In this work, we used the matching function proposed in Zitnick et al. (2004). It is based on the histogram of pixel gains, and chosen due to its ability in dealing with illumination variations across cameras. Let us consider a triangle r_j in the reference image I. For each color channel $c \in \{R, G, B\}$ and for each pixel x in r_j, we compute the ratios $ratio_d^c(x) = I^c(x)/I'^c(x+d)$, where I' is the other image in the stereo pair, and d is the disparity value. An ideal match at the correct disparity value should lead to similar ratios at all pixels and color channels, so that a sharp peak at the histogram of $ratio_d^c(x)$ is expected. Following the idea presented in Zitnick et al. (2004), we compute such histogram for each color channel using 20 bins $\beta_l^c(d)$, ranging from 0.8 to 1.2, and ignoring the values outside this interval. Then, the matching function for m_j at disparity d is given by

$$m_j(d) = \frac{1}{3A_j} \max_l \left\{ \sum_{c \in \{R,G,B\}} \beta_{l-1}^c(d) + \beta_l^c(d) + \beta_{l+1}^c(d) \right\}, \tag{3.2}$$

where A_j is the area of triangle r_j and l is the bin index. It is important to note that $0 \leq m_j(d) \leq 1$ for any disparity d, and good matches should lead to $m_j(d) \approx 1$, since the maximum sum of three adjacent bins should be close to the total number of pixels (i.e. the area of the triangle).

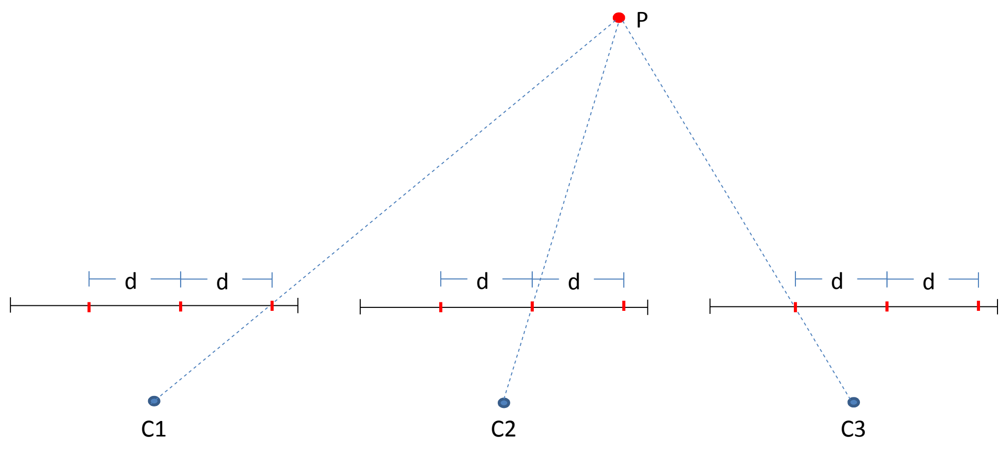
When more than two cameras are available, the joint information of the additional cameras may potentially improve the estimated disparity map. Without loss of generality, let us assume that the first camera C_1 is the reference camera. If the disparity of a triangle t with respect to the adjacent camera (C_2) is d, then its disparity with respect to any other camera C_l is given by $d + \frac{b_l - b_2}{b_2} d$, which reduces to $(l-1)d$ if all the cameras are equally spaced. Figure 3.2 illustrates the relation between the disparity of 3 equally spaced cameras.

For a given disparity range $\mathcal{D}$, we want to find the disparity $d_j \in \mathcal{D}$ that maximizes a multiview matching function

$$m'_j(d) = \frac{1}{N_{\mathcal{C}}} \sum_{l \in \mathcal{C}} m^l_j \left(d + \frac{b_l - b_2}{b_2} d \right), \tag{3.3}$$

where $\mathcal{C}$ is the set of cameras used for matching, $N_{\mathcal{C}}$ is the number of cameras, and $m^l_j(d)$ is the matching function for triangle t_j at disparity d with respect to camera l. As in Equation (3.2), the matching values $m'_j(d)$ are also restricted to the interval $[0, 1]$.

Figure 3.2 – Disparity estimation using multiple cameras equally spaced. Between any two neighbor cameras, the disparity is d.



3.3 Aggregation

Aggregation is widely used in the traditional stereo matching pipeline, and it consists mainly of filtering the matching function in a spatial neighborhood at each disparity before finding the best match. Its main goal is to provide smoother disparity map, and at the same time avoid blurring the disparity discontinuities. To accomplish this task, there are several possibilities, such as bilateral filtering (RICHARDT et al., 2010) or using an adaptive mask that combines only the costs of pixels with similar colors (MEI et al., 2011), among others. In Fickel et al. (2012), the matching function was aggregated using the information of all triangles that share vertices. However, the corresponding spatial neighborhood used in the aggregation step is highly related to the triangle sizes: for small triangles (detailed portions of the image), the neighborhood ends up being small, whereas for large triangles (less details) the neighborhood is very large.

In Fickel et al. (2013) the aggregation window was a circular region with fixed radius (we set the default value of 24 pixels for this radius based on experiments, since this is directly linked to the average size of the triangles, which is independent from the size of the images). This approach is also a local method that only evaluates the neighboring regions, so that large homogeneous regions (for which the initial disparity estimate tends to be not reliable) can not benefit fully from the aggregation step. On the other hand, global methods tend to present better results, but at the cost of increased computational cost. However, Yang (2015) presented a non-local aggregation step for pixel-based methods that is fast and runs in linear time. In this work, we adapt the approach of Yang (2015) for our triangular domain, as described next.

The non-local aggregation filter matching costs are based on region similarity, using a graph structure that is obtained from our image triangulation. The nodes in the graph are the triangular regions, and triangles that share an edge in the mesh are connected by an edge in the graph, i.e. each region has exactly three neighbors. The weight of the edge connecting triangles r and s is given by

$$W(r, s) = \delta_{rs}\gamma_{rs}, \tag{3.4}$$

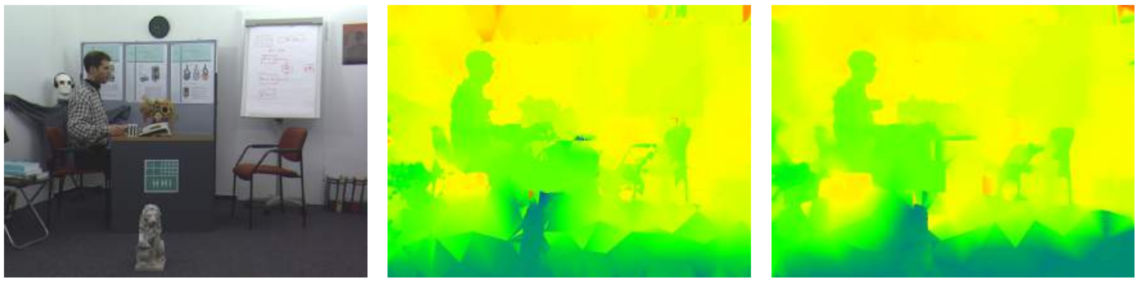
where δ_{rs} and γ_{rs} are the color and spatial distances between r and s. More precisely, δ_{rs} is given by the RGB distance between the mean colors of the regions, and γ_{rs} is the Euclidean distance between the centroids of r and s. In Yang's original work, the cost involved only color distances, and it was applied to every pixel in the image using the 8 neighboring pixels. Hence, the proposed extended cost given by Equation (3.4) coincides with the original solution in Yang (2015), and allows to compute the cost of non-adjacent regions taking into account their spatial distance.

The next step consists in generating a Minimum Spanning Tree (MST) in which the cost filtering will be computed. The similarity between any pair of regions is computed as the sum of weights of the minimal cost path between them. With $\mathcal{D}(r, s) = \mathcal{D}(s, r)$ being the distance between r and s along the MST, the aggregated costs can be computed using the MST structure:

$$m''_r(d) = \sum_s \exp\left(-\frac{\mathcal{D}(r, s)}{\sigma_a}\right) m'_s(d), \tag{3.5}$$

where $m'_r(d)$ and $m''_r(d)$ are the original and aggregated matching costs for region r and disparity d, respectively, $\sigma_a = 0.2 \times width$ controls the decay of the costs within the MST, with $width$ being the width of the input image sequence.

Figure 3.3 – Disparity maps estimated with different aggregation techniques. a) Reference image b) Bilateral filtering aggregation (from Fickel et al. (2013)) c) Proposed non-local aggregation.



(a) (b) (c)

An interesting property of this non-local aggregation is that all connected triangular regions with similar colors (i.e. a painted wall) may be used in the aggregation stage, regardless of their distances, which avoids the problem of a limited search range in local techniques. Also, the cost function given by Equation (3.4) generates a final distance in the MST that is more similar to geodesic than Euclidean, so that two regions with a similar color will have a small aggregation weight if there is another region with different color between them in the path, despite their spatial distance. This property avoids the agggregation of regions that have similar color but belongs to different objects, which is a problem for local methods with large search ranges (including the bilateral filter). Finally, using the linear time algorithm from Yang (2015), this non-local cost filtering technique is faster than the original bilateral filter of Fickel et al. (2013) and typically presents better results for real world videos, as illustrated in Figure 3.3.

Finally, the aggregated disparity value for each triangle r_j is given by $D_j = \text{argmax}\{m''_j(d)\}$, i.e. it is the one that maximizes the value of the aggregated matching function, as in traditional winner-takes-all strategies.

3.4 Refinement

At the end of the aggregation procedure, a piece-wise constant disparity map is obtained, so that all three corners[1] of a given triangle present the exact same disparity (depth). In the refinement step, the disparity of each corner is changed based on the similarity of the triangles that share the same vertex, so that corners related to similar triangles have their disparity values moved closer to each other. Since all three corners of each triangle are potentially changed, the final result is a piece-wise linear (planar)

[1]In computational geometry, corner is an abstract association of a triangle with a vertex. In our triangular 2D mesh, each corner encodes a disparity value.

representation of the disparity map.

Let us consider a vertex v that is shared by N_v triangles, and let D_j $(j = 1, ..., N_v)$ be the disparity of the corner related to each of the N_v triangles at that vertex. Our goal is to find refined disparity values d_j for each corner such that the disparity difference is reduced when the color of triangles are similar (to smooth the disparity map), and kept mostly unchanged when the triangles are not similar (to preserve disparity discontinuities along the boundaries of the objects). The refinement step is formulated as an optimization problem, and the goal is to minimize the following objective function E is given by

$$E(d_1, ..., d_{N_v}) = \frac{2}{N_v(N_v+1)} \sum_{i=1}^{N_v} \sum_{j=i+1}^{N_v} \omega_{ij} \left(d_i - d_j\right)^2 + \frac{1}{N_v} \sum_{j=1}^{N_v} \lambda_j \left(d_j - D_j\right)^2, \quad (3.6)$$

which is a quadratic function. And since E is an concave upward function, it has only one global minima which can be easily found by solving an $N_v \times N_v$ linear system $\boldsymbol{Ad} = \boldsymbol{D}$ for each vertex V, where

$$\begin{aligned} \boldsymbol{A} &= [a_{ij}] \text{ with } a_{ij} = \begin{cases} \lambda_i + \frac{2\sum\limits_{k \neq i} \omega_{ik}}{N_v+1}, & \text{if } i = j \\ -\omega_{ij}, & \text{if } i \neq j \end{cases}, \\ \boldsymbol{d} &= \begin{bmatrix} d_1 & d_2 & \cdots & d_{N_v}, \end{bmatrix}^T \\ \boldsymbol{D} &= \begin{bmatrix} \lambda_1 D_1 & \lambda_2 D_2 & \cdots & \lambda_{N_v} D_{N_v} \end{bmatrix}^T. \end{aligned} \quad (3.7)$$

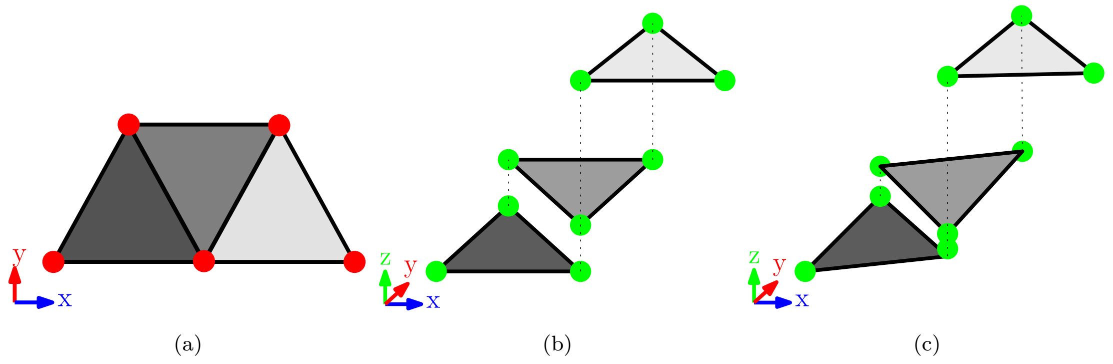
Since each vertex is related to an average of 6 corners in a triangular mesh, the refinement step consists of solving an average of $\#\mathcal{V}$ 6×6 linear equations, where $\#\mathcal{V}$ is the number of vertices in the mesh.

In Equation (3.6), the first term is a weighted average of the squared disparity differences of the corners. It may be interpreted as a regularization term, and it is minimized when all disparity values d_j are the same. In particular, the similarity term ω_{ij} should be large when triangles r_i and r_j are similar, to enforce coherence of the corresponding corners. On the other hand, ω_{ij} should be small when r_i and r_j are not similar, to allow disparity discontinuities between different objects. The second term in Equation (3.6) is a weighted average of the squared disparity differences between the initial disparity estimates and the refined disparities. It accounts for data fidelity, and it is minimized when $d_j = D_j$, i.e., when the refined disparity values are exactly the same as the initial disparity estimates. In particular, data fidelity should be enforced for a given corner if the initial disparity value D was already a good estimate, and relaxed when such estimate is bad. Hence, λ_j should be large when the initial disparity of triangle t_j was estimated with good confidence, and smaller otherwise.

Based on these comments, we define $\lambda_j = m''_j(D_j)$, recalling that m''_j is the matching function for r_j at disparity D_j according to Equation (3.5). Also, we define $\omega_{ij} = e^{-\delta_{ij}/\gamma_c}$, where δ_{ij} is the similarity distance, defined as the Euclidean distance between the mean RGB values of r_i and r_j, andγ_c is a threshold that control the decay of the color distance. According to experimental results, we have selected $\gamma_c = 10$.

Figure 3.4 illustrates the refinement process. Notice that each triangle has its own set of 3 corners, with the z coordinate being represented by the disparity value. After the refinement, the corners of each triangle are changed according to their matching costs and color similarity of the neighboring triangles.

Figure 3.4 – Vertices and corners examples. a) Triangles viewed in xy plane, with corners in red. b) Triangles viewed in a perspective angle, with corners in green. c) Corners after the refinement step.



3.5 Generation of the 3D Mesh with Texture

After the refinement step, the three corners of each 2D triangle in the mesh encode a disparity value, leading to a piece-wise linear disparity map. The next step of the proposed approach is to build a full 3D triangular mesh and assign textures to the triangles, so that view synthesis/interpolation reduces to a simple rendering problem (probably running on a GPU).

We assume that each vertex in the mesh is either related to a single object, presenting a continuous variation of disparity, or is located at the boundary between two objects, where discontinuities are allowed[2]. More precisely, let us consider the set of corners values $C(v) = \{d_1, d_2, ..., d_{N_c}\}$ related to a given vertex v. If a v is located at a disparity discontinuity, some of the triangles that share v will belong to the foreground (larger disparity) and others to the background (lower disparity), so that the difference between the maximum and minimum disparity values tends to be large. On the other hand, continuous regions tend to present low discrepancies in the disparity values (ideally they should be the same, but errors in the stereo matching process introduce noise). Based on these observations, a binary function $o(v)$ that identifies if v is classified as being in discontinuous disparity region is defined as

$$o(v) = \begin{cases} 0, & \text{if} \quad \max\{C(v)\} - \min\{C(v)\} < \mathcal{T}_d \\ 1, & \text{otherwise} \end{cases}, \tag{3.8}$$

where $\mathcal{T}_d$ is a disparity threshold. If $o(v) = 0$, all the corner values are set to the mean disparity value. Otherwise, they are clustered into two disparity values based on Otsu's adaptive threshold (OTSU, 1979), and the disparities of the corners at v are set to either one of the two cluster centers based on a nearest neighbor approach.

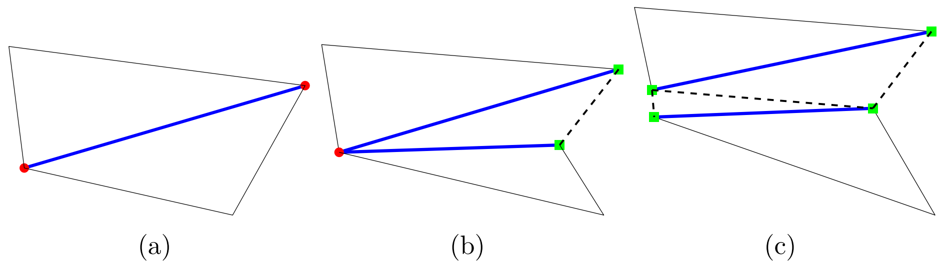
At the end of this process, all the corners at the same vertex either present the same disparity (we call it vertex type 1), or are clustered in exactly two disparity values (vertex type 2). We perform an exhaustive scan of all the edges of the triangular tessellation, and check the status at their vertices. If both of them are type 1, nothing is done, since the mesh is already continuous at both extremities. If one of the vertices is type 1 and the other type 2, we simply create another triangle (orthogonal to the image plane) by splitting vertex type 2 into two other vertices (such that the corners at each new vertex present the same disparity), and create a new edge between them. If both vertices are type 2, we split each of them into two new vertices (similarly to the previous case), and connect them with three new edges (generating two new triangles, both orthogonal to the image plane. Figs. 3.5(a)-(c) illustrate these three possibilities, respectively. In these figures, the thick blue edge is under consideration, a red dot indicates a type 1 vertex, and a green square a type 2 vertex. The dashed lines indicate the new inserted edges, so that new triangles are created.

The final step consists in assigning a texture for each triangle. For the regular triangles related to the initial 2D tessellation of the image domain, their texture can be

[2]Although we acknowledge that a vertex may be at the junction of three more objects as well, we do not deal with those situations

directly obtained from its projection onto the reference image. However, this process does not work for the orthogonal triangles that were created to complete the mesh, since they are not visible in the reference image. Fortunately, they are usually visible from the adjacent cameras, from where their texture can be retrieved. More precisely, we project these triangles to the closest cameras at the right and left of the reference camera, and extract the texture from the camera where the triangle is visible (it is visible only at the right or only at the left camera).

Figure 3.5 – Completion of the 3D mesh. The thick blue edge is under consideration, red dots indicate type 1 vertices, and green squares type 2 vertices. The dashed lines indicate the new inserted edges. (a) Type 1 - type 1 vertices (no triangle created); (b) Type 1 - type 2 vertices (one triangle created); (c) Type 2 - type 2 vertices (two triangles created).



3.6 Experimental Results

Table 3.1 – List of parameters. They were fixed for all the tests on this thesis, except for N_C which depends on the test.

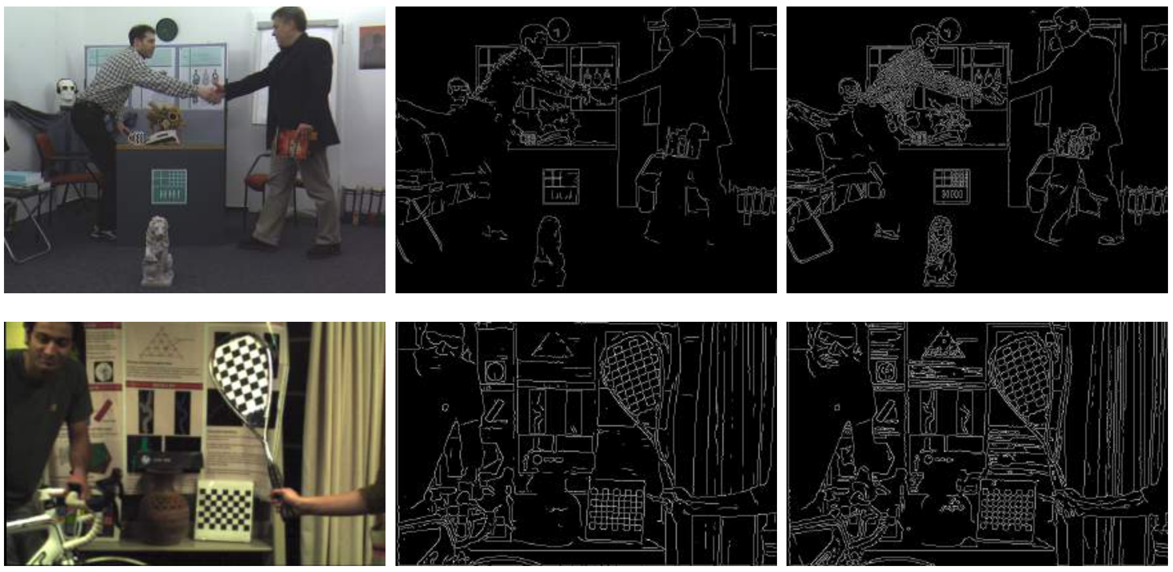
Parameter	Value	Description
N_C	–	Number of cameras. When not explicitly indicated on the test, it is used 5 cameras.
P_{min}	9	Points density on the image edges
g	6	Points density in borderless regions
γ_c	10	Decay of the color distance
$\mathcal{T}_d$	1.5	Disparity threshold to merge corners

This section will present the experimental results regarding the 3D mesh generation, which includes the segmentation of the image into triangular regions, the disparity map estimation and generation of the final textured mesh. The results regarding the use of this approach for view interpolation will be shown in Chapter 5. All the experiments were run using the parameters as in Table 3.1.

