Technische Universität München
Fakultät für Elektrotechnik und Informationstechnik
Lehrstuhl für Datenverarbeitung

Hole-Filling Algorithms for Depth-Image-Based Rendering

Julian Albert Habigt

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzende(r): Prof. Dr. Oliver Hayden

Prüfer der Dissertation:

1. Prof. Dr.-Ing. Klaus Diepold
2. apl. Prof. Dr.-Ing. Walter Stechele

Die Dissertation wurde am 18.06.2020 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 07.09.2020 angenommen.

Julian Albert Habigt. *Hole-Filling Algorithms for Depth-Image-Based Rendering.* Dissertation, Technische Universität München, Munich, Germany, 2020.

Abstract

View synthesis is a process for generating novel views from a scene which has been recorded with a 3-D camera setup. It has important applications in 3-D post-production and 2-D to 3-D conversion. Being able to modify the baseline of a stereo camera setup after a movie has been recorded is crucial for the correct representation of depth at the site of the observer. View synthesis is also the only practical way to generate the many views which are required for autostereoscopic displays which enable more than one viewer to watch a movie at the same time. However, virtual view synthesis generates holes in the output image due to disocclusions, which appear when background content which was occluded by foreground objects becomes unveiled in the virtual view. How to properly fill in these holes is a question of great research interest because it is one of the most important factors to determine the quality of the output. In this thesis, I provide a comprehensive review of the state of the art in hole-filling algorithms and propose two novel hole-filling algorithms which are able to fill in disocclusions in a visually plausible way. The design goal with the highest priority when developing these algorithms was superior image quality, while still being computationally tractable. The results of both algorithms compare favorably to the ones generated by state-of-the-art hole-filling algorithms in terms of visual quality, which I will demonstrate with suitable objective quality metrics, some of which have been tailored to analyze synthesized views. I substantiate these results with a crowd-sourced subjective study, a novel method for evaluating virtual views which I will validate against an independent laboratory study.

Acknowledgments

First and foremost, I would like to thank Prof. Dr.-Ing. Klaus Diepold for giving me the opportunity to work at his chair, giving me the freedom to conduct the research for this thesis, for all his personal support and motivation, for the excellent team which he congregated and for creating an inspiring and motivational culture. I will always look back fondly at all the memories of my years at the LDV. I would like to thank Prof. Dr.-Ing. Walter Stechele for reviewing this thesis, for giving me the opportunity to present my research at his seminar and for his constructive feedback. I would like to thank Prof. Dr. Oliver Hayden for chairing the defense of my thesis. Thanks to all the bright colleagues at LDV and GOL for the great time we had together, especially Christian for mentoring and supporting me in this research topic, Clemens, Dominik, Hao, Johannes, Marko, Matthias, Simon and Uli for all the fun times and inspiring technical discussions, Martin R. for always lending me an open ear or borrowing mine, Martin K. for personal and professional support, Flo for introducing me to the chair, and Ricarda and Michi for navigating me through the administrative jungle a university may present. My heartfelt thanks goes to my family and friends for their continuing support through all these years, especially my brother Tim being such a great companion and Dominik for never stopping to motivate me to finish this thesis, and my parents for always supporting me with everything they can offer. Lastly, my special thanks go to Christin for always being at my side, for your loving support and your first-hand understanding for the privations necessary for completing a doctoral thesis.

Contents

List of Figures

List of Tables

List of Abbreviations

3DSwIM	3-D Synthesized View Image Quality Metric
ACR	Absolute Category Rating
ANN	Approximate Nearest Neighbour
ATTEST	Advanced Three-dimensional Television System Technologies
CSH	Coherency Sensitive Hashing
DERS	Depth Estimation Reference Software
DIBR	Depth-Image-Based Rendering
DSIS	Double Stimulus Impairment Scale
ETRI	Electronics and Telecommunications Research Institute
FTV	Free-viewpoint Television
GIST	Gwangju Institute of Science and Technology
IBR	Image-Based Rendering
IST	Information Society Technologies
IQA	Image Quality Assessment
LSH	Locality Sensitive Hashing
MOS	Mean Opinion Score
MPEG	Moving Picture Experts Group
MP-PSNR	Morphological Pyramid Peak Signal-to-Noise Ratio

MRF Markov Random Field

MSR Microsoft Research

MS-SSIM Multi Scale SSIM

MVC Multi-View (Video) Coding

MW-PSNR Morphological Wavelets Peak Signal-to-Noise Ratio

PC Pairwise Comparison

PQM Perceptual Video Quality Metric

PSNR Peak Signal-to-Noise Ratio

SIQE Synthesized Image Quality Evaluator

SSIM Structural Similarity Index Metric

VQEG Visual Quality Experts Group

VS-BP View Synthesis with Belief Propagation

VS-CSH View Synthesis with Coherency Sensitive Hashing

VSNR Visual Signal-to-Noise Ratio

VSRS View Synthesis Reference Software

1. Introduction and Motivation

1.1. The State of the Art in 3-D Video Reproduction and Broadcasting

3-D television is dead, for now. In 2018, Samsung as one of the last manufacturers of 3-D television sets decided to no longer produce new models with 3-D functionality. After the cinematic release of the movie Avatar in the year 2010, 3-D technology seemed to be on a steady rise. For the first time, high quality 3-D stereoscopic content was produced by movie studios and with the increasing dissemination of HD television sets capable of reproducing this content, 3-D soon became a commodity feature of every TV set produced by the large manufacturers. However, the actual adoption of the technology at home was much lower than anticipated. While customers still seem to be willing to pay a premium for watching a cinema movie in 3-D at the movie theaters, the adoption at home never really took of, resulting in its eventual demise [1]. The reasons for this course of events are probably too diverse to capture them here completely. The need to wear glasses, low availability of high quality content and bad 2-D to 3-D conversion are some of the most frequently mentioned reasons [2]. However, there are still major technological issues which have not been solved yet and which probably contributed to this decline.