The first experiment is related to the edge detection algorithm used in our triangular domain segmentation. In Figure 3.6 we can see a comparison between the resulting edge maps from (CANNY, 1986) and (DOLLÁR; ZITNICK, 2013) for two different images. The Canny edge detector was able to retrieve some edges on the objects boundaries. However, the Dollár and Zitnick (2013) solution was able to distinguish better the textured areas from the objects boundaries.

The resulting meshes using both techniques can be seen in Figure 3.7. As a result of having less detected edges from Dollár and Zitnick (2013), the respective mesh presented a small number of triangles, which is good for alleviating matching ambiguities, but was not able to correctly model some objects boundaries. The resulting disparity map and view interpolation using both techniques were similar, so, for the sake of simplicity and runtime, we choose to use the Canny detector.

Figure 3.6 – Edge detector comparison, with the original images on the left, the edges computed using (DOLLÁR; ZITNICK, 2013) on the center, and on the right are the edges computed using the Canny operator.



For the single frame disparity map evaluation, we use the standard and well known Middlebury database (MIDDLEBURY, 2012), which can be seen on Figure 3.8. It contains sets of accurately rectified images taken with the same camera moving along a fixed baseline (i.e. exactly the same intrinsic camera parameters), with uniform illumination. Also, they provide the ground truth data for some of the views, allowing a quantitative evaluation of the estimated disparity maps.

Despite being very useful for quantitative evaluations and benchmarks, the Middlebury datasets are obtained with controlled and ideal conditions that are very hard to

Figure 3.7 – Mesh comparison using different edge detectors. On the top left and top right are the meshes computed using the edge detectors from (DOLLÁR; ZITNICK, 2013) and (CANNY, 1986), respectively. For better visualization, on the bottom of each top image it is a croped and zoom portion of them.

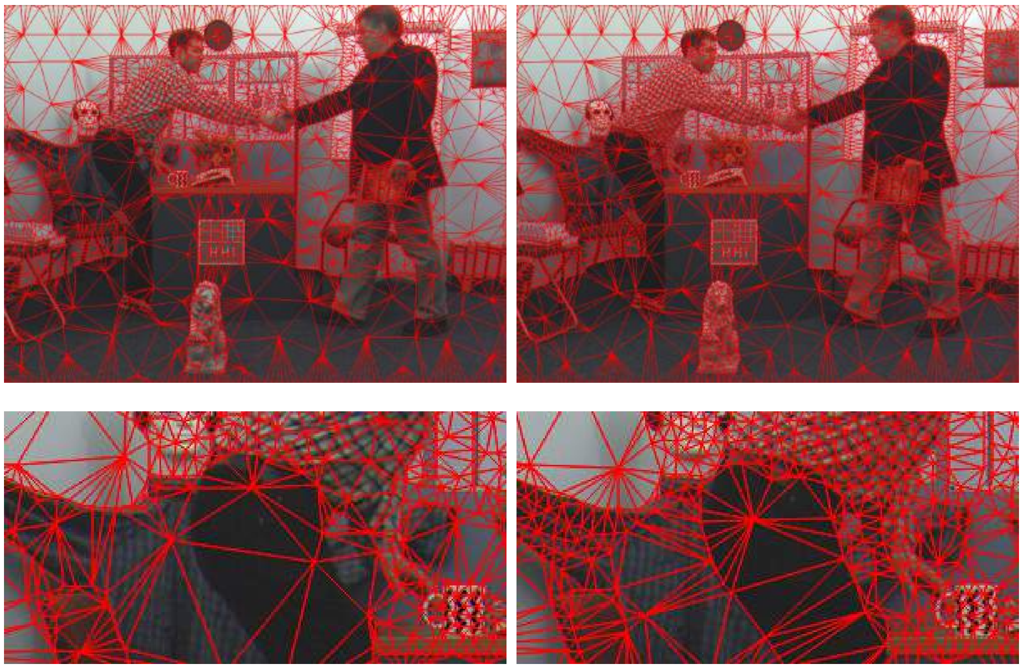
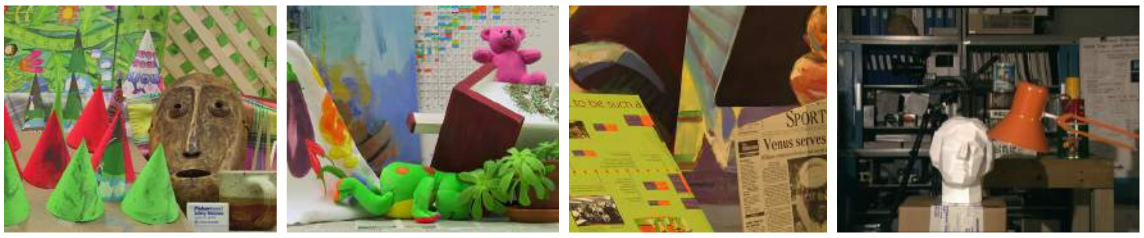


Figure 3.8 – Middlebury dataset.



reproduce in real applications. In most stereo/multiview capture systems, each image is acquired by a different camera, which may present slightly different intrinsic parameters. Also, the camera array may not be perfectly aligned, so that the corresponding images should be rectified before performing the matching procedure. In this work, we have also used a linear array of 20 cameras (called Herodion array[3]), and show that the proposed method performs well even in non-ideal conditions.

In the next experiment, we evaluate the effect of increasing the number of cam-

[3]The Herodion array generates a video stream from 22 synchronized uncompressed images operating at 30 Hz on a single PC (BAKER; TANGUAY, 2006). The intrinsic and extrinsic parameters of all cameras were obtained using the Caltech Camera Calibration Toolbox (<http://www.vision.caltech.edu/bouguetj/calib_doc/>), and images were rectified via software at the end.

eras on the quality of the disparity map. Figure 3.9 shows the disparity maps obtained for the `Teddy` dataset (Middlebury) using a varying number of cameras, along with the MSE (Mean Squared Error) values. As it can be observed, the disparity map with 5 cameras presents a smooth disparity variation within the same object while maintaining the boundaries between objects clear, and also presenting progressively a smaller MSE value as the number of views increase.

A similar tendency can be noticed for the Herodion array, as illustrated in Figure 3.10. In this example, we used 3, 5, 9, 13 and 17 cameras to produce the disparity map. With only three cameras, the areas with a periodic pattern, such as the checkerboard pattern on the tennis racket and along the curtains, are very difficult to correctly estimate the disparity since they produce several local maximum values at different disparity values. However, as we add more cameras, we get more redundant information and the quality in the disparity map improves dramatically in those problematic areas. Unfortunately ground truth is not available for the `Herodion` dataset, so that the MSE can not be computed.

Figure 3.9 – Disparity maps and MSE errors for the `Teddy` sequence (MIDDLEBURY, 2012) using different number of cameras (N_C) with resolution 450×375.

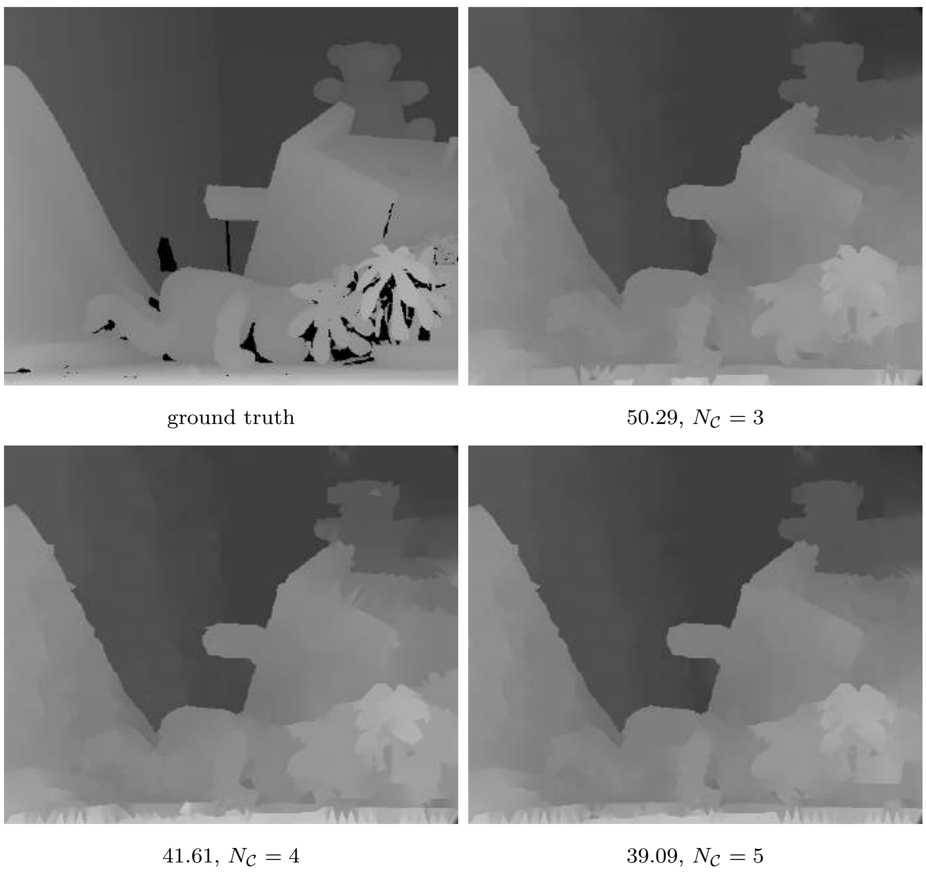
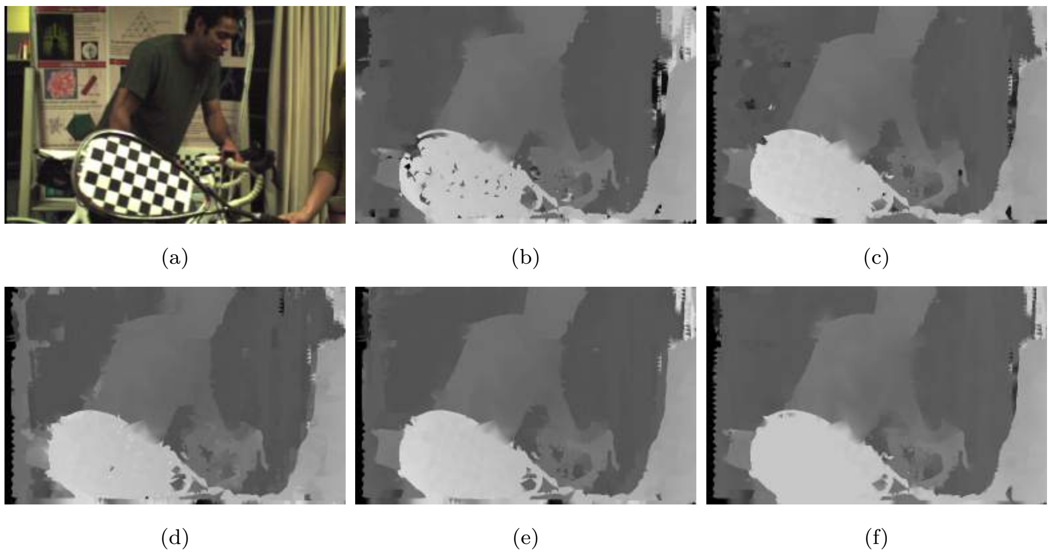


Figure 3.10 – Examples of disparity maps for the `Herodion` dataset. (a) Reference image (752×480 pixels) (b)-(e) Disparity maps using 3, 5, 9, 13 and 17 cameras, respectively.



Although the main goal of this thesis is the generation disparity maps that are suitable for view interpolation, we show in Figure 3.11 a qualitative (visual) and quantitative (MSE) comparison of our results with competitive approaches for some of the multiview sequences in the Middlebury database. We show a similar comparison for another multiview dataset acquired with the Herodion array, illustrated in Figure 3.12. Software implementations of (YANG, 2015)[4], (RHEMANN et al., 2011)[5], and (MOZEROV; WEIJER, 2015)[6] were provided by the authors, and the tests were performed using the default parameters as in their respective papers. Results from (ZITNICK et al., 2004) were retrieved from the Middlebury site.

We can observe in Figure 3.11 that the proposed approach does not outperforms its competitors quantitatively in the Middlebury datasets. Indeed, pixel-wise approaches have a better capability of preserving thin and curved objects than the proposed approach, since the triangulation procedure may not be accurate enough in such regions. It is important to note that our disparity maps could also be post-processed to achieve better results (in terms of MSE). For the sake of illustration, we filtered our disparity maps using a weighted median filter (ZHANG; XU; JIA, 2014), and Figure 3.13 shows that this simple approach can greatly improve the results (particularly in the defintion of the contours and small structures). However, this post-processing stage is applied in a pixel-wise manner,

[4]<http://www.cs.cityu.edu.hk/~qiyang/publications/cvpr-12/code/>
[5]<http://www.ims.tuwien.ac.at/research/costFilter/filterCode.zip>
[6]<http://www.cvc.uab.es/~mozerov/Stereo/>

destroying the domain triangulation. For that reason, we do not use any image-based post-processing in the remainder of this thesis.

However, it presents a more coherent result for the Herodion dataset: the results of both (YANG, 2015) and (RHEMANN et al., 2011) present discontinuities in the racket region and very bad disparity estimates at the bottom-left portion of the image. We believe that the quality degradation of the disparity maps produced by (YANG, 2015; RHEMANN et al., 2011) for poorly rectified images is due to the pixel-wise disparity computation, while the proposed approach is based on regions. And the disparity map from (MOZEROV; WEIJER, 2015) managed to present better results due to their energy minimization approach. Furthermore, it is important to emphasize that our main goal is view interpolation, and the generation of the disparity map is just an intermediate step. In Section 5.3.1 we show experimental results of view interpolation, and show that even though we do not produce the best disparity maps (in terms of MSE), our synthesized views present higher PSNR values than competitive approaches.

In Figure 3.14 we can see the impact of the resolution of the input images on the quality of the disparity map. As expected, by using images with higher resolution the 3D mesh is able to better represent the finer objects of the scene, which generates a better quality disparity map.

Finally, an illustration of the textured 3D mesh for the `Cones` dataset can be seen in Figure 3.15. More precisely, this figure was generated by simply rendering the 3D mesh with a synthetic camera at two different locations (and not aligned with the baseline of the cameras) from the original view. As expected, a novel view at any desired position can be achieved by simply using the textured 3D mesh of a single view. It can be observed that the use of orthogonal triangles to fill the mesh (images on the right) indeed generate views without holes, despite the creation of some artifacts (due to the fact that depth discontinuities are filled by "flat" regions). More examples of 3D meshes for the `Teddy` and `Baby1` datasets can be seen in Figures 3.16 and 3.17. The combined use textured 3D meshes relatives to different reference cameras to produce interpolated views will be tackled in Chapter 5.

The proposed approach was implemented in C++, and code was not optimized. All the tests were run on a Core i5 2.2GHz processor (notebook version), 3GB RAM and Windows 7 operating system. The triangular tessellation takes about 0.1s for a typical 450×375 image (Middlebury). The stereo matching with refinement takes 0.5s using 3 cameras (reference + left + right) and 0.95s using 9 cameras in our C++ code, so that adding more cameras has little impact on execution time. Using 3 cameras and the same disparity range (30 pixels) for 752×480 images, the processing time linearly increases to 1.35s (2.29× slower using 2.14× more pixels) with a small memory increase, since most of the memory is allocated to the matching costs. The 3D mesh generation step takes 0.1s.

Using the code provided by the authors, the average execution time of (YANG, 2015) for a 450 × 375 image was about 0.8s. Using our own partial implementation of (ZITNICK et al., 2004), the run time for the initial disparity map and refinement process is approximately 40s. The runtime of the C++ implementation of (MOZEROV; WEIJER, 2015) provided by the authors is approximately 90s for the Middlebury dataset. The MATLAB implementation of (RHEMANN et al., 2011) provided by the authors has a running time of approximately 500s, but they mention an execution time of 0.065s using their CUDA implementation on a high end GPU.

3.7 Chapter Conclusions

This chapter presented all the steps needed to generate a textured 3D mesh for every rectified view (camera) along a linear array. The first step consists in segmenting the reference image into triangular regions by densely populating vertices along the images edges, since the objects boundaries are usually related to the image edges. The remaining vertices are distributed within the image according to the Scale Map, which measures the local image complexity. With those vertices, a Delaunay triangulation is performed, and the triangular regions are found.

Afterwards, a disparity value is associated to each triangle is obtained by using a region matching technique. This piece-wise disparity is then refined to achieve sub-pixel accuracy and generate a smoother disparity map, by analyzing the color similarity of adjacent triangles. Finally, the mesh is closed by considering the disparity differences at the vertices, and texture is assigned based on the content of the reference region and its neighboring views.

Figure 3.11 – Top row: ground truth. Other rows: disparity maps with corresponding MSE using (YANG, 2015), (RHEMANN et al., 2011), (ZITNICK et al., 2004), (MOZEROV; WEIJER, 2015) and the proposed approach, respectively. We used 9 cameras (camera 0 through 8) for the `Cones` (450×375 pixels), `Venus` (434×383 pixels) and `Teddy` (450×375 pixels) datasets and all 5 cameras for the `Tsukuba` (384×288 pixels)) dataset.

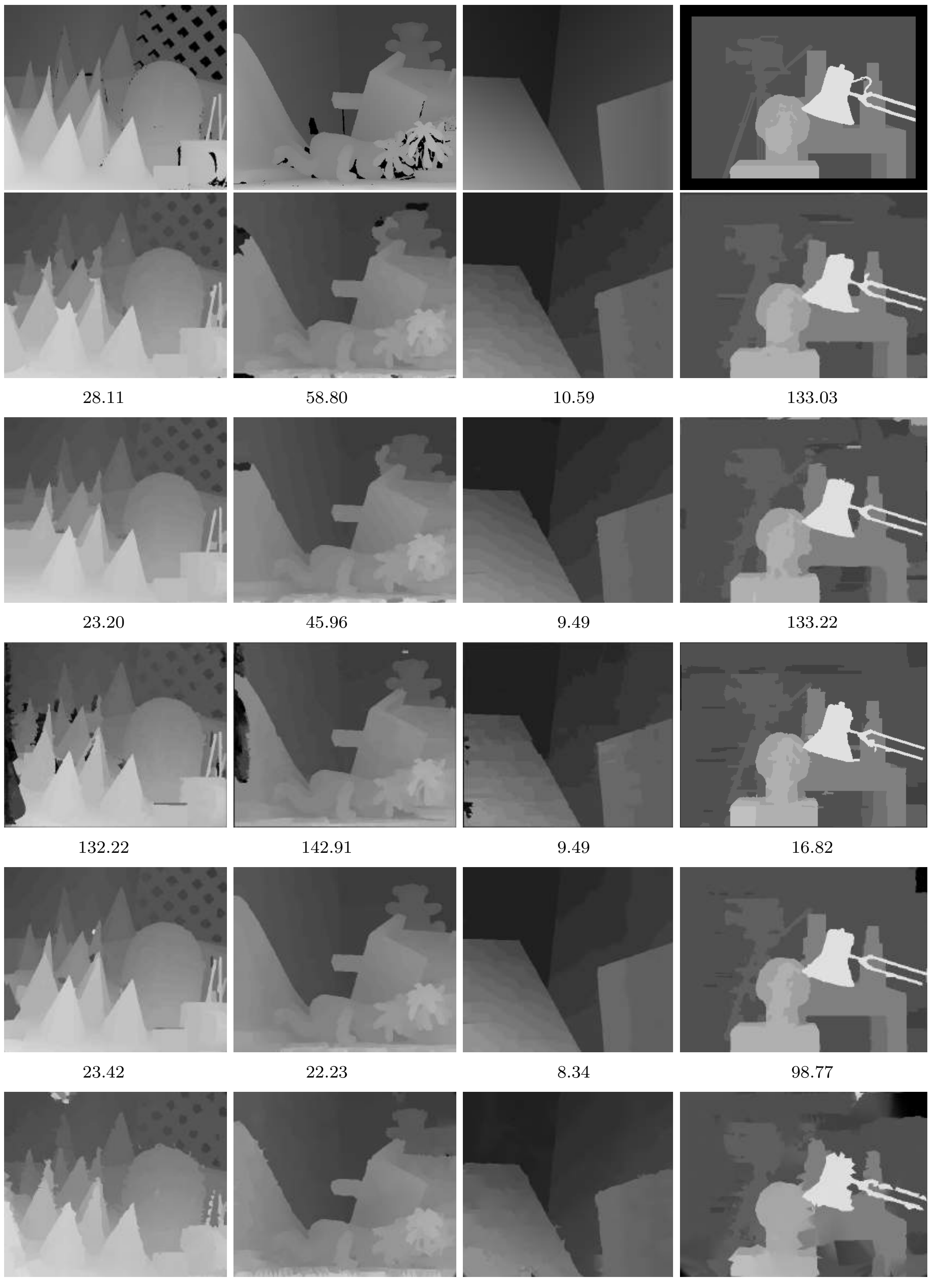


Figure 3.12 – (a) Reference image. (b)-(e) Estimated disparity maps using Yang's, Rhemman's, Mozerov's and the proposed approaches using 17 cameras, respectively.

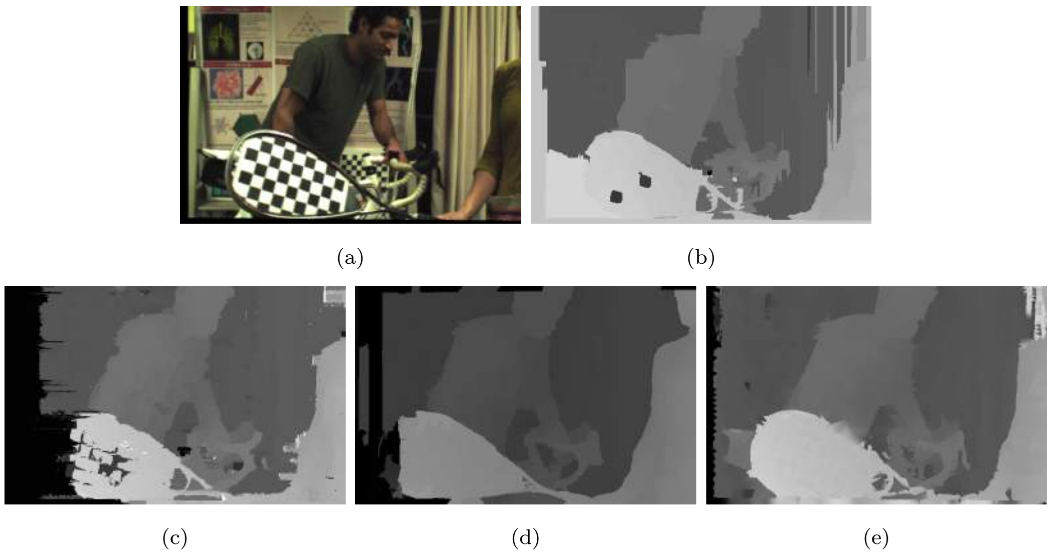


Figure 3.13 – Comparison between disparity maps with and without post-processing. On top row are the disparity map without post processing, and the bottom using the weighted median filter post processing.

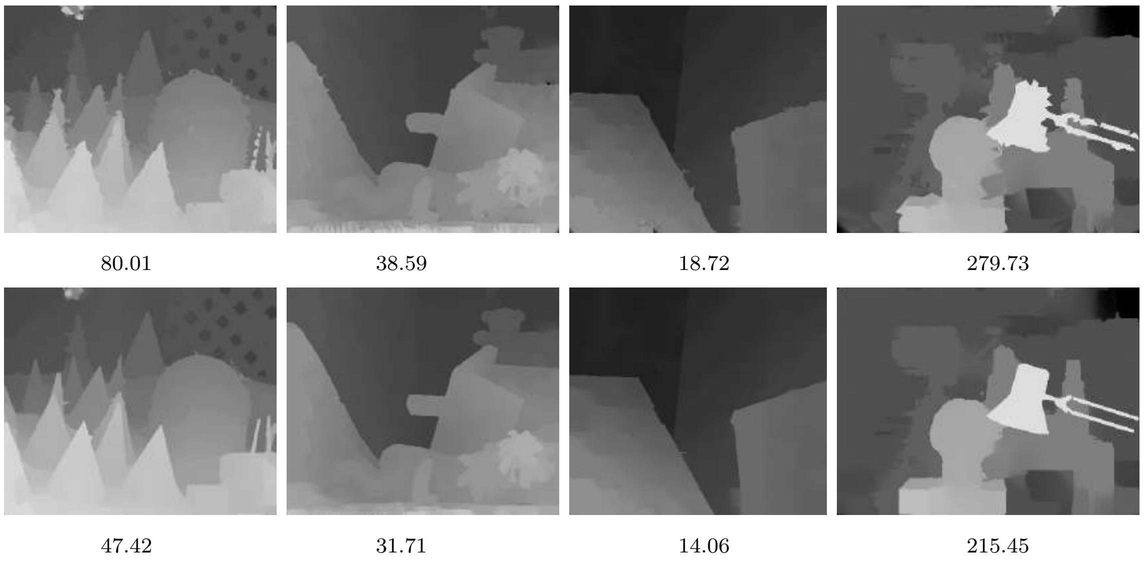


Figure 3.14 – (a) Reference image. (b) Disparity ground truth. (c) Estimated disparity for the 450×350 pixels dataset. (d) Estimated disparity for the 1800×1500 pixels dataset.

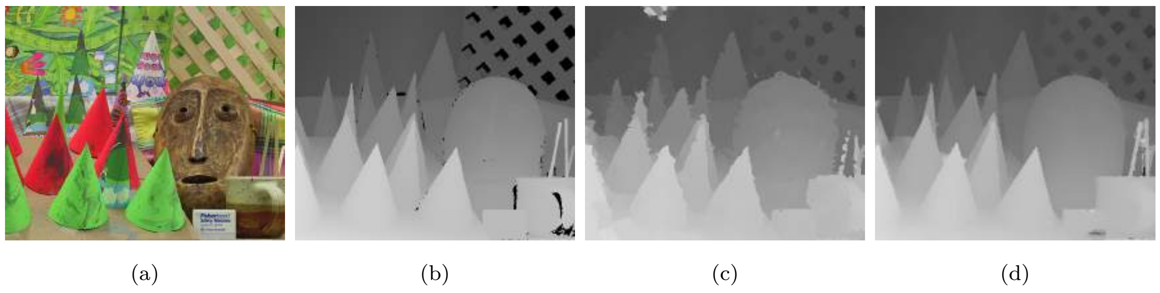


Figure 3.15 – 3D textured mesh for the `Cones` dataset rendered using two different viewpoints for the synthetic camera (top and bottom). On the left column are the mesh rendered without orthogonal triangle, and on the right column are the meshes rendered with the orthogonal triangles.

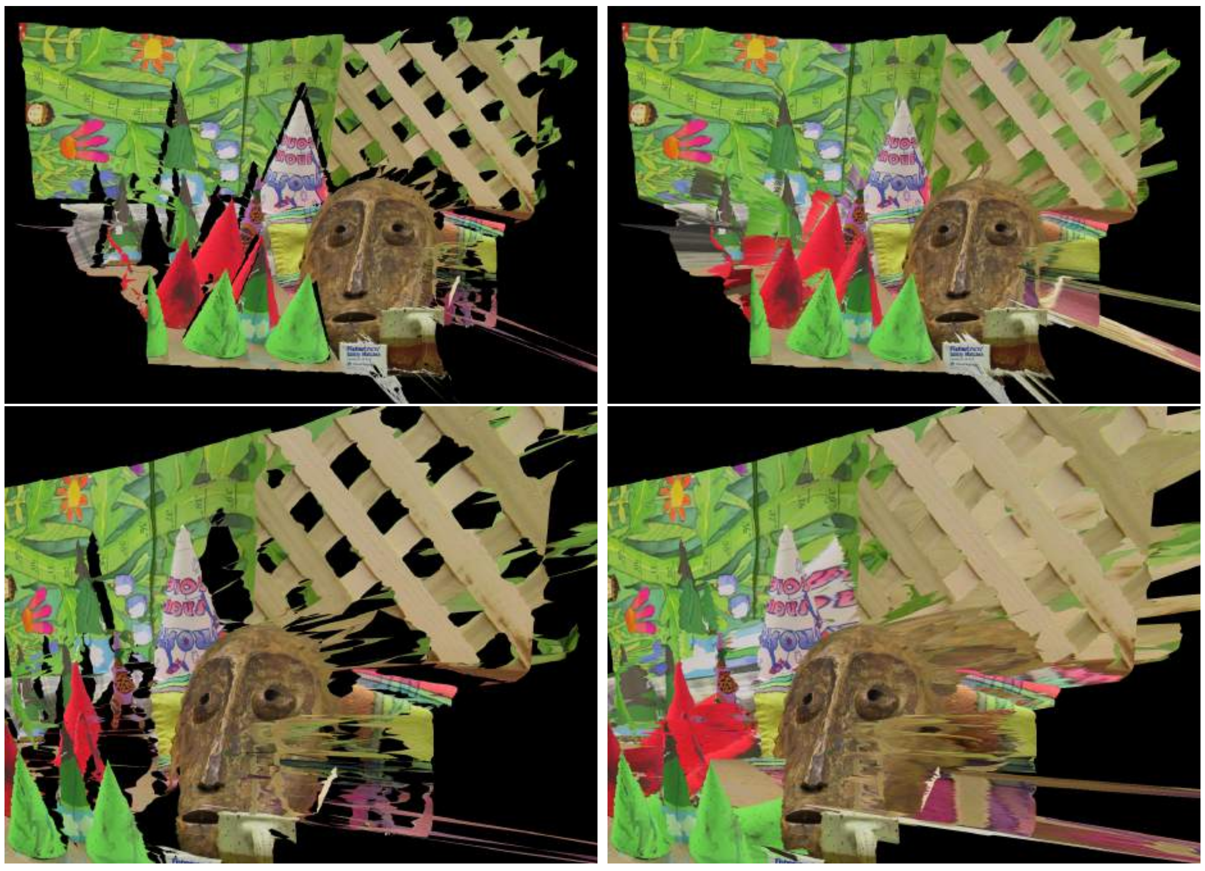


Figure 3.16 – 3D textured mesh for the `Teddy` dataset rendered using two different viewpoints for the synthetic camera (top and bottom). On the left column are the mesh rendered without orthogonal triangle, and on the right column are the meshes rendered with the orthogonal triangles.

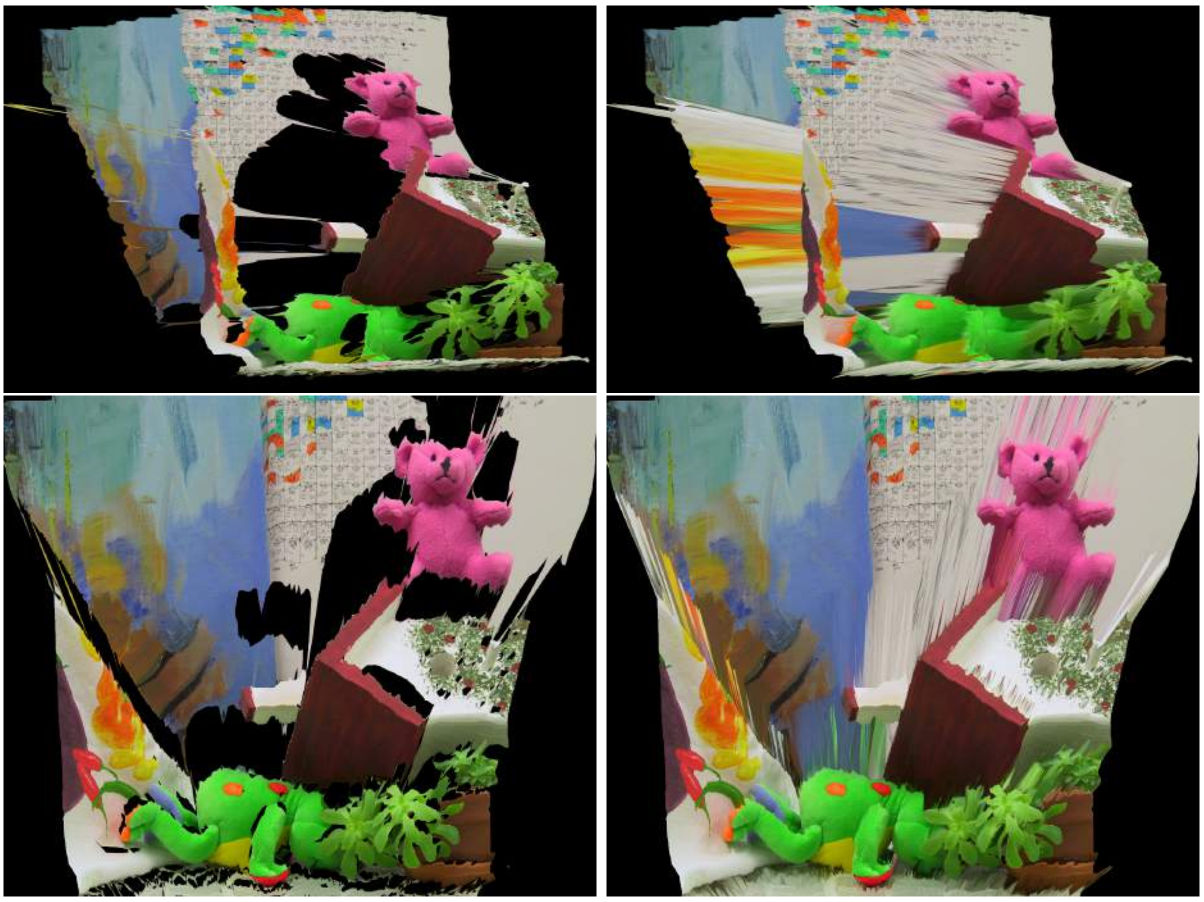
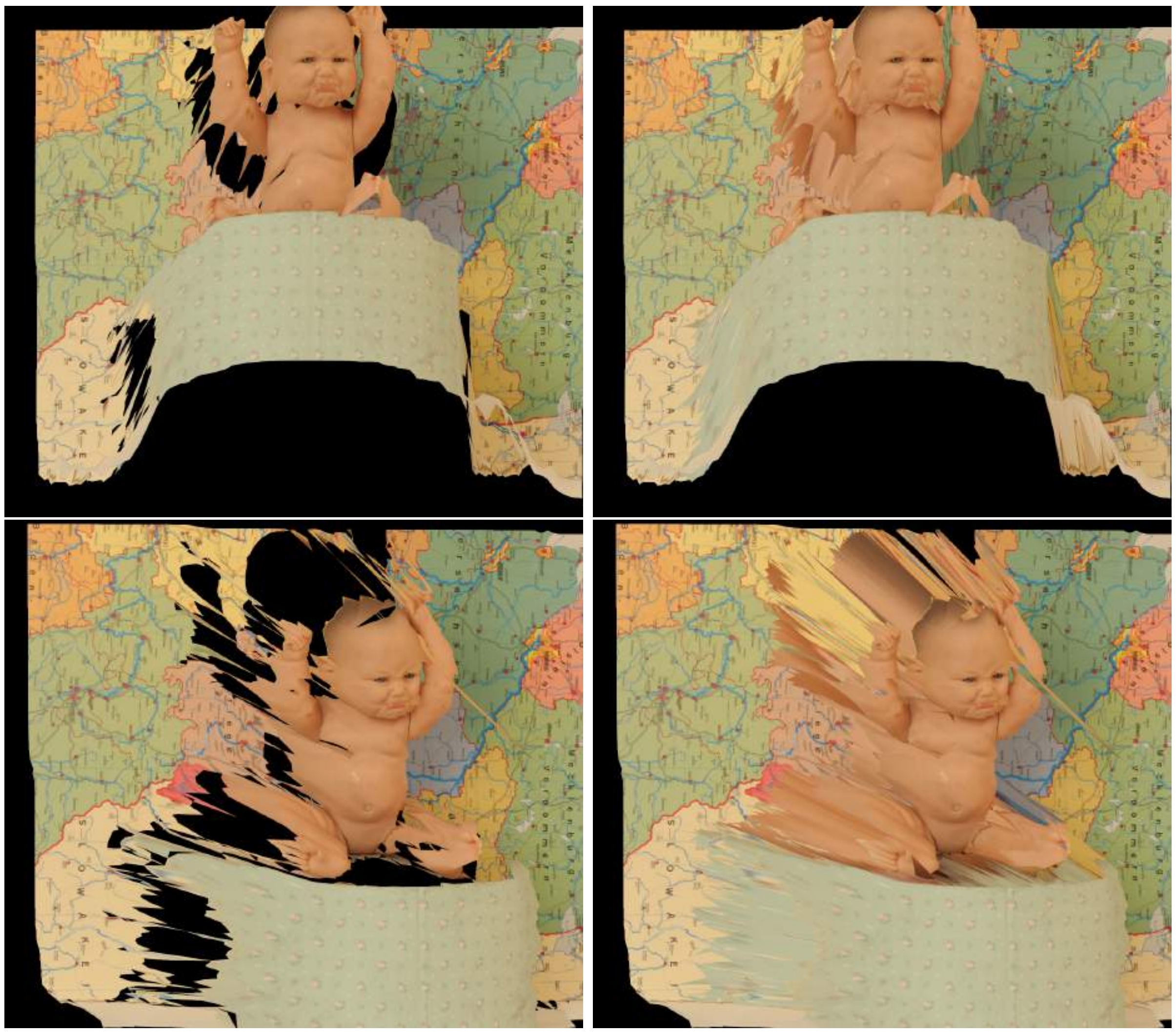


Figure 3.17 – 3D textured mesh for the `Baby1` dataset (from Middlebury) rendered using two different viewpoints for the synthetic camera (top and bottom). On the left column are the mesh rendered without orthogonal triangle, and on the right column are the meshes rendered with the orthogonal triangles.



4 TEMPORAL UPDATE OF THE 3D MESH

In the last chapter, a technique to generate a textured 3D mesh that models the scene that can be use for view interpolation process was proposed. However, this approach is not suitable for videos, since a new 3D mesh must be created for each new frame, which tends to present flickering in the interpolated video (as in most frame-by-frame methods). In fact, imposing temporal coherence to methods based on a domain triangulation is not trivial, since the mesh should be updated in time, including the addition and removal of triangles.

One key aspect in imposing temporal coherence is to identify which pixels/regions are “related” at different frames. In this work, the correspondence of triangular regions along time is achieved by dynamically adapting the 3D mesh, instead of generating a new one for each frame. This ensures that the shape of the objects will be coherent, also allowing the computation of a temporally coherent disparity map, which will generate an interpolated view with fewer temporal artifacts.

An overview of the complete pipeline of the proposed approach is presented in Algorithm 1, with each step indicated with their respective sections. The first step consists in deforming the mesh from the previous frame by first moving, and then removing and adding vertices. The removal and adition processes aim to model the occlusion and disocclusion that occurs on the scene. With the mesh updated, the disparity is calculated exactly as in the previous chapter, by region matching and then aggregating the costs. On the current step we have a piece-wise disparity, since the disparity refinement process is not yet done. The disparity of each time-persistent triangle is then calculated using an HMM-like procedure that uses the past disparity information. A propagation scheme (Disparity Filtering) is used to improve the initial estimative of new or recent triangles, and the refinement process is done. And finally a subpixel filtering process based on the Kalman Filter is used to generate the disparity map.

4.1 Triangular Mesh Deformation

The proposed approach consists in first updating the 2D domain triangulation using only three vertex-based operations: translation, deletion and addition. Moving vertices relates to objects that are either moving or deforming; deletion removes vertices in areas that are being occluded; and adding vertices is related to disocclusions. Those

```
for every new frame $f$ do
    Deform previous mesh (Sections 4.1.1 and 4.1.2);
    Add vertices (Section 4.1.3);
    Region Matching (Section 3.2);
    Cost Aggregation (Section 3.3);
    HMM Disparity (Section 4.2.1);
    Disparity Filtering (Section 4.2.2);
    Disparity Refinement (Section 3.4);
    Kalman Disparity (Section 4.2.3);
    3D Mesh Creation (Section 3.5);
end
```

Algorithm 1: Pipeline of the proposed technique.

operations are performed according to Algorithm 1 and will be explained next.

4.1.1 Moving Vertices

Since our domain triangulation approach is guided by the vertices, motion/deformation of triangles will be based on vertex displacements, which can be estimated using a particle tracker. One particular characteristic of the adopted triangulation scheme is that some (or several) edges are expected to lie along image edges. In those cases, neither color nor gradient are constant along consecutive frames, which are characteristics that several state of the art particle trackers (e.g. Sundaram, Brox and Keutzer (2010), Sand and Teller (2008)) explore to identify "good" particles to track. In fact, such trackers tend to discard particles along edges, since they are unreliable. In our application, however, it is crucial to keep (and track) particles along image edges, since they may potentially relate to disparity discontinuities, which should be detected and kept.

In this work, we used a modified version of the tracker from Sundaram, Brox and Keutzer (2010). Such tracker is simple to compute but produces good results in practical video sequences when the optical flow is accurate, and our modification adds disparity information to guide the tracking process. The first step consists in computing the optical flow using Brox et al. (2004), which is fast and accurate. Then, a binary mask $M_t(p)$ that indicates if pixel p presents a "reliable" optical flow value is computed by cross-checking the direct and reverse flows as well as color similarity:

$$M_t(p) = \begin{cases} 1, & \text{if } \left\| \boldsymbol{u}_{t-1}^{t}(\boldsymbol{p}) + \boldsymbol{u}_{t}^{t-1}\left(\boldsymbol{p} + \boldsymbol{u}_{t-1}^{t}(\boldsymbol{p})\right) \right\| > \mathcal{T}_o \\ 1, & \text{if } \left\| \boldsymbol{I}_{t-1}(\boldsymbol{p}) - \boldsymbol{I}_t\left(\boldsymbol{p} + \boldsymbol{u}_{t-1}^{t}(\boldsymbol{p})\right) \right\| > \mathcal{T}_c \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

where $\boldsymbol{p}$ is the location of pixel p. Here, $\boldsymbol{u}_{t-1}^{t}(\boldsymbol{p})$ is the (direct) optical flow from frame $t-1$ to t at location $\boldsymbol{p}$, $\boldsymbol{u}_{t}^{t-1}(\boldsymbol{p})$ is the (reverse) optical flow from frame t to $t-1$, and $\mathcal{T}_o$ is a distance threshold experimentally set to 0.25. Also, $\boldsymbol{I}_t(\boldsymbol{p})$ is the RGB color vector of $\boldsymbol{p}$ at frame t, and $\mathcal{T}_c = 20$ is a color threshold, defined experimentally. To account for uncertainties in the cross-checking process, a morphological dilation operator with is a circular structural element (radius experimentally set to 3) is applied to $M_t(p)$, obtaining an over estimation of the optical flow errors. Since it is better to classify a good vertice as bad than the opposite, this dilation operation produces and over-estimation of bad pixels, which reduces the risk of propagating a bad vertex that would generate a bad quality mesh.

Besides the optical flow, the binary function $o(v)$ defined in Equation (3.8) indicates if each vertex of the 3D mesh lies on a disparity discontinuity or in a smooth region. This information is important, since objects that share a boundary (i.e. lie in a discontinuity) may have distinct motion patterns, which intrinsically affects the optical flow. The final step consists in simply updating the position of each vertex v from frame $t-1$ to frame t based on the optical flow and the "reliability" of each tracked vertex. More precisely, the update is given by

$$\boldsymbol{v}_t = \boldsymbol{v}_{t-1} + \boldsymbol{u}_{t-1}^{t}(\hat{\boldsymbol{v}}_{t-1}), \tag{4.2}$$

where $\boldsymbol{v}_t$ is location of vertex v at frame t. The value of $\hat{\boldsymbol{v}}_{t-1}$ depends on the tracking status of v at frame $t-1$: if it was considered as reliably tracked, then $\hat{\boldsymbol{v}}_{t-1} = \boldsymbol{v}_{t-1}$, i.e., the location of v at frame $t-1$; otherwise, the location of a neighboring vertex in the mesh that was reliably tracked and presents the closest disparity is used. More precisely, $\hat{\boldsymbol{v}}_{t-1}$ is the location of a vertex w at frame $t-1$ given by

$$w = \begin{cases} v, & \text{if } o_{t-1}(v) = 0 \wedge M_t(v_i) = 0 \\ \underset{s \in V(v)}{\operatorname{argmin}} \left\{ |d_{t-1}(s) - d_{t-1}(v)| \right\}, & \text{otherwise} \end{cases}, \tag{4.3}$$

where $d_{t-1}(s)$ denotes the *foreground* disparity of vertex s at frame $t-1$. Recall that at disparity discontinuities ($o(s) = 1$), there are two disparity values associated with the 2D vertex s: in this case, the foreground disparity used in Equation (4.3) is the largest value. Also, the set $V(v)$ is defined as

$$V(v) = \{w \in \mathcal{V}_{t-1}(v) \mid M_t(w) = 0 \land o_{t-1}(w) = 0\}, \tag{4.4}$$

with $\mathcal{V}_{t-1}(v)$ being the set of all vertices that share an edge with v at frame $t-1$. It is important to note that typical particle trackers (e.g. Sundaram, Brox and Keutzer (2010)) tend to remove/ignore particles along image edges, since neither color nor gradient tend to remain stable in time along the boundaries of moving object. With our modification, however, vertices lying along disparity discontinuities are still tracked, inheriting the motion from reliably tracked neighbors. To avoid drifting of such vertices along time (since their actual motion is typically not the same as the neighbors), an additional refinement step is applied at vertices v for which $o_{t-1}(v) = 1$: the Canny edge map is computed for frame t, and these vertices are attached to the closest image edge within a circular region with radius $\mathcal{T}_e$ (set to 8 pixels based on experiments), since disparity discontinuities tend to arise at image edges.

The motion of all vertex locations from frame $t-1$ to t intrinsically leads to an updated mesh, considering the same edges in the triangulation. Moving vertices (and triangles) can correctly maintain the triangulation of a static background as well of moving or deforming objects. However, problems may appear in occlusions/disocclusions, since some vertices need to be deleted or created, as explained next.

4.1.2 Deleting Vertices

Occlusions are common in video sequences, and they are caused either by the movement of objects or the camera (or both). Since occluded regions are no longer visible by the camera, the corresponding vertices/triangles should be deleted/updated. This deletion process begins by firstly identifying the vertices (particles) that are likely to be within an occluded region, and then removing them along with their connected edges in the mesh.

Typically, an occluded particle can be identified by significant color variation in time, or inconsistent optical flow cross-check, which could be detected by Equation (4.1). However, vertices along image edges, which are important for the 3D mesh, also tend to be detected as occluded using Equation (4.1). On the other hand, the vertex update strategy summarized by Equations (4.2) and (4.3) works well for tracking particles along image edges, but may also propagate an occluded vertex, since it might be guided by a neighboring non-occluded vertex. The proposed method for identifying occluded vertices considers jointly the results of Equations (4.1), (4.2) and (4.3). More precisely, the set O_t of occluded vertices from frames $t-1$ to t is given by

$$O_t = \{v | o(v) = 0 \wedge (M_t(v) = 1 \vee \|\boldsymbol{I}_t(\boldsymbol{v}_t) - \boldsymbol{I}_{t-1}(\boldsymbol{v}_{t-1})\| > \mathcal{T}_c)\}, \tag{4.5}$$

where $\boldsymbol{I}_{t-1}(\boldsymbol{v}_{t-1})$ and $\boldsymbol{I}_t(\boldsymbol{v}_t)$ are the RGB color values of vertex v at frames $t-1$ and t, respectively, $\mathcal{T}_c$ is the same color threshold used in Equation (4.1). In other words, vertices that are not located at a disparity discontinuity and either change their color above a given threshold or have an inconsistent optical flow cross-check value are detected as occluded, and they are deleted from the mesh (as well as all the triangles that are connected to it).

Another criterion for deleting vertices is based on vertex distances: due to temporal motion, two (or three) neighboring vertices of the same triangle may get too close. In these cases, the corresponding triangles tend to be very small (in area) or highly deformed, which is not adequate for the region-based stereo matching process. To cope with this situation, for every pair of vertices that are close enough given a threshold (experimentally set to $0.6P_{min}$ pixels), we remove the one that does not lie at a disparity discontinuity. In case neither of them lies at a disparity discontinuity, the one with the smaller disparity is removed, aiming to maintain the shape of foreground objects.

4.1.3 Adding Vertices and Retriangulating the Mesh

After translating and removing vertices, some regions may contain a low density of vertices (mostly due to disocclusions). To cope with this issue, the set of vertices that remained from the previous frame are kept, and the scale map (used for obtaining the vertices in the first frame, as explained in Section 3.1) is explored to add vertices based on the local image complexity.

Since the main goal of temporal coherence is to keep track of triangles in time, the new triangulation should try to keep the triangles in the previous frame for which all three vertices remained in the current frame. However, some triangles may become very thin (elongated) due to the motion of vertices. To avoid these irregular triangles, the well-known compactness measure (BOGAERT et al., 2000)

$$C(r) = \frac{perimeter^2(r)}{area(r)} \tag{4.6}$$

is computed for each triangle r with three correctly tracked vertices. This value is minimal for an equilateral (ideal) triangle, so that triangles for which $C(r)$ is above a given threshold (set to 2 in this work) are considered irregular, and the corresponding edges are removed from the triangulation. Finally, a constrained Dalaunay triangulation (CHEW, 1987) with the current vertices and remaining edges is applied, leading to the final 2D domain triangulation in the current frame.

An example of the mesh deformation scheme for a synthetic video sequence is shown in Figure 4.1. Figure 4.2(a) shows the 2D mesh at frame $t-1$, highlighting some vertices (in black) that are occluded at frame t. Figure 4.2(a) shows some added vertices (blue) at frame t, as well as some thin elongated triangles due to vertex motion. The final 2D mesh is shown in Figure 4.2(c). In this example it can be seen the importance of the use of Equation 4.6 in order to avoid a triangulation with very deformed triangles.

Figure 4.2 illustrates an example comparing the meshes in two consecutive frames using the single-frame approach from Chapter 3 and the proposed 3D mesh temporal update scheme for a real, multiview sequence (`Hall2` dataset). Even for two consecutive frames with little change on the scene, the triangulation from the single-frame approach present considerable changes on the triangulation. However, the proposed approach was able to maintain most of the previous triangles, even along the object boundaries where the optical flow tends to fail. More results (including videos) of the temporal evolution of meshes can be seen on the authors webpage at <http://inf.ufrgs.br/~gpfickel/phdthesis>.

4.2 Temporal Coherence of the Disparity Maps

In order to enforce a smooth variation of the disparity value of a given triangle tracked in time, we propose the following procedure: i) first to use a Hidden Markov Model (HMM)-like approach to enforce the disparity coherence on the initial disparity estimation of each triangle; ii) then, apply a spatial filter within the disparity values to propagate the temporal information of each triangle to their neighbors; and finally iii) apply a Kalman filter on each vertex of every triangle to enforce spatio-temporal continuity along the vertices of the 3D mesh.

Figure 4.1 – Example of the mesh deformation. a) Mesh at frame $t-1$, black dots indicate vertices to be occluded in the next frame. b) Mesh at frame t based on vertex displacement, removal and addition (blue dots). c) Final 2D mesh after removing thin triangles.

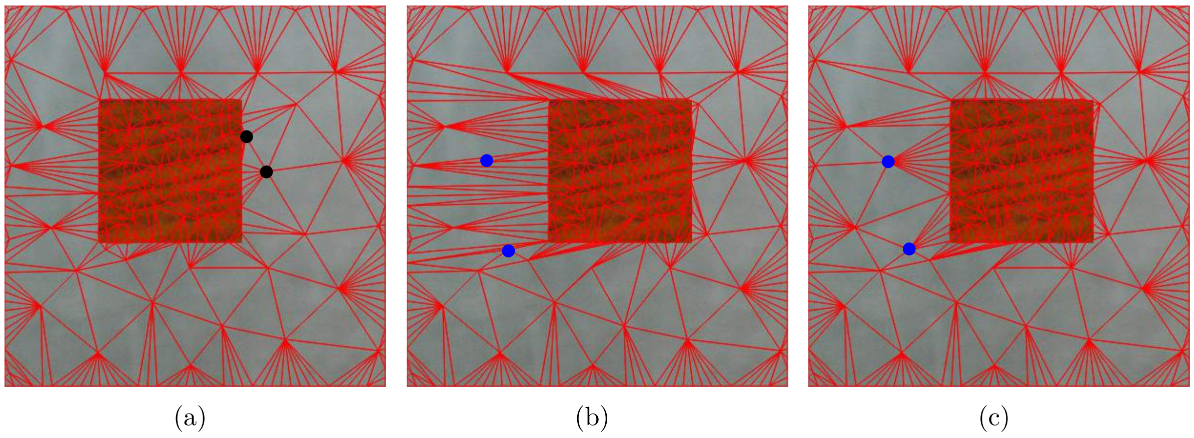
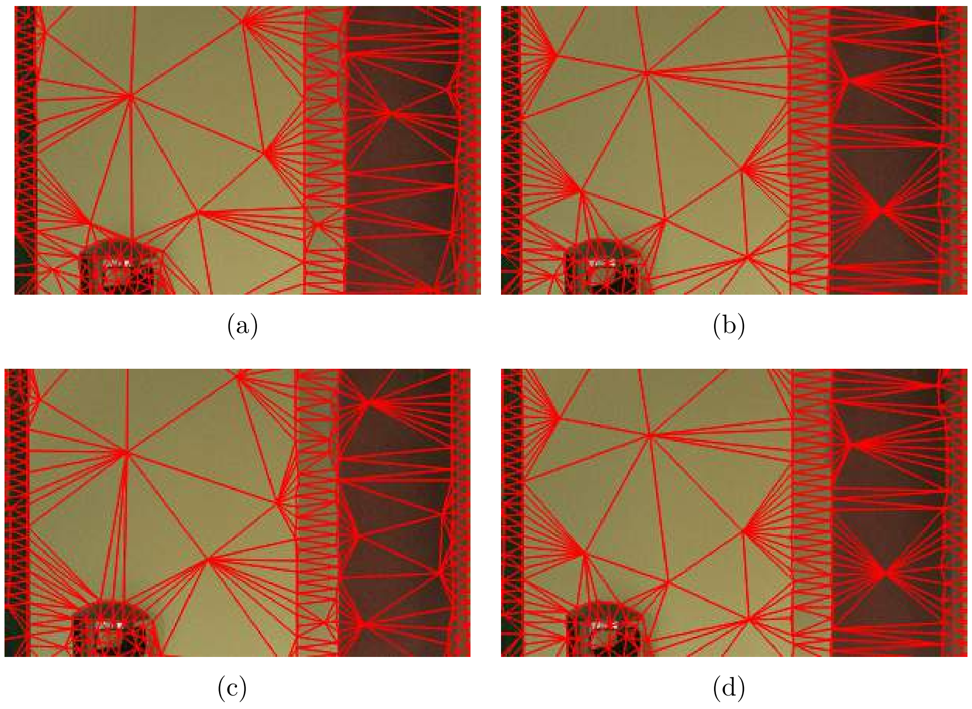


Figure 4.2 – Example of meshes within consecutive frames using the `Hall2` dataset. The left column shows the meshes using the single-frame approach from Chapter 3, and the right illustrates the meshes using the proposed temporal updating 3D mesh. The meshes on top row are generated from frame 15, and the bottom row using frame 16.



4.2.1 HMM Disparity Estimation

In the (intermediate) piece-wise representation of the disparity map, each triangle presents a constant (integer) disparity value based on the aggregated matching cost m''_d provided by Equation (3.5). If a triangle r persists in time, such disparities can be considered observations o_t at frame t, and Bayesian filters can be used to infer the actual disparities d_t. Among the several possibilities, Hidden Markov Models (HMMs) seem a good choice, since they can handle both adherence to observed values and temporal continuity at the same time in a simple manner.

For a given triangular region r that exists for T frames, with a sequence of observed values $O = o_1\ o_2 \cdots o_T$, the goal is to find the sequence of estimated disparities (states) $D = d_1\ d_2 \cdots d_T$ that maximize the posterior probability

$$P(D|O) = P(d_1)P(o_1|d_1)\prod_{t=2}^{T} P(d_t|d_{t-1})P(o_t|d_t), \tag{4.7}$$

where $d_t \in \{d_{min}, ..., d_{max}\}$ is the state at frame t (d_{min} and d_{max} are the minimum and maximum disparity values in the search range), $P(d_t|d_{t-1})$ are the transition probabilities and $P(o_t|d_t)$ is the observables distribution per state.

To enforce temporal coherence, the transition probabilities should penalize large disparity changes in adjacent frames. A simple way to enforce this behavior is to choose a discrete Gaussian probability

$$P(d_t|d_{t-1}) = \frac{1}{Z}\exp\left(-\frac{(d_t - d_{t-1})^2}{2\sigma_d^2}\right), \tag{4.8}$$

where Z is a normalization factor and σ_d controls how tight temporal consistency is. This term is defined experimentally as $\sigma_d = 0.05N_s$, with $N_s = d_{max} - d_{min} + 1$ being the disparity search range used in the matching process (which is equal to the number of states).

The model should also allow larger disparity changes when there is fast variation in the depth axis (e.g. an object getting closer to the camera). In the HMM framework, that can be achieved by choosing an adequate observable probability model $P(o_t|d_t)$. Note that when the matching costs $m''(d)$ used to obtain the observables o_t present a sharp peak, it means that the observed value o_t presents a better match than all other possible disparities, resulting a reliable match (i.e., the actual disparity d_t tends to be close to o_t). On the other hand, if $m''(d)$ is approximately flat, it means that all possible disparities present similar match values, and observed disparity o_t is typically not reliable.

Also, the optimal state sequence D can be computed with $\mathcal{O}(TN_s^2)$ computations using the Viterbi algorithm (RABINER, 1989). This algorithm, based on dynamic programming, requires the knowledge of all state transition probabilities $P(d_t|d_{t-1})$, given by Equation (4.8), and also the evaluation of $P(o_t|d_t)$ for the observed values o_t $(t = 1, ..., T)$. Hence, in practice, the full observables distribution is not needed, and we propose to use $P(o_t|d_t) = f(d_t) = m''(d_t)$, so that $P(o_t|d_t)$ is large for $d_t = o_t$ when o_t is obtained with high confidence (which allows fast state transitions) and lower otherwise.

4.2.2 Disparity Filtering

The proposed HMM-based approach indeed reduces temporal flickering in the piece-wise disparity map for triangles that persist in time. However, the triangular mesh deformation described in Section 4.1 also adds triangles, which have no temporal persistence when they are created. Since it is a fair assumption that triangles that exist along several frames present a more accurate disparity value due to the temporal cohesion process, we propose to apply a spatial filtering within the regions to propagate the temporal information to these recently created regions. The filtered disparity $d'(r)$ of a given triangle r is given by

$$d'(r) = \frac{\sum_{s \in \mathcal{R}(r)} d(s) w_b(r, s)}{\sum_{s \in \mathcal{R}} w_b(r, s)}, \tag{4.9}$$

where $\mathcal{R}(r) = \{s \mid \gamma_{rs} \leq \mathcal{T}_s\}$ is a neighborhood of r based on a distance threshold $\mathcal{T}_s$ (set experimentally to 128), $d(s)$ is the unfiltered disparity of a region $s \in \mathcal{R}(r)$. The term $w_b(r, s)$ is the filtering weight given by

$$w_b(r, s) = \exp\left(-\frac{\delta_{rs}}{\sigma_c} - \frac{\gamma_{rs}}{\sigma_d} - \frac{\sigma_a}{N_f(s)}\right), \tag{4.10}$$

recalling that δ_{rs} and γ_{rs} are color and spatial distances, as used in Equation 3.4. Also, $N_f(s)$ is the age of s (number of frames where s existed), and σ_c, σ_d and σ_a controls the decay of those terms (set experimentally to $\sigma_c = 10$, $\sigma_d = 25$, $\sigma_a = 5$). Although $\mathcal{R}(r)$ could be based on the MST, as the aggregation step described in Section 3.3, we noticed that smaller neighborhoods suffice for the filtering step. It is important to note that it is easy to obtain $\mathcal{R}(r)$ based on the data structure that stores the mesh.

By using this filtering process, we propagate the disparity information of time-persisting triangles to newly created ones within their neighborhoods. This process ensures that even in highly dynamic scenes where several triangular regions are removed and added, the temporal coherence procedure can improve the disparity estimation of all the image, including recently created triangles.

4.2.3 Kalman Filter Disparity Estimation

As shown in Algorithm 1, a refinement process is applied after the Disparity Filtering in order to achieve a smoother disparity map with sub-pixel representation. As described in Section 3.4, such process adjusts the disparity values at the corners of the 2D mesh so that similar adjacent triangles should have similar disparity values in the corners of the connecting vertices, leading to a piece-wise linear representation of the disparity map. However, the refinement process applied independently for each frame may not lead to a temporally coherent disparity map, so we add another stage to enforce temporal smoothness on the corners of each triangle.

Since the disparity values at the corners are real-valued, instead of using an HMM-like procedure as in Section 4.2.1, we chose a Kalman Filter to smooth the observed corner disparities $z_t^c(r)$, obtaining the estimated (final) disparities $d_t^c(r)$ for corner c of triangle r at frame t. The essence of the Kalman filter is to assume that the true state (disparity) at time t is evolved from the state at $t-1$ according to

$$d_t^c(r) = F_t(r) d_{t-1}^c(r) + w_t(r), \tag{4.11}$$

where F_t is the proportionality constant in the linear state transition model and $w_t(r) \sim \mathcal{N}(0, Q_t(r))$ is the Gaussian process noise with variance $Q_t(r)$. In out context, F_t is set to 1 to enforce temporal consistency. As for the variance of the Gaussian noise, if $Q_t(r)$ is small, the state (disparity) tends to be kept constant, yielding more temporal smoothness. On the other hand, larger values for $Q_t(r)$ allow wider state transitions, which is good when there is motion in the depth axis, or during occlusions/disocclusions. In such situations, the appearance of the corresponding triangles tends to change. Based on these considerations, we set the variance $Q_t(r)$ as

$$Q_t(r) = \exp\left(-\frac{\sigma_Q}{\delta(r_t, r_{t-1})}\right), \tag{4.12}$$

where $\delta_{t,t-1}$ is the RGB distance between the mean colors of region r at frames t and $t-1$, σ_Q controls the decay of this function, with value experimentally set to 100. Another important part of the Kalman Filter is the relation between the observed variable $z_t^c(r)$ to the estimated state $d_t^c(r)$, given by

$$z_t^c(r) = H_t d_t^c(r) + v_t(r), \tag{4.13}$$

where $H_t(r)$ is the constant of the linear observation model, and $v_t \sim \mathcal{N}(0, R_t(r))$ is the Gaussian observation noise. In our problem, $H_t(r) = 1$ so that the state is expected to be the same as the observation. The process noise $w_t(r)$ is associated with the confidence of the observation: when $R_t(r)$ is small, the state tends to be close the the observation, while wider fluctuations may arise when $R_t(r)$ is large. Similarly to the HMM-like modeling shown in Section 4.2.1, a higher confidence is expected when the plot of the cost function $m''_{r,t}$ for the corresponding triangle r presents a sharper peak at the winning disparity. Hence, the variance $R_t(r)$ is defined as

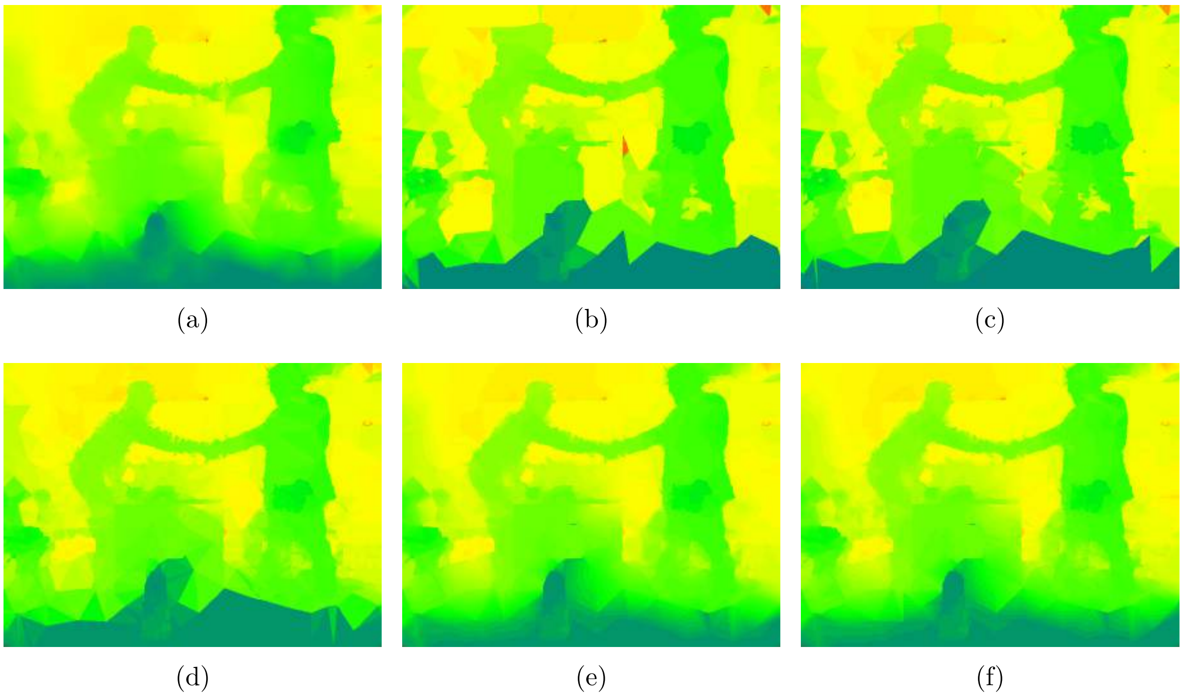
$$R_t(r) = \exp\left(-\frac{m''_{r,t}}{\sigma_R}\right) N_R, \tag{4.14}$$

where σ_R controls the decay of this exponential. According to our experiments, we defined $\sigma_R = 0.01$ and $N_R = 10$.

The complete pipeline for imposing temporal coherence to the estimated disparity maps is illustrated in Figure 4.3. As it can be observed, results get gradually better from the initial disparity estimation at frame t (Figure 4.4(b)) to the final Kalman Filter (Figure 4.4(f)). In fact, the improvement achieved by using the Kalman Filter is difficult to notice based on a single frame, but it helps reducing temporal flickering and produces better quality interpolated video sequences in all the datasets used. More results regarding the view interpolation process can be seen in Section 5.3.2.

An example of disparity estimation in time with and without (i.e. using the frame-wise approach described in Chapter 3) temporal coherence is illustrated in Figure 4.4. More precisely, we chose a single background pixel (marked in blue) in the BookArrival dataset, which gets occluded by a moving person who enters from the right, and then disoccluded (as shown in the top row of Figure 4.4). The estimated disparities are shown in the plot on the bottom. It is clear that the single-frame approach presents noisier disparity estimation, while the use of the temporal information allowed to both generate a smooth disparity within time while modelling abrupt disparity changes that occur due

Figure 4.3 – Temporal disparity estimation steps. (a) Final disparity map at frame $t-1$. (b) Initial disparity map at frame t. (c) Disparity map after the HMM-like coherence. (d) Spatial disparity filtering. (e) Refined disparity. (f) Final disparity map in frame t, after Kalman Filtering.



to occlusions/disocclusions. In particular, it can be noticed that the disparity of static objects, such as the wall on which the point is placed, should be constant. From frame 40 and onwards, it is clear that the proposed technique for videos presents a more stable disparity estimation.

4.3 Experimental Results

In order to evaluate the proposed temporal coherent disparity estimation (3DMesh-T), we compared with the previous single-frame approach (3DMesh) and with the solution from Richardt et al. (2010), which presents a single-frame approach and a temporal coherent version, respectively called DCBG and DCBG-T. We also tested with the technique from Mozerov and Weijer (2015), called TSGO, which is a recent technique that does not have temporal coherence, but was included in this comparative study because it currently is the second best ranked technique in the Middlebury setereo evaluation benchmark as of june 2015.

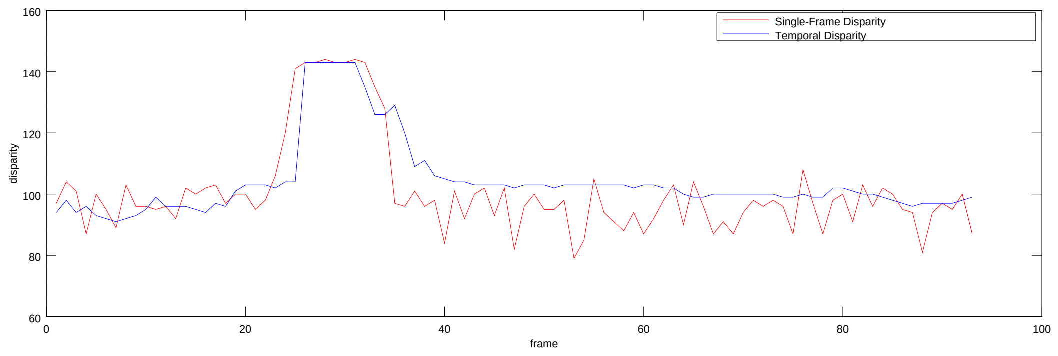
The proposed techniques, 3DMesh and 3DMesh-T, were implemented in C++, using the OpenCV (BRADSKI, 2000) and CGAL (FABRI; CACCIOLA; WEIN, 2014)

Figure 4.4 – Disparity plot of a single pixel in time. (a) First frame of the video, with the chosen pixel marked with a blue cross. (b) Frame 26. (c) Frame 52. (d) Plot of the disparities using the single frame approach and with temporal coherence. The sudden change of disparity corresponds to a person passing in front of the observed pixel.

(a)

(b)

(c)



(d)

Table 4.1 – List of parameters, which were fixed for all the tests on this thesis.

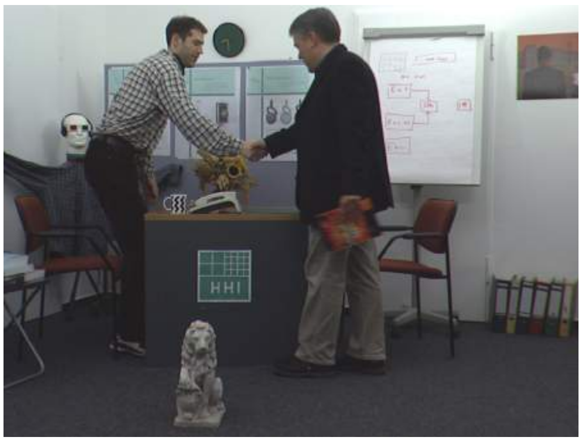
Parameter	Value	Description
N_C	5	Number of cameras
P_{min}	9	Points density on the image edges
g	6	Points density in borderless regions
γ_c	10	Decay of the color distance
$\mathcal{T}_d$	1.5	Disparity threshold to merge corners
$\mathcal{T}_o$	0.25	Optical flow error threshold for the optical flow cross checking
$\mathcal{T}_c$	20	Color threshold for the optical flow cross checking
$\mathcal{T}_e$	8	Radius of a circular search region; the vertex will be attached tothe closest pixel border inside this region
σ_d	$0.05N_s$	Controls how tight the temporal consistance is for the HMM, with N_s being the disparity search range
$\mathcal{T}_s$	128	Window size for the Disparity Filtering
σ_c	10	Controls the decay of the color similarity for the Disparity Filtering
σ_d	25	Controls the decay of the spatial distance term for the Disparity Filtering
σ_a	5	Controls the decay of the age term for the Disparity Filtering
σ_Q	100	Controls the decay of the variance $Q_t(r)$ on the Kalman Filtering
σ_R	0.01	Controls the decay of the variance $R_t(r)$ on the Kalman Filtering
N_R	10	Controls the magnitude of the variance $R_t(r)$ on the Kalman Filtering

libraries for image processing and computational geometry, respectively. All the tests were run with the parameters described in Table 4.1.

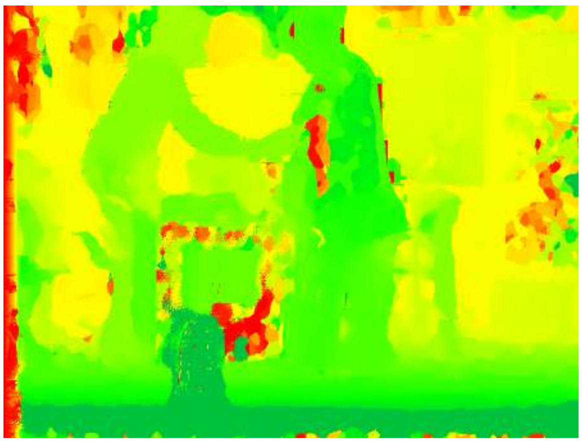
The evaluation of disparity maps for videos is a hard task since we did not find any dataset with disparity ground-truth values. However, a qualitative evaluation can be made since incongruencies in the scene 3D structure are usually apparent to the viewer. Figures 4.5 and 4.6 illustrate the estimated disparity maps at a given frame for the `BookArrival` and `Hall2` video sequences, respectively. For example, the disparity estimation on the wall of the `BookArrival` dataset is a challenge for all the tested techniques due to the lack of texture, however the use of temporal information helped to alleviate this ambiguity. The same is valid for the ground and walls in the `Hall2` sequence. The results from the DCBGrid-T were not that good, especially in the regions related to moving objects. This shows the limitation of simply aggregating the cost function using a fixed window within the time, since it presents problems when dealing with pixels that abruptly change the disparity. On the other hand, the TSGO technique presented very good results, even without using the temporal information. However, it still had some problems on the white wall from the `BookArrival` dataset and presented a blurrier disparity map on the `Hall2` dataset.

Figure 4.7 presents the results for the `CarPark` dataset, and it can be observed that the disparity maps of both 3DMesh and 3DMesh-T had problems on correctly seg-

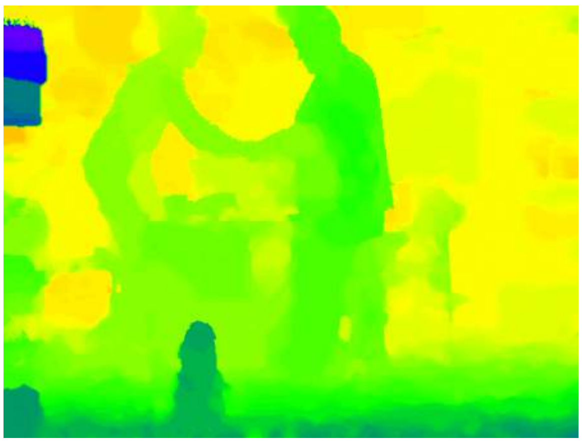
Figure 4.5 – Results of disparity maps with temporal coherence on `BookArrival` database. (a) Ground truth. The following pictures are disparity maps generated using: (b) DCBG-T, (c) TSGO, (d) 3DMesh, (e) 3DMesh-T.



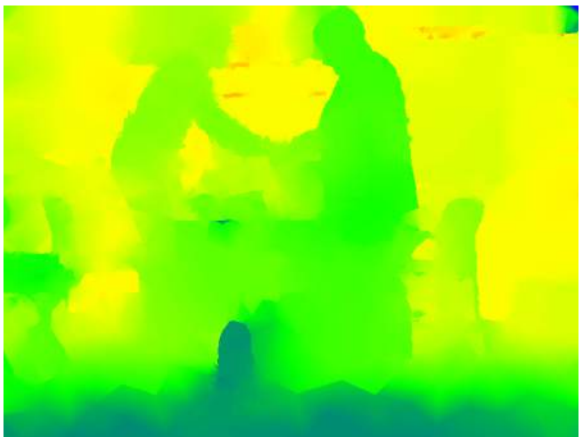
(a)



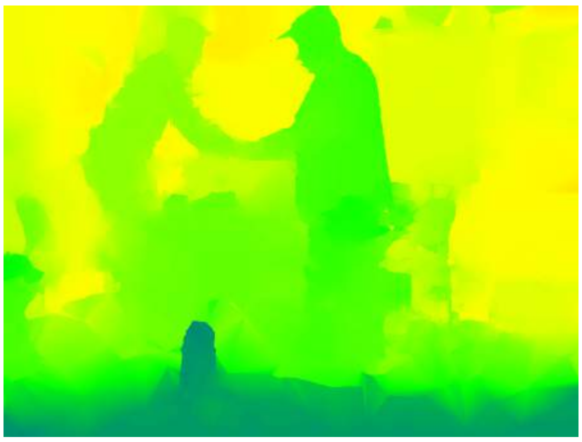
(b)



(c)

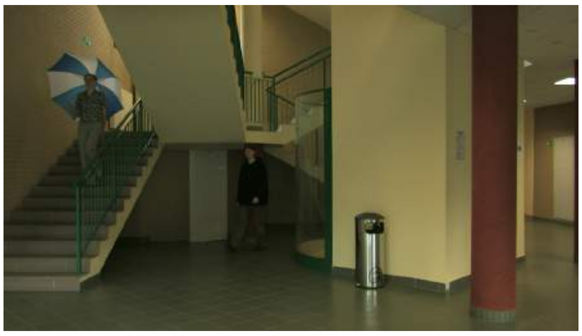


(d)

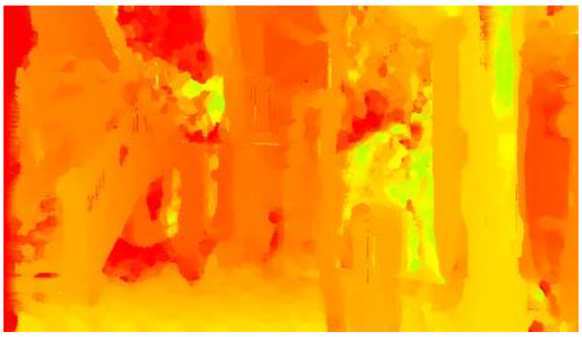


(e)

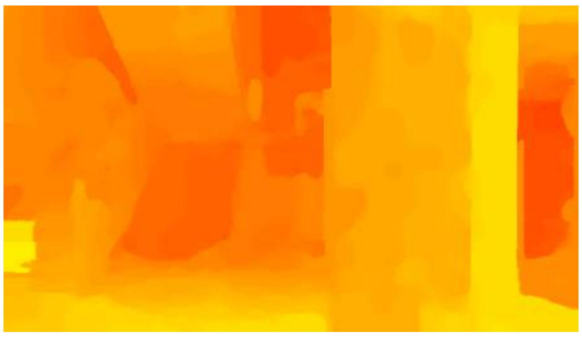
Figure 4.6 – Results of disparity maps with temporal coherence on `Hall2` database. (a) Ground truth. The following pictures are disparity maps generated using: (b) DCBG-T, (c) TSGO, (d) 3DMesh, (e) 3DMesh-T.



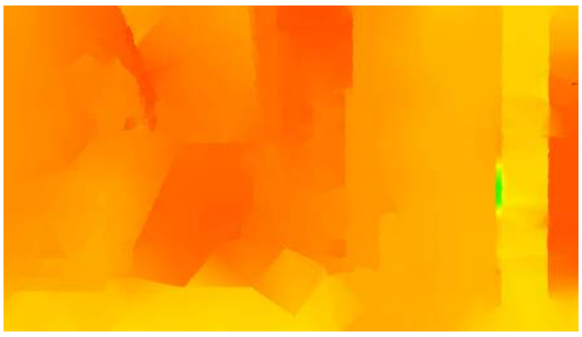
(a)



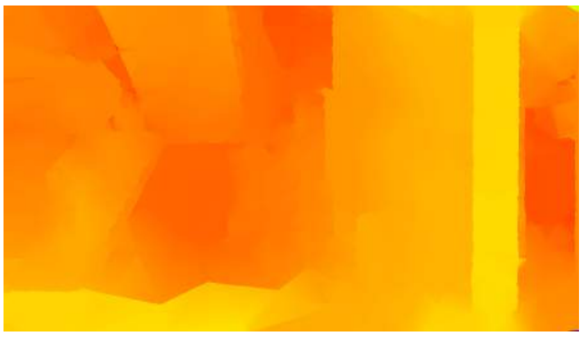
(b)



(c)



(d)



(e)

menting the persons. This problem occurs due to the highly textured area within their silhouette, which makes the vertices distribution a difficult task. Since those vertices were not correctly distributed along the boundaries of the persons, the resulting triangles and disparities were compromised. This affected the temporal coherence process since the propagation of vertices generated a more distorted mesh.

The evaluation of the temporal smoothness of the proposed approach and competitive techniques is illustrated in Figures 4.8, 4.9, and 4.10. The results of 3DMesh-T proved to be better than 3DMesh most of the tests, and comparable to competitive approaches. In fact, for the `Hall` dataset shown in Figure 4.8, the proposed approach presented a much more stable disparity on the low-textured walls. Figure 4.10 also illustrates that our technique present a more stable disparity estimation, however it presents some oscilation around the frames 100-150. On those frames there is a person passing, and as it can be seen in Figure 4.7, there are some disparity problems around the persons.

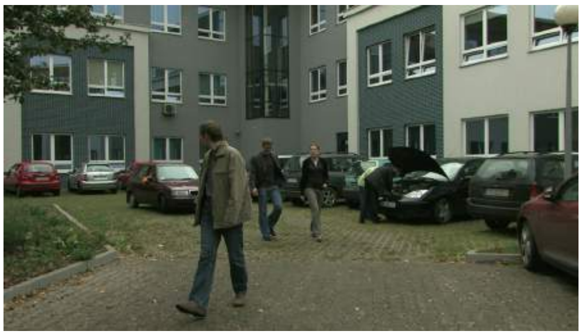
For a video comparison between those disparity estimation techniques the reader can access the authors webpage: <http://inf.ufrgs.br/~gpfickel/phdthesis/> .

4.4 Chapter Conclusions

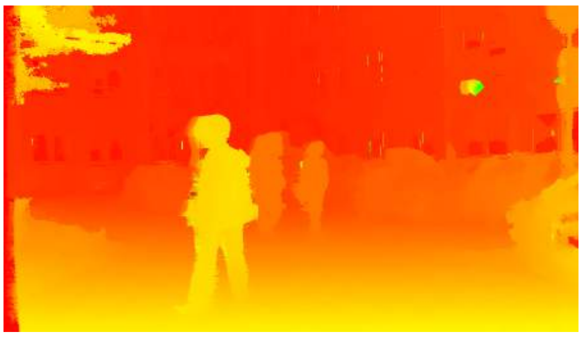
This chapter presented a temporal update scheme for the initial 3D mesh of the scene, which aims to impose temporal coherence to the estimated disparity map. This process is done by applying the following operations: i) moving vertices ii) deleting vertices and iii) adding vertices. With those three simple operations, it was possible to model movement, occlusions and disocclusions, respectively. However, to track points within the objects boundaries is a challenging task, which several particle trackers do not tackle. It was then proposed to enhance the particle tracker from Sundaram, Brox and Keutzer (2010) by adding the disparity information, which made possible to deal with those hard to track particles that would be discarded otherwise.

Finally, temporal consistency was imposed to the adaptative mesh. Such coherence starts with using and HMM-like filtering process for each triangle that persists in time. A propagation scheme is used to improve the initial estimative of new (or recent triangles), and a subpixel filtering process based on the Kalman Filter is used to generate the final result.

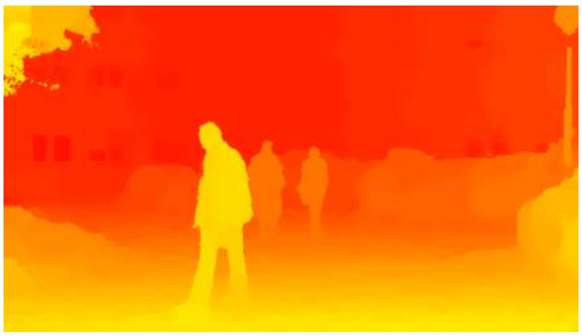
Figure 4.7 – Results of disparity maps with temporal coherence on `CarPark` database. (a) Ground truth. The following pictures are disparity maps generated using: (b) DCBG-T, (c) TSGO, (d) 3DMesh, (e) 3DMesh-T.



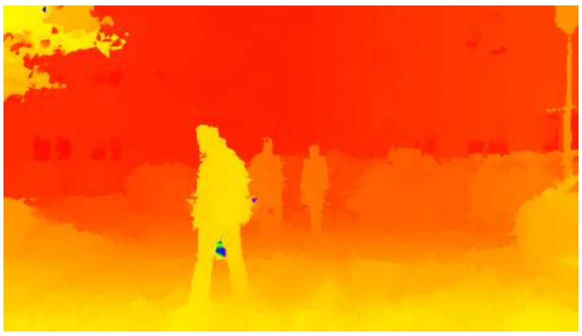
(a)



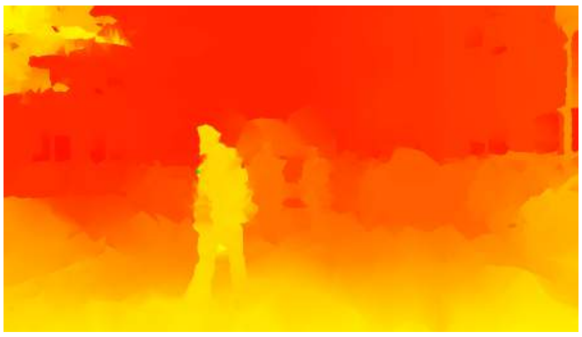
(b)



(c)

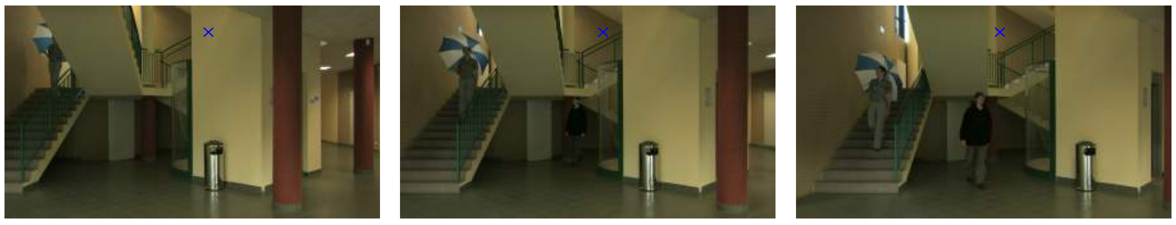


(d)

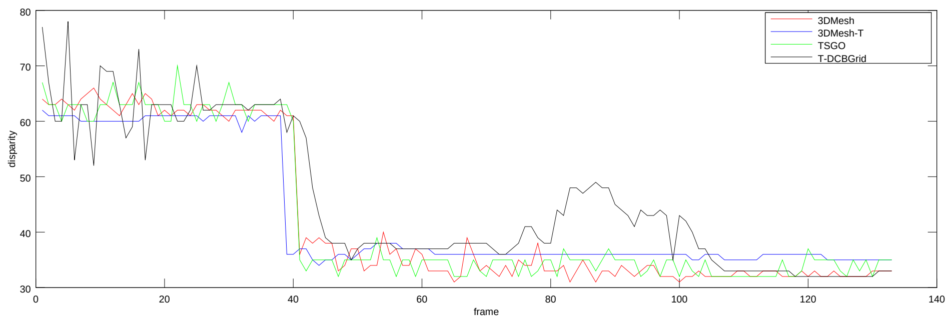


(e)

Figure 4.8 – Disparity plot of a single pixel in time. (a) First frame of the video, with the chosen pixel marked with a blue cross. (b) Frame 65. (c) Frame 130. (d) Plot of the disparities using 3DMesh, 3DMesh-T, DCBG-T and TSGO.

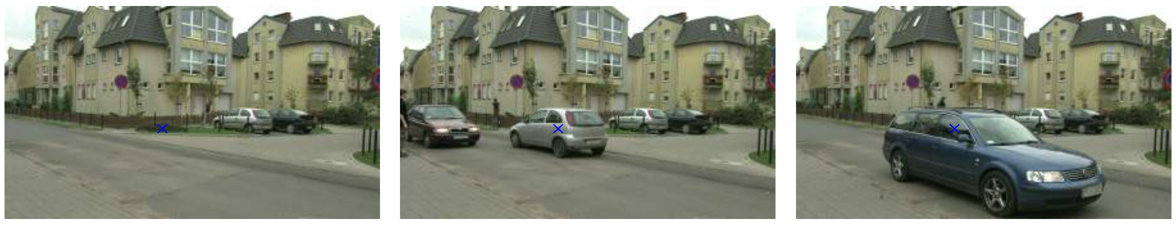


(a) (b) (c)

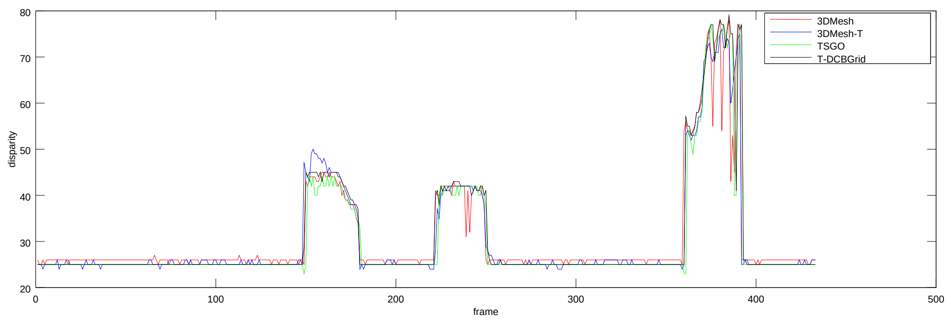


(d)

Figure 4.9 – Disparity plot of a single pixel in time. (a) First frame of the video, with the chosen pixel marked with a blue cross. (b) Frame 180. (c) Frame 380. (d) Plot of the disparities using 3DMesh, 3DMesh-T, DCBG-T and TSGO.

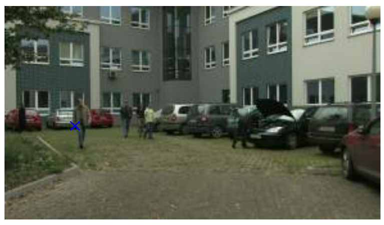


(a) (b) (c)



(d)

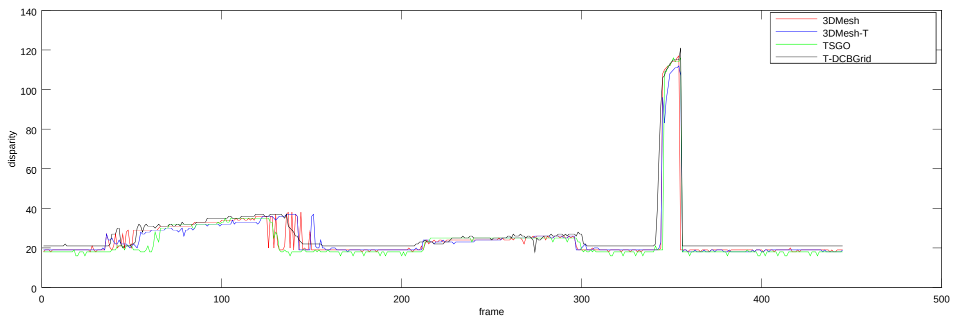
Figure 4.10 – Disparity plot of a single pixel in time. (a) Frame 100, with the chosen pixel marked with a blue cross. (b) Frame 250. (c) Frame 350. (d) Plot of the disparities using 3DMesh, 3DMesh-T, DCBG-T and TSGO.



(a)

(b)

(c)



(d)

5 VIEW INTERPOLATION

In a linear array with $N_{\mathcal{C}}$ cameras, we apply the procedure described so far using each of the cameras as the reference image, so that $N_{\mathcal{C}}$ 3D triangular meshes of the same scene are obtained. In theory, these models should be coherent, such that the projection of any 3D model at the position of the actual cameras should yield exactly the image produced by that camera. However, since we have to deal with several problems such as noisy disparity estimates and limited view of the scene, it is very difficult to have a fully coherent set of 3D models.

Since the problem of generating a single coherent 3D mesh from all the $N_{\mathcal{C}}$ representations is very difficult, we believe that combining the projections of the 3D models directly in the image domain is much simpler and produces good visual results when compared to other DIBR methods. In the following sections we present two different view interpolation approaches depending on the number of cameras available, i.e. 2 cameras or more.

5.1 View Interpolation Using Two Cameras

When using a linear array of cameras, the usual view interpolation process consists in warping the views from the closest cameras to the synthetic view position. These images are combined by weighing both images based on the distance of the synthetic view to the neighboring cameras. More precisely, let $0 \leq \alpha \leq 1$ denote a normalized distance between two adjacent cameras, such that $\alpha = 0$ corresponds to the left view and $\alpha = 1$ to the right view. The interpolated view I_s can be written as

$$I_s(x, y) = (1 - \alpha'(x, y))I_L^{\alpha}(x, y) + \alpha'(x, y)I_R^{\alpha}(x, y), \tag{5.1}$$

where I_L^{α} and I_R^{α} are the warped versions of the left and right images to the desired viewpoint. They are obtained by projecting each pixel (x, y) from the reference image to the desired position by displacing them according to the estimated disparity maps $D_L(x, y)$ and $D_R(x, y)$, and weighted by α and $1 - \alpha$, respectively. Also, $\alpha'(x, y)$ is the blending value for each pixel, given by

$$\alpha'(x,y) = \begin{cases} \alpha, & \text{if } I_L^\alpha(x,y) \text{ and } I_R^\alpha(x,y) \text{ are valid} \\ 1, & \text{if only } I_R^\alpha(x,y) \text{ is valid} \\ 0, & \text{if only } I_L^\alpha(x,y) \text{ is valid} \end{cases}, \tag{5.2}$$

where a pixel (x, y) is considered valid from a given projection I^α (left or right) if it has any information projected. Figure 5.1 illustrates an example of view interpolation, with the projections from the left and right views combined into the final synthetic view. It is important to notice that the orthogonal triangles are not capable of removing the black region around the image boundaries (left or right, depending on where the virtual view is), since they do not have correpondence with the other views. For those regions, only one of the projections contributes to the synthetic view.

The typical alpha blending approaches generate a view with possibly several holes (i.e. pixels without any projection), depending on the quality of the disparity map. This problem usually is tackled with a hole filling technique, which usually are computationally expensive if there is need for a more complex texture synthesis procedure. However, the 3D meshes obtained with the proposed approach generate a warped view without any holes due to the use of orthogonal triangles. As explained in Section 3.5, by using this technique we were able to achieve high quality interpolated views while avoiding the common and computationally costly hole filling process.

5.2 View Interpolation Using Multiple Cameras

The previous solution is conceptually very simple and fast, but visual artifacts are still noticeable, particularly along the boundary of the objects. Unfortunately, due to errors in the disparity map, the projections may present artifacts.. Those artifacts can be subdivided in the following 2 categories, illustrated in Figure 5.2:

1. wrong projection: with this error we have a projection value for the pixel p_i, but it is the projection of a wrong value.

2. ghost effect: also a wrong projection error, but it occurs only near the disparity discontinuities due to the limitations of the disparity map to correctly represent the objects boundaries. This is especially difficult when the boundaries are blurred and not well defined, usually related to elements out of focus.

Figure 5.1 – View interpolation using two views.

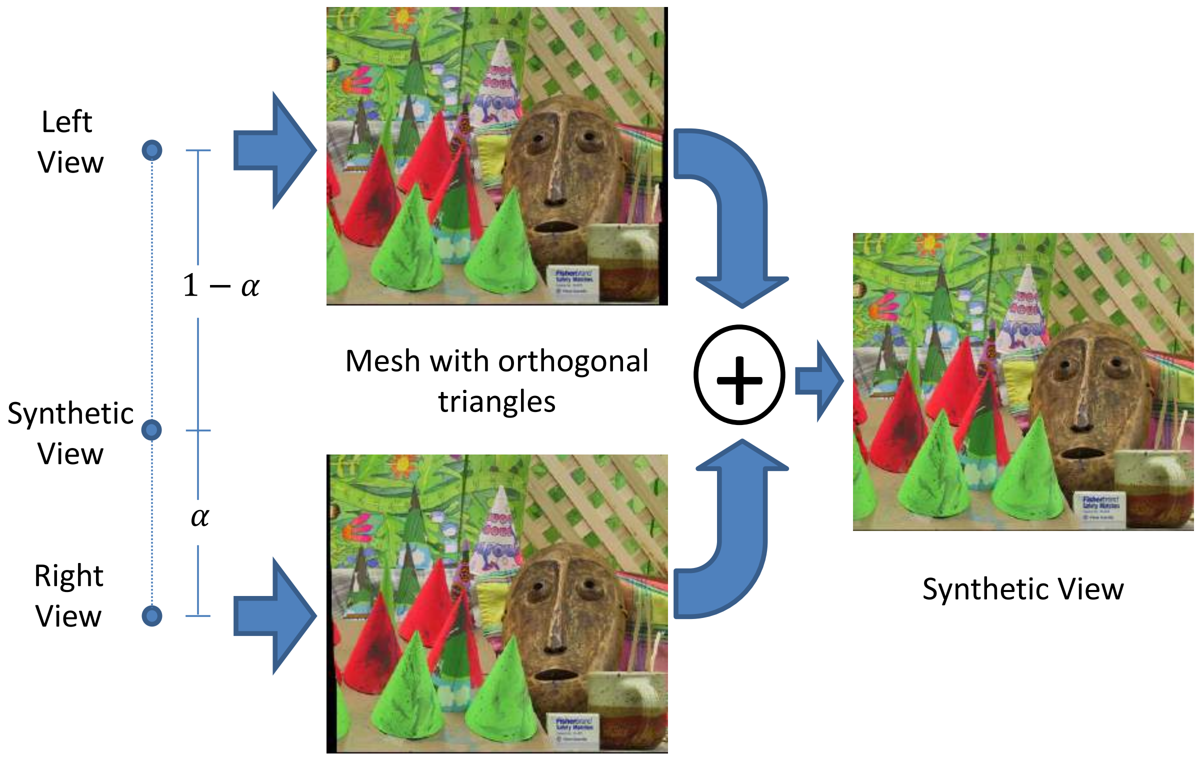
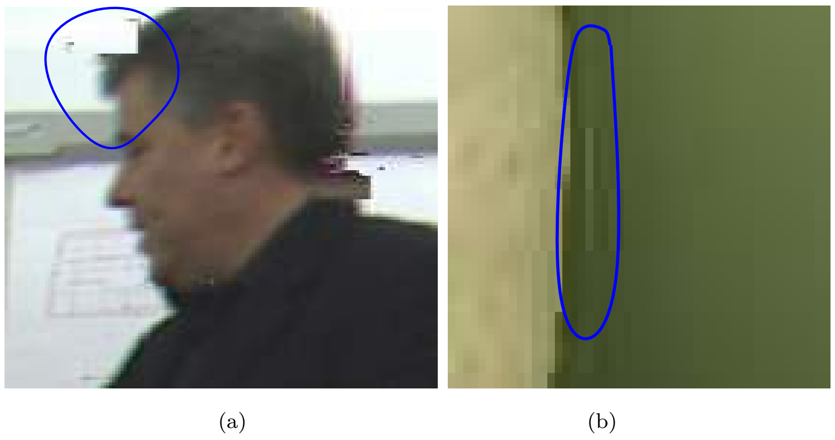
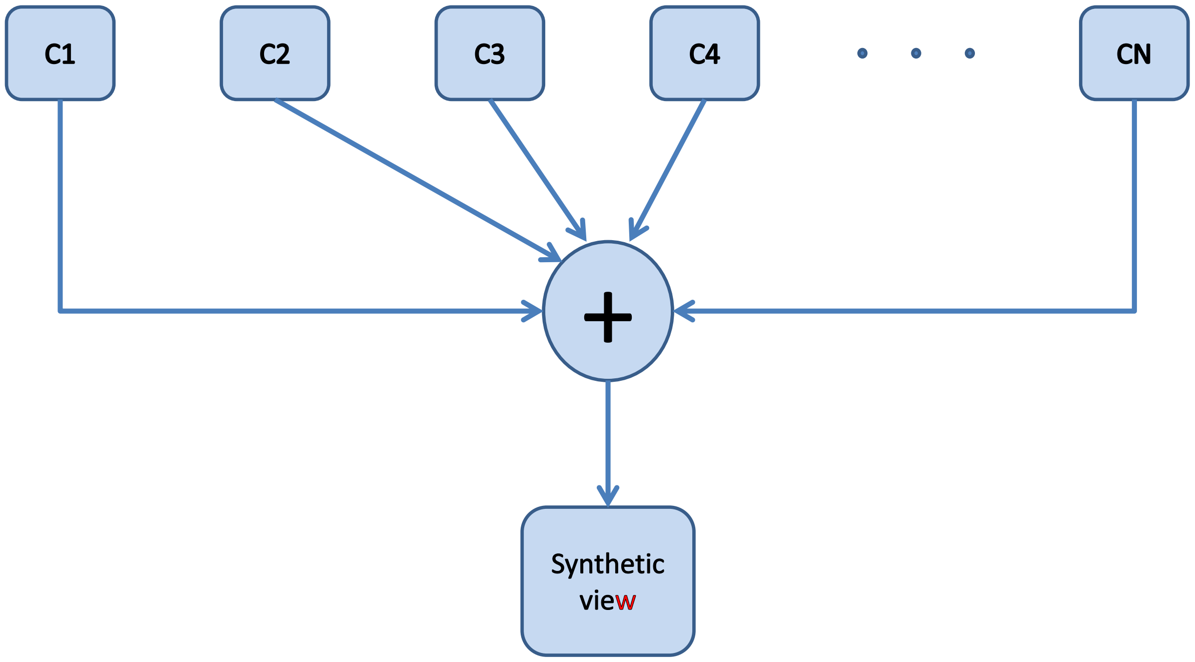


Figure 5.2 – Examples of projection errors, with a blue marker around the important regions. (a) Example of error type 1. Notice the projections of the white wall when should be the the person hair. (b) Example of error type 2. Notice the erroneous clearer pixels within the green background.



One possible way to generate a better interpolated view is to use the projections from all available cameras, such as shown in Figure 5.3, since they provide more redundant information about the scene. And by using more views it is possible to not use the orthogonal triangles to render the meshes, and let that other projections fill those holes with their information. However, to combine them in a coherent manner to generate the final synthesized view is not trivial. In an ideal scenario, all projections should be very similar, and the simple mean value would provide a good estimate of the synthesized view. However, outlier pixels tend to arise due to occlusions and inconsistencies in the computation of the disparity map, and they can corrupt the quality of the synthesized view.

Figure 5.3 – View interpolation using multiple cameras. C_i is the projection from a given camera i.



In order to compute the average of coherently warped pixels we propose the use of a Weighted Vector Median Filter (WVMF) (ASTOLA; HAAVISTO; NEUVOS, 1990). The WVMF is a vector processing operator that present a robust data smoothing ability while preserving sharp edges in the signal, so it is appropriate for our application.

A given pixel $\boldsymbol{p}$ in the synthesized image domain can be reached by the warping of different reference images in the multiview configuration. Let $\boldsymbol{p}_i$ denote the RGB color value of each pixel that projects to $\boldsymbol{p}$, for $i = 1, ..., N_C$. In a WVMF, the first step is to compute the distance d_i from each vector to all others:

$$d_i = d(\boldsymbol{p}_i) = \sum_{j \neq i} \|\boldsymbol{p}_i - \boldsymbol{p}_j\|, \tag{5.3}$$

where $\| \cdot \|$ is a vector norm (we used the L_2 norm in this work). The WVMF is then given by

$$\boldsymbol{p}_s = \frac{1}{\sum_{i=1}^{N_C} W_i} \sum_{i=1}^{N_C} W_i \boldsymbol{p}_i, \tag{5.4}$$

where the weight $W_i = f(d_i)$ is computed based on a monotonically decreasing function f applied to the distance values d_i. For inliers, the values of d_i tend to be smaller, and they carry more weight in the computation. On the other hand, outliers are far from the other samples, thus presenting larger distances d_i and a smaller weight.

In the context of view interpolation, it is natural that the actual views closer to the synthetic viewpoint should be more similar to the desired synthesized image. Hence, instead of considering only the color distances of the RGB pixel values to compute the WVMF, we also include the physical distances between the synthetic view and each existing view. More precisely, the proposed weights W_i used in Equation (5.4) are given by

$$W_i = f(d_i, x_i) = \exp\left(-\frac{d_i^2}{2\sigma_1^2} - \frac{x_i^2}{2\sigma_2^2}\right), \tag{5.5}$$

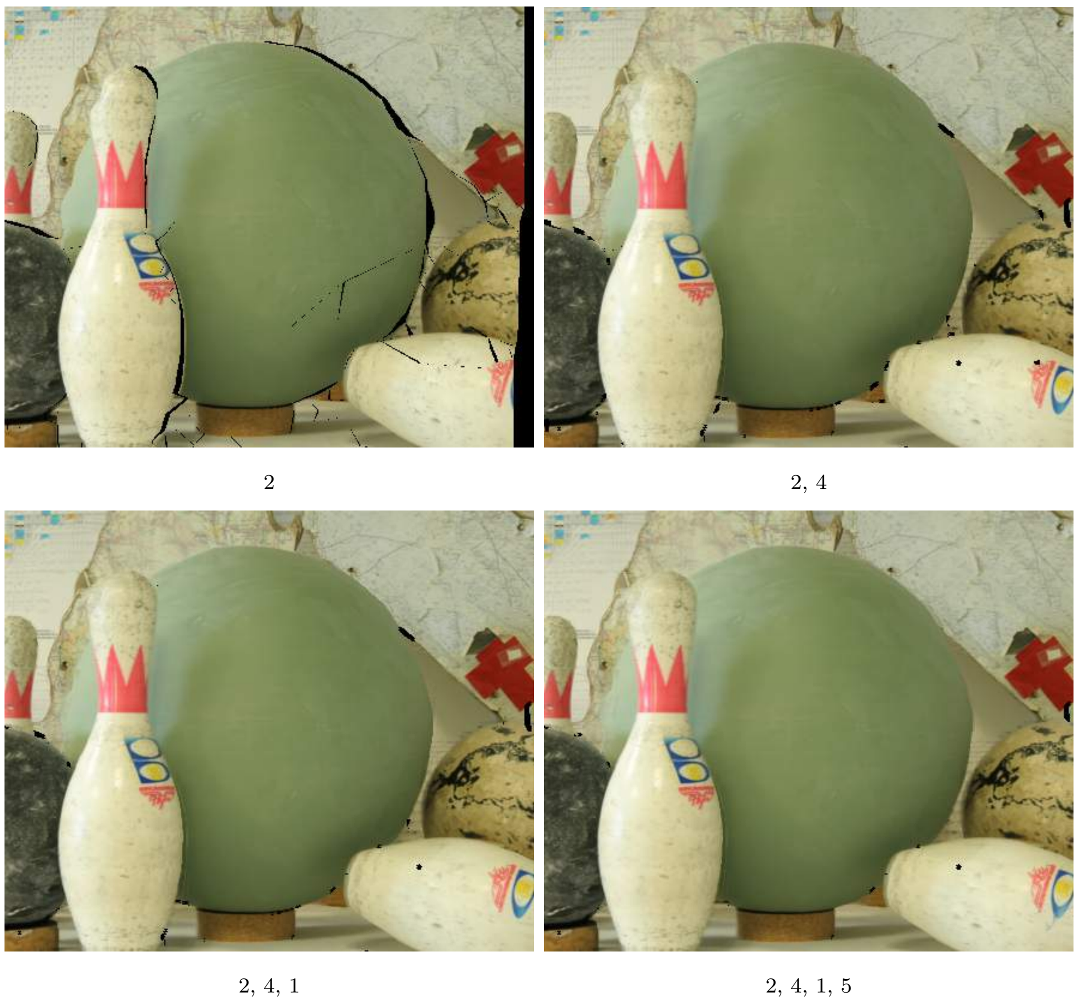
with $\sigma_1 = 3$ and $\sigma_2 = \frac{3}{\mathcal{P}(\boldsymbol{p})}$ that control the Gaussian decay of the color distance and the physical distance, respectively, with $\mathcal{P}(\boldsymbol{p})$ being the number of valid projections in point $\boldsymbol{p}$. Their values were defined based on experimental analysis.

As the number of cameras in the multiview setup increase, the number of holes in the synthesized image decrease significantly, since the occlusions generated by a given pair of cameras can be compensated by others. So we choose to refrain from using the orthogonal triangles (defined in Section 3.5), since the projections of the other cameras usually leads to a better result. As we can see in Figure 5.4, after using 4 views there are only a few and small remaining holes, which could be easily filled filled with simple image-based methods, such as the median filter. But since we have an estimative for the holes, given by the orthogonal triangles, we use them only to fill the remaining holes after combining all the projections.

5.3 Experimental Results

In general, quantitative evaluations of the disparity map estimation are based on the direct comparison of the obtained maps with ground truth data (MIDDLEBURY, 2012). For a quantitative comparison of the view interpolation results, we use a subset of

Figure 5.4 – WVMF view interpolation example of the view 3 from `Bowling2`'s Middlebury database, without using orthogonal triangles to complete the 3D mesh. The used views in the interpolation process are indicated below each interpolated view.



the cameras to estimate the disparity maps, and put the virtual camera at the locations of the actual cameras that were not used to obtain the disparity maps (such cameras are used only for validation purposes).

In the following section we will present the view interpolation results using the single-frame 3D mesh approach from Chapter 3. And afterwards we present our results for video view interpolation using the 3D mesh with temporal update from Chapter 4.

5.3.1 Single-Frame View Interpolation

In order to evaluate the results of our approaches, we compared visually and quantitatively (PSNR) with the synthetic views produced by the MPEG view synthesis reference software VSRS 3.5 (TANIMOTO; FUJII; SUZUKI, 2009), using the disparity maps estimated by Yang (2015), Rhemann et al. (2011), and Mozerov and Weijer (2015), called VSRS-nonlocal, VSRS-volume and VSRS-TSGO, respectively. It is interesting to note that the execution time of VSRS 3.5 (without considering the computation of the disparity maps) is around 1s for a typical 442×370 Middlebury image (2006 dataset), whereas our 2 view interpolation process (without the 3D model computation) is approximately 0.015s and the multiple view interpolation is 0.3s on a Core i5 2.2GHz processor (notebook version), 3GB RAM.

Figure 5.5 shows the ground truth images and the interpolated views produced by our approach using 2 views and multiple views using WVMF, VSRS-nonlocal, VSRS-volume and VSRS-TSGO for Middlebury datasets `Baby3` and `Bowling2` (MIDDLEBURY, 2012; HIRSCHMULLER; SCHARSTEIN, 2007)[1], as well as for the `Herodion` dataset. Also, a quantitative comparison in terms of the PSNR of the synthesized views for other Middlebury datasets is presented in Table 5.1. As it can be observed, all synthetic views are mostly coherent with the actual views for the Middlebury datasets, but the proposed approach appears to produce more realistic colors. And as expected, the redundant information used in the WVMF-ours approach was able to outperform the 2view-ours approach in almost all the tests. It is clear in the WVMF-ours results from the bowling sequence that the extra information was able to greatly alleviate ghost artifacts around the bowling pin. However, in the `Flowerpots` and `Baby1` sequences there was some large occluded regions for which our stereo matching algorithm failed, so the extra cameras only contributed with wrong information in those areas.

As for the `Herodion` example, all synthetic views presented some distortion on the checkerboard pattern behind the bike (bottom-right), and both VSRS-nonlocal and VSRS-volume present ghosts around the tennis racket. In terms of PSNR, the proposed approaches (both 2- and multiple- views) presented larger values consistently in all examples.

In another experiment, we have used a higher resolution dataset (the 1330×1100 version of `Bowling2`) and moved the virtual camera continuously from the left camera position ($\alpha = 0$) to the right camera position ($\alpha = 1$) using the 2 view interpolation. The views generated by this approach using images 1 and 5 as left and right reference images, with $\alpha \in \{0.2, 0.4, 0.6, 0.8\}$, are shown in Figure 5.6. Despite the presence of

[1]For view interpolation we used 2006 datasets `Baby3` and `Bowling2` because they are provided with camera parameters, unlike 2001 and 2003 sequences Teddy, Venus, Tsukuba and Cones, used only to evaluate the disparity maps.

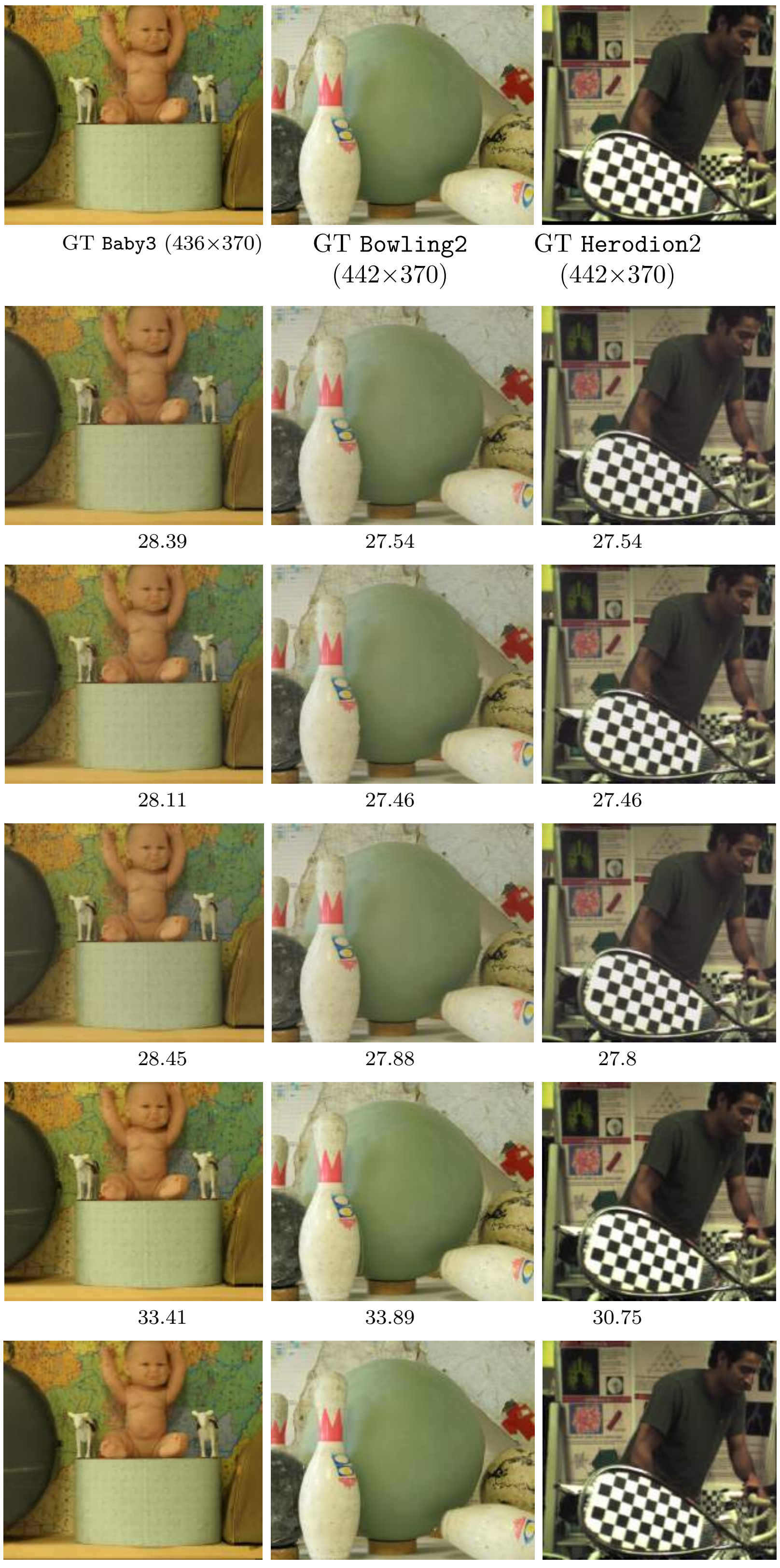
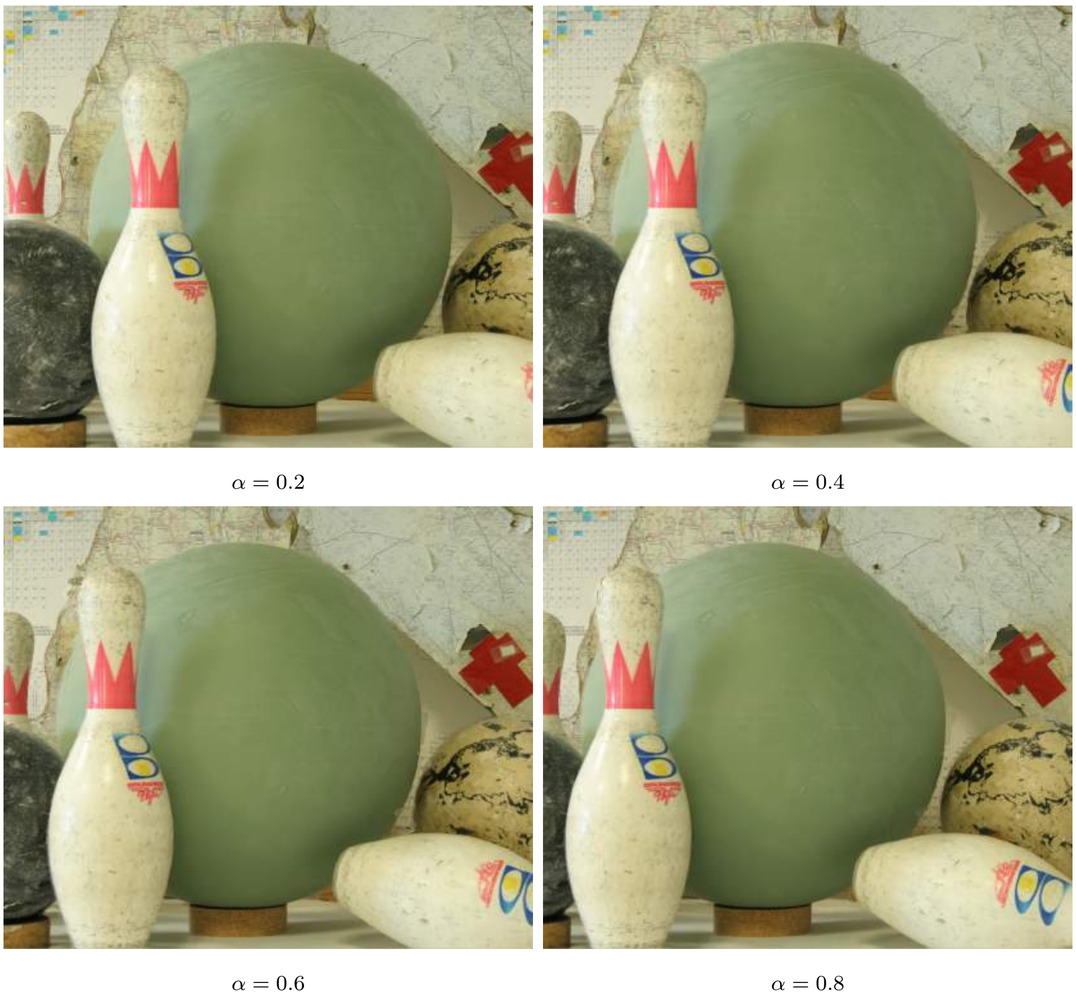


Figure 5.5 – Ground truth images (first row) and interpolated views with the respective PSNR values for VSRS-nonlocal (row 2), VSRS-volume (row 3), VSRS-TSGO (row 4), the proposed 2 view interpolation approach (row 5) and the WVMF approach (last row).

Figure 5.6 – Interpolated views between cameras 1 and 5 of the `Bowling2` dataset using our approach.



some small artifacts, the generated views are visually coherent. A video sequence with the synthetic views produced by a continuous camera movement is available at <http://www.inf.ufrgs.br/~gpfickel/phdthesis/>, as well as video sequences obtained with VSRS-nonlocal, VSRS-volume and VSRS-TSGO. It can be observed the 2 view interpolation approach produces a smooth transition between the reference images, since the full 3D meshes present no holes to be filled *a posteriori*. On the other hand, the interpolation approach adopted in reference software VSRS 3.5 may need to fill gaps directly in the image domain. Since this procedure is applied independently to each synthesized image, spatial coherence between the views of two close virtual cameras is not guaranteed. In fact, this behavior can be observed in sequence `VSRS-nonlocal.avi`, on the right of the central bowling bowl. It should also be noted that these camera sweep views can be obtained in real-time with our approach (unlike with VSRS 3.5), since the 3D model is computed once and the interpolation itself is very simple.

Table 5.1 – PSNR values of synthesized views using the following approaches for several Middlebury datasets (best values in bold): VSRS-volume (RHEMANN et al., 2011), VSRS-nonlocal (YANG, 2015), VSRS-TSGO (MOZEROV; WEIJER, 2015), our interpolation approach for 2 views (2views-ours) and for multiple views (WVMF-ours)

Method	Dataset							
	`Aloe`	`Baby1`	`Cloth1`	`Cloth2`	`Cloth3`	`Flowerpots`	`Rocks1`	`Woods1`
VSRS-volume	24.96	25.99	27.03	24.66	25.69	27.16	28.20	27.40
VSRS-nonlocal	24.42	25.66	26.85	25.10	25.99	24.77	28.65	27.34
VSRS-TSGO	26.13	28.45	29.23	25.51	27.05	27.50	25.57	27.06
2views-ours	29.35	**34.45**	37.19	33.45	33.64	**33.70**	33.01	35.01
WVMF-ours	**30.12**	33.26	**37.23**	**33.54**	**34.16**	32.88	**36.57**	**36.52**

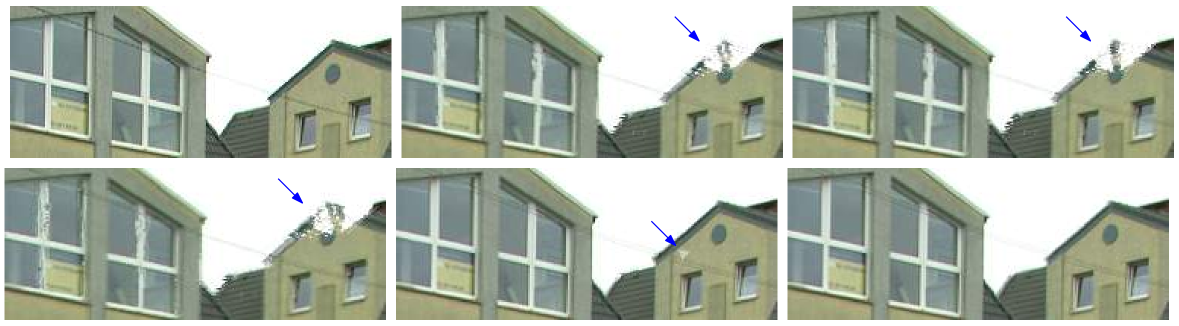
5.3.2 Video View Interpolation

Our results were evaluated qualitatively (visual inspection) and also quantitatively, using an objective Quality Assessment (QA) metric suited for video sequences, namely the STRRED (Spatio-Temporal Reduced Reference Entropical Difference) (SOUNDARARAJAN; BOVIK, 2013). Such metric is shown to correlate well with human judgments of quality, and it encompasses both the visual errors within a given frame and the temporal flickering that occur within consecutive frames.

For the sake of comparison, we have also produced synthetic views using the MPEG view synthesis reference software VSRS 3.5 (TANIMOTO; FUJII; SUZUKI, 2009) in conjunction with other disparity estimation algorithms: DCBG and DCBG-T for the regular and temporal cohesion techniques (RICHARDT et al., 2010) (code available at <http://www.cl.cam.ac.uk/research/rainbow/projects/dcbgrid/>), and also TSGO (MOZEROV; WEIJER, 2015) (code available at <http://www.cvc.uab.es/~mozerov/Stereo/>), which is based on global optimization technique. We tested our approach suited for videos that was shown on Chapter 4 and the single frame approach from Chapter 3, respectively referred as 3DMesh-T and 3DMesh.

Table 5.2 shows the quantitative STRRED results using the `BookArrival`, `CarPark`, `Hall2`, and `Street` datasets (available at <http://sp.cs.tut.fi/mobile3dtv/stereo-video/>). Despite the more complex view interpolation procedure employed by VSRS, with several pre- and post-processing procedures of both the disparity map and the interpolated view, the proposed approach presented the best results (smaller is better) in three of the four sequences. The `CarPark` sequence presents several textured regions, so that our domain triangulation scheme is not able to capture correctly the boundaries of the objects. Nevertheless, our temporal consistency produced improved results over our sin-

Figure 5.7 – Example of view interpolation using the `Street` dataset, frame 68, camera 5, generated using cameras 4 and 6. A blue arrow is indicating the most noticeable visual artifact. From left to right: Ground truth, VSRS+DCBG, VSRS+DCBG-T, VSRS+(MOZEROV; WEIJER, 2015), 3DMesh, 3DMesh-T



gle frame approach. It is also interesting to note that for the `Book` dataset, the temporal consistent disparity map (DCBG-T) presented worse view interpolation results than its frame-wise counterpart DCBG. This fact indicates that although temporal coherence is in theory important for better view interpolation, it should also allow abrupt disparity transitions in newly disoccluded regions, otherwise temporal consistency can degrade the results.

Regarding the proposed approach for videos, in all tests there was an improvement over the single-frame approach. Even for the dataset Hall2 that present camera movement, the 3D mesh was able to track correctly the background while adapting the mesh to both the occlusions and disocclusions. In the datasets Street and CarPark the interpolation results were also good, even when using natural lighting and with objects/persons whose distance to the camera changes. This indicates that our technique can both enforce a smooth disparity variation on the background objects while allowing it to change gradually for moving objects.

As for the visual inspection, the full interpolated video sequences are available on the author personal webpage, where the temporal dimension can be better observed. For the sake of illustration, Figure 5.7 shows the actual image (ground truth) for frame 275, camera 5, of the `Street`, dataset, as well as the interpolated views with the methods shown in Table 5.2. As it can be observed, VSRS+DCBG, VSRS+DCBG-T and VSRS+TSGO present strong artifacts on the top of the right building, the single frame approach presents a smaller artifact on the wall of the same building. Our approach, on the other hand, presents an image that is very similar to the actual view.

Table 5.2 – STRRED results of VSRS, the frame-by-frame proposed approach (3DMesh) and the proposed approach for videos (3DMesh-T). The VSRS was tested using the disparity maps estimated by Mozerov and Weijer (2015) and Richardt et al. (2010), with DCBG and DCBG-T being the regular and temporal cohesion techniques, respectively. The best results are in bold.

Method	BookArrival	CarPark	Hall2	Street
VSRS+DCBG	33.14	59.33	251.64	66.20
VSRS+DCBG-T	37.16	52.63	224.41	62.20
VSRS+TSGO	18.69	**48.47**	202.67	78.59
3DMesh	20.99	130.14	154.77	47.65
3DMesh-T	**18.21**	108.61	**137.01**	**35.63**

5.4 Chapter Conclusions

This chapter presented two different approaches for the interpolation process, which starts by rendering the available 3D meshes on the novel view position. Ideally, all the projections should be identical, but inconsistencies arise due to bad disparity estimation, camera noise and limited view of the scene. To generate the interpolated view we then propose to combine those projections directly in the image domain. This can be done using only the 2 closest cameras or all of them (if more than two are available).

The 2-camera view interpolation procedure consists in alpha blending both projections, with the alpha being proportional to the distance of the synthetic view and the real camera. This approach is simple, and it has the advantage that it has a spatial coherence in all the views that are generated along the baseline. The second approach uses the projections of all cameras, and combines them in a pixel-wise manner using the WVMF (Weighted Vector Median Filter). We defined the WVMF weights according to both the color similarity and the distance of the virtual view to the camera (the farthest they are, the less confident is the contribution of this given 3D mesh). This approach was able to reduce some artifacts, especially the ghost effect, without introducing noticeable blur on the highly textured areas.

6 CONCLUSIONS

In this thesis, we proposed a new scheme for stereo matching and view interpolation based on triangular tessellations of the image domain. The first step of the proposed approach is to detect edges and find local estimates of the image complexity so that the vertices for a Delaunay triangulation are placed mostly along edges and highly textured regions. A region-based approach is used to find an initial disparity for each triangle, based on histograms of pixel ratios followed by a non-local aggregation step based on Minimum Spanning Trees. Then, a refinement procedure is applied to smooth the initial disparity map, generating a piece-wise linear representation of the disparity. A full 3D mesh is created by connecting corners with sufficiently small disparity difference, and inserting new orthogonal triangles along the disparity discontinuities. The textures of visible triangles are retrieved from the reference image, whereas textures of orthogonal triangles are obtained from the two closest neighboring cameras.

We also proposed a 3D mesh update scheme that dynamically deforms the triangulation according to the changes on both the scene and camera when multiview video sequences are used. In order to correctly track the particles (vertices) along the objects boundaries, we extend the tracker from Sundaram, Brox and Keutzer (2010) to use the disparity map. This information allowed us to identify the particles that do not have a well-defined optical flow value (usually along image edges) and move them according to well tracked neighbors. After deleting and inserting new partciles (due to mostly occlusions and disocclusions, respectively), a constrained Delaunay triangulation is computed, aiming to keep and track triangles for which the vertices persist. By using HMM and Kalman Filter in the temporal persistent regions and a spatial Disparity Filtering procedure that propagates the temporal information to newly created triangles, we were able to generate a disparity maps with higher quality and less temporal flickering than competitive approaches.

It is important to point out that the main goal of the proposed stereo/multiview matching approach is not to obtain a very accurate disparity map, but to provide an easy way to generate full 3D meshes at each camera for view interpolation. However, the proposed temporally coherent disparity estimation was able to achieve good results for challenging videos. By using an adaptive region-based triangular segmentation and subsequently applying a non-local aggregation, it was possible to estimate correctly the disparity in large low-textured regions, even with the presence of camera noise. And by propagating this information in time (and subsequently in space), the final disparity map was good, even when comparing to other competitive approaches such as Richardt et al. (2010) and Mozerov and Weijer (2015).

With the obtained 3D meshes, it is possible to generate high quality interpolated views with low execution time. To generate the interpolated views we proposed to combine the individual projections of the 3D meshes directly in the image domain, using only the 2 closest cameras or all of them. The 2-cameras view interpolation performs alpha blending using both projections, with the alpha value being proportional to the distance of the synthetic view and the real camera. This approach is simple, and it has the advantage that it has a spatial coherence. The multiple cameras view interpolation combines the projections in a pixel-wise manner using the WVMF (Weighted Vector Median Filter). With the WVMF weights being dependent on both the color similarity and the distance of the virtual view to the camera, it was able to both reduce artifacts while keeping the textured areas sharp. And since both methods used the textured orthogonal triangles, there is no need for a hole filling procedure.

Regarding the quality of the interpolated views, our experimental results using ground truth data indicate that the generated synthetic views match closely the actual views. By comparing the proposed approach with the MPEG view synthesis reference software, we tested our approach using both still-images and videos. We used the disparity maps estimated by Yang (2015), Rhemann et al. (2011), and Mozerov and Weijer (2015) for the still-images comparison, and both the 2-cameras and multiple cameras view interpolation process presented superior using PSNR. And as expected, the results of the multiple camera interpolation approach was able to reduce some artifacts, especially the ghost effect, without introducing noticeable blur on the highly textured areas.

For the videos comparison we used VSRS with the disparity maps from Richardt et al. (2010) and Mozerov and Weijer (2015), and it was clear that the proposed scheme indeed produced interpolated views with higher visual quality according to the used STRRED quality metric, which is shown to correlate well with human judgments of quality. The technique was able to adapt well for the challenging datasets, with natural lighting, camera movement, and with objects and persons moving within the scene. This indicates that our technique can both enforce a smooth disparity variation on the background objects while allowing it to change gradually for moving objects.

6.1 Future Work

As future work, we plan to improve the initial triangulation for better capturing the boundaries of the objects. This is a difficult task since there is a trade-off between having a finer granularity on the vertices distribution, necessary to model better the small details of the scene, and generating larger triangles to reduce the matching ambiguities. One possible solution is to use the disparity information of past frames to distinguish highly textured areas, which have smooth disparity, from the objects boundaries that occur on disparity discontinuities.

Since the quality of the triangulation will affect the disparity estimation process, e.g. small triangles will have problems on the region matching process, we plan to evaluate the correlation of the triangles quality with the correctness of the obtained disparity. This information will be important to decide if some improvements can be made on the triangulation step, and more importantly, where the problematic triangles occur.

Another improvement that can be made is to generate a single 3D mesh using jointly all the cameras, instead of estimating one for each view. This can be done by firstly creating the mesh of the central camera and then updating it according to the neighboring cameras (spatial domain), similarly to the temporal domain adaptation presented in Chapter 4. By having a single 3D mesh that is computed from all the cameras, the disparity estimation can be further improved by using the matching information of all the views. Also in the temporal mesh update scheme, it could be possible to use the optical flow from all the cameras, alleviating the problem of unreliable optical flow values on occluded/disoccluded regions.

One extension of this work could be the codification of DIBR videos. In fact, image domain triangulations have been used in the past for image compression (BOUGLEUX; PEYRE; COHEN, 2009), and disparity information could be easily encoded in the context of this thesis. The tradeoff between the quality of the disparity and the size of the encoded disparity map can be controlled by the number of vertices, i.e. with a small number of vertices the disparity will be a more rude representation of the scene, however it will contain less information to encode.

Still regarding DIBR videos, this technique can be used for baseline-retargeting, i.e. changing the baseline distance between the two cameras on a stereo video. There is a relationship between the baseline of the camera acquisition system, the size of the screen where the content will be displayed and the distance of the observer from the screen (RAYMOND; NIGEL., 1953; LAMBOOIJ; IJSSELSTEIJN; FORTUIN, 2009). Hence, a 3D content designed for display on a 70 ft. movie screen may not be adequate for a 46" television. Since the camera array used to acquire the images is usually fixed, i.e. the position between the cameras does not change over time, synthetic views generated with different baselines (distance between the cameras) can be used to adjust the content to different displays.

REFERENCES

ASTOLA, J.; HAAVISTO, P.; NEUVOS, Y. Vector median filters. **Proceedings of the IEEE**, v. 78, p. 678–689, 1990.

BAKER, H. H.; TANGUAY, D. A multi-imager camera for variable-definition video (XDTV). In: **Proceedings of International Conference on Multimedia Content Representation, Classification and Security**. [S.l.: s.n.], 2006. p. 594–601.

BERNARDINI, F. et al. The ball-pivoting algorithm for surface reconstruction. **IEEE Transactions on Visualization and Computer Graphics**, v. 5, n. 4, p. 349–359, 1999.

BLEYER, M.; GELAUTZ, M. Temporally consistent disparity maps from uncalibrated stereo videos. In: **Image and Signal Processing and Analysis, 2009. ISPA 2009. Proceedings of 6th International Symposium on**. [S.l.: s.n.], 2009. p. 383–387. ISSN 1845-5921.

BLEYER, M. et al. Object stereo – joint stereo matching and object segmentation. In: **IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2011. p. 3081–3088.

BOGAERT, J. et al. Alternative area-perimeter ratios for measurement of 2d shape compactness of habitats. **Applied Mathematics and Computation**, v. 111, n. 1, p. 71 – 85, 2000. ISSN 0096-3003. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0096300399000752>.

BOUGLEUX, S.; PEYRE, G.; COHEN, L. Image compression with anisotropic triangulations. In: **Computer Vision, 2009 IEEE 12th International Conference on**. [S.l.: s.n.], 2009. p. 2343–2348. ISSN 1550-5499.

BRADSKI, G. Opencv library. **Dr. Dobb's Journal of Software Tools**, 2000.

BROX, T. et al. High accuracy optical flow estimation based on a theory for warping. In: **European Conference on Computer Vision**. [S.l.]: Springer, 2004. p. 25–36.

CALAKLI, F.; TAUBIN, G. SSD: Smooth signed distance surface reconstruction. **Computer Graphics Forum**, v. 30, n. 7, p. 1993–2002, 2011.

CANNY, J. A computational approach to edge detection. **Pattern Analysis and Machine Intelligence, IEEE Transactions on**, PAMI-8, n. 6, p. 679–698, Nov 1986. ISSN 0162-8828.

CHEN, D. et al. An improved non-local cost aggregation method for stereo matching based on color and boundary cue. In: **Multimedia and Expo (ICME), 2013 IEEE International Conference on**. [S.l.: s.n.], 2013. p. 1–6. ISSN 1945-7871.

CHEW, L. P. Constrained delaunay triangulations. In: **Proceedings of the Third Annual Symposium on Computational Geometry**. New York, NY, USA: ACM, 1987. (SCG '87), p. 215–222. ISBN 0-89791-231-4. Available from Internet: <http://doi.acm.org/10.1145/41958.41981>.

DOLLÁR, P.; ZITNICK, C. L. Structured forests for fast edge detection. In: **ICCV**. [S.l.: s.n.], 2013.

FABRI, A.; CACCIOLA, F.; WEIN, R. Dense point trajectories by gpu-accelerated large displacement optical flow. In: DANIILIDIS, K.; MARAGOS, P.; PARAGIOS, N. (Ed.). **CGAL User and Reference Manual**. CGAL Editorial Board, 2014, (Lecture Notes in Computer Science, v. 6311). p. 438–451. Available from Internet: <http://doc.cgal.org/4.5/Manual/packages.html#PkgBGLSummary>.

FICKEL, G. et al. Stereo matching and view interpolation based on image domain triangulation. **Image Processing, IEEE Transactions on**, v. 22, n. 9, p. 3353–3365, Sept 2013. ISSN 1057-7149.

FICKEL, G. et al. Stereo matching based on image triangulation for view synthesis. In: **Proc. of the 19th IEEE International Conference on Image Processing**. Orlando, USA: IEEE, 2012. p. 2733–2736.

FUHR, G. et al. An evaluation of stereo matching methods for view interpolation. In: **Proc. of the 20th IEEE International Conference on Image Processing**. Melbourne, Australia: IEEE, 2013. p. 2733–2736.

HIRSCHMULLER, H.; SCHARSTEIN, D. Evaluation of cost functions for stereo matching. In: **Proceedings of IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2007. p. 1–8.

HORN, B. K.; SCHUNCK, B. G. Determining optical flow. **Artificial Intelligence**, v. 17, n. 1–3, p. 185 – 203, 1981. ISSN 0004-3702. Available from Internet: <http://www.sciencedirect.com/science/article/pii/0004370281900242>.

HOSNI, A. et al. Temporally consistent disparity and optical flow via efficient spatio-temporal filtering. In: HO, Y.-S. (Ed.). **Advances in Image and Video Technology**. Springer Berlin Heidelberg, 2012, (Lecture Notes in Computer Science, v. 7087). p. 165–177. ISBN 978-3-642-25366-9. Available from Internet: <http://dx.doi.org/10.1007/978-3-642-25367-6_15>.

HUGUET, F.; DEVERNAY, F. A variational method for scene flow estimation from stereo sequences. In: **Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on**. [S.l.: s.n.], 2007. p. 1–7. ISSN 1550-5499.

HUNG, C.; XU, L.; JIA, J. Consistent binocular depth and scene flow with chained temporal profiles. **International Journal of Computer Vision**, Springer US, v. 102, n. 1-3, p. 271–292, 2013. ISSN 0920-5691. Available from Internet: <http://dx.doi.org/10.1007/s11263-012-0559-y>.

JARVIS, R. A perspective on range finding techniques for computer vision. **Pattern Analysis and Machine Intelligence, IEEE Transactions on**, PAMI-5, n. 2, p. 122–139, March 1983. ISSN 0162-8828.

KAZHDAN, M.; BOLITHO, M.; HOPPE, H. Poisson surface reconstruction. In: **Proceedings of the fourth Eurographics symposium on Geometry processing**. [S.l.: s.n.], 2006. p. 61–70.

LAMBOOIJ, M.; IJSSELSTEIJN, W.; FORTUIN, M. Visual discomfort and visual fatigue of stereoscopic displays: A review. **Journal of Imaging Technology and Science**, v. 53, p. 1–14, 2009.

LANG, M. et al. Practical temporal consistency for image-based graphics applications. **ACM Trans. Graph.**, ACM, New York, NY, USA, v. 31, n. 4, p. 34:1–34:8, jul. 2012. ISSN 0730-0301. Available from Internet: <http://doi.acm.org/10.1145/2185520.2185530>.

LEMPITSKY, V.; ROTHER, C.; BLAKE, A. Logcut - efficient graph cut optimization for markov random fields. In: **Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on**. [S.l.: s.n.], 2007. p. 1–8. ISSN 1550-5499.

LEVOY, M.; HANRAHAN, P. Light field rendering. In: **Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques**. New York, NY, USA: ACM, 1996. (SIGGRAPH '96), p. 31–42. ISBN 0-89791-746-4. Available from Internet: <http://doi.acm.org/10.1145/237170.237199>.

MEI, X. et al. On building an accurate stereo matching system on graphics hardware. In: **Proceedings of Workshop on GPUs for Computer Vision (ICCV Workshops)**. [S.l.: s.n.], 2011. p. 467 –474.

MIDDLEBURY. **Middlebury's Database**. 2012. <http://vision.middlebury.edu/stereo/data/>.

MIN, D.; LU, J.; DO, M. Joint histogram-based cost aggregation for stereo matching. **Pattern Analysis and Machine Intelligence, IEEE Transactions on**, v. 35, n. 10, p. 2539–2545, Oct 2013. ISSN 0162-8828.

MIN, D. et al. Disparity search range estimation: Enforcing temporal consistency. In: **Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on**. [S.l.: s.n.], 2010. p. 2366–2369. ISSN 1520-6149.

MORI, Y. et al. View generation with 3D warping using depth information for FTV. **Signal Processing: Image Communication**, v. 24, n. 1–2, p. 65 – 72, 2009.

MOZEROV, M.; WEIJER, J. van de. Accurate stereo matching by two-step energy minimization. **Image Processing, IEEE Transactions on**, v. 24, n. 3, p. 1153–1163, March 2015. ISSN 1057-7149.

MüLLER, K. et al. Coding and intermediate view synthesis of multiview video plus depth. In: **Proceedings of IEEE International Conference on Image Processing**. [S.l.: s.n.], 2009. p. 741–744.

NDJIKI-NYA, P. et al. Depth image based rendering with advanced texture synthesis. In: **Proceedings of IEEE International Conference on Multimedia and Expo**. [S.l.: s.n.], 2010. p. 424 –429. ISSN 1945-7871.

OLIVEIRA, A. et al. Selective hole-filling for depth-image based rendering. In: **Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on**. [S.l.: s.n.], 2015.

OTSU, N. A threshold selection method from gray-level histograms. **IEEE Transactions on Systems, Man and Cybernetics**, v. 9, n. 1, p. 62–66, 1979.

PUJADES, S.; DEVERNAY, F. Viewpoint interpolation: Direct and variational methods. In: **Image Processing (ICIP), 2014 IEEE International Conference on**. [S.l.: s.n.], 2014. p. 5407–5411.

RABINER, L. A tutorial on hidden markov models and selected applications in speech recognition. **Proceedings of the IEEE**, v. 77, n. 2, p. 257–286, Feb 1989. ISSN 0018-9219.

RAYMOND, S.; NIGEL., S. Book. **The theory of stereoscopic transmission and its application to the motion picture**. [S.l.]: University of California Press Berkeley, 1953. 177 p. p.

RHEMANN, C. et al. Fast Cost-Volume Filtering for Visual Correspondence and Beyond. In: **Proceedings of IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2011. p. 3017–3024.

RICHARDT, C. et al. Real-time spatiotemporal stereo matching using the dual-cross-bilateral grid. In: **Proceedings of European conference on computer vision: Part III**. [S.l.: s.n.], 2010. p. 510–523.

SAND, P.; TELLER, S. Particle video: Long-range motion estimation using point trajectories. **Int. J. Comput. Vision**, Kluwer Academic Publishers, Hingham, MA, USA, v. 80, n. 1, p. 72–91, oct. 2008. ISSN 0920-5691. Available from Internet: <http://dx.doi.org/10.1007/s11263-008-0136-6>.

SCHARSTEIN, D.; SZELISKI, R. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. **International Journal of Computer Vision**, v. 47, p. 7–42, 2002.

SEITZ, S. M.; DYER, C. R. View morphing. In: **Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques**. New York, NY, USA: ACM, 1996. (SIGGRAPH ’96), p. 21–30. ISBN 0-89791-746-4. Available from Internet: <http://doi.acm.org/10.1145/237170.237196>.

SIZINTSEV, M.; WILDES, R. Spatiotemporal stereo via spatiotemporal quadric element (stequel) matching. In: **Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on**. [S.l.: s.n.], 2009. p. 493–500. ISSN 1063-6919.

SIZINTSEV, M.; WILDES, R. Spacetime stereo and 3d flow via binocular spatiotemporal orientation analysis. **Pattern Analysis and Machine Intelligence, IEEE Transactions on**, v. 36, n. 11, p. 2241–2254, Nov 2014. ISSN 0162-8828.

SOLH, M.; ALREGIB, G. Hierarchical hole-filling for depth-based view synthesis in ftv and 3d video. **Selected Topics in Signal Processing, IEEE Journal of**, v. 6, n. 5, p. 495–504, Sept 2012. ISSN 1932-4553.

SOUNDARARAJAN, R.; BOVIK, A. Video quality assessment by reduced reference spatio-temporal entropic differencing. **Circuits and Systems for Video Technology, IEEE Transactions on**, v. 23, n. 4, p. 684–694, April 2013. ISSN 1051-8215.

SUN, X. et al. Real-time local stereo via edge-aware disparity propagation. **Pattern Recognition Letters**, v. 49, n. 0, p. 201 – 206, 2014. ISSN 0167-8655. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S016786551400227X>.

SUNDARAM, N.; BROX, T.; KEUTZER, K. Dense point trajectories by gpu-accelerated large displacement optical flow. In: DANIILIDIS, K.; MARAGOS, P.; PARAGIOS, N. (Ed.). **Computer Vision – ECCV 2010**. Springer Berlin Heidelberg, 2010, (Lecture Notes in Computer Science, v. 6311). p. 438–451. Available from Internet: <http://dx.doi.org/10.1007/978-3-642-15549-9_32>.

SZELISKI, R. **Computer Vision: Algorithms and Applications**. 1st. ed. New York, NY, USA: Springer-Verlag New York, Inc., 2010. ISBN 1848829345, 9781848829343.

TAGUCHI, Y.; WILBURN, B.; ZITNICK, C. Stereo reconstruction with mixed pixels using adaptive over-segmentation. In: **IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2008. p. 1 –8.

TANIMOTO, M.; FUJII, T.; SUZUKI, K. **View Synthesis Algorithm in View Synthesis Reference Software 2.0 (VSRS2.0)**. 2009. MPEG document M16090.

TIAN, D.; VETRO, A.; BRAND, M. A trellis-based approach for robust view synthesis. In: **Proceedings of IEEE International Conference on Image Processing**. [S.l.: s.n.], 2011. p. 605–608.

YANG, Q. A non-local cost aggregation method for stereo matching. In: **Proceedings of IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2012. p. 1402–1409.

YANG, Q. Hardware-efficient bilateral filtering for stereo matching. **PAMI**, pp, n. 99, 2013.

YANG, Q. Stereo matching using tree filtering. **Pattern Analysis and Machine Intelligence, IEEE Transactions on**, 2015.

YOON, K.-J.; KWEON, I. S. Adaptive support-weight approach for correspondence search. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 28, n. 4, p. 650 –656, april 2006. ISSN 0162-8828.

ZHANG, K.; LU, J.; LAFRUIT, G. Cross-based local stereo matching using orthogonal integral images. **IEEE Transactions on Cicuits and Systems for Video Technology**, v. 19, n. 7, p. 1073–1079, jul. 2009.

ZHANG, Q.; XU, L.; JIA, J. 100+ times faster weighted median filter (wmf). In: **Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on**. [S.l.: s.n.], 2014. p. 2830–2837.

ZHAO, Y.; TAUBIN, G. Real-time stereo on GPGPU using progressive multi-resolution adaptive windows. **Image and Vision Computing**, v. 29, n. 6, p. 420 – 432, 2011.

ZITNICK, C. L. et al. High-quality video view interpolation using a layered representation. In: **ACM SIGGRAPH**. [S.l.: s.n.], 2004. p. 600–608.