Currently, common 3-D video technology is based on stereo systems, i.e., during production, the content is being filmed with two cameras and these two views are then reproduced at the receiver. The reproduction with stereoscopic displays requires the user to wear glasses so that the two views can be separated and displayed independently for each eye. Different technologies exist for separating the views, either with active shutter glasses which synchronize to the refresh rate of the display to facilitate temporal multiplexing of the views, matched polarization filters in the glasses and the display to separate the images

based on their polarization, or very narrow-band color filters in the glasses which utilize minimal wavelength differences in the reference color stimuli of the display.

In any case, the quality of the separation of the two views has a significant influence on the quality of experience of the 3-D representation, as an insufficient separation leads to ghosting artifacts in the view. Even if one would achieve a perfect separation of the views, a major drawback leading to viewer discomfort lies in an incorrect baseline of the two views. 3-D content is typically produced and optimized for the reproduction on large cinema screens. Depending on the size of the TV screen and the distance of the observer from the screen, this baseline may be incorrect which leads to an incorrect representation of depth [3], and as a result, viewer discomfort [4]. To solve this problem, one or both of the

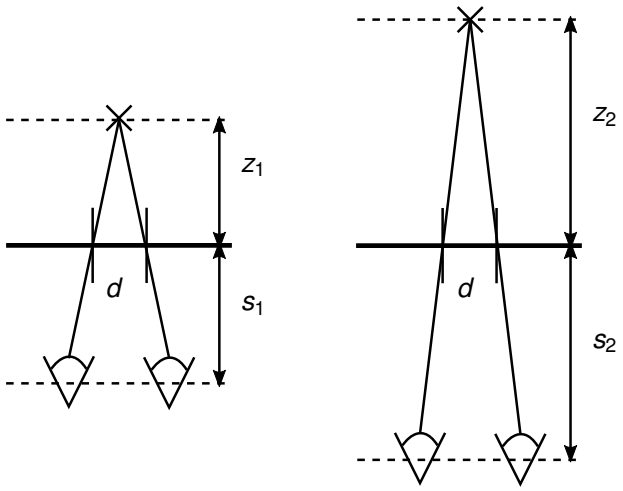


Figure 1.1.: Given a fixed disparity d, a change in the viewing distance of the observer from s_1 to s_2 changes the perceived depth from z_1 to z_2. This leads to a misrepresentation of the depth of the objects in the scene. Similarly, a change in the physical size of the screen might lead to a change in d, e.g. when displaying cinema content on a home screen.

two views need to be repositioned depending on the distance of the observer, a technique known as stereo repurposing [5].

The fact that one has to wear glasses in itself creates a discomfort for viewers which aren't accustomed to wearing these glasses. Autostereoscopic displays promise to rid the user from having to wear glasses by separating the different views spatially. By using lenticular lenses or parallax barriers in front of the display, the user will see different pictures from different viewing angles with respect to the display. While this enables a 3-D reproduction

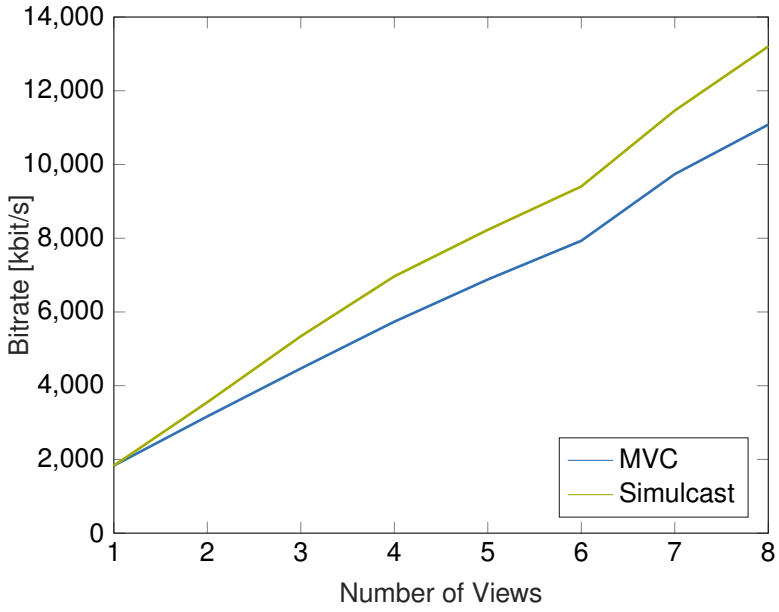


Figure 1.2.: This plot shows the cumulative bitrate required for the transmission of multiple views with the same PSNR. Even though Multi-View (Video) Coding (MVC) reduces the bitrate required for the transmission significantly compared to simulcast, there is still a linear increase with the number of views. Data from [7].

without glasses, the user has to be at a certain spot and at a certain distance in front of the display for this technique to work. If one wants to create a display where the user has more freedom to move in front of the display or where multiple viewers sitting next to each other can watch at the same time, the autostereoscopic display must reproduce many more than only two views of the same scene. In fact, current prototypes of autostereoscopic displays require up to 70 views of the same scene [6].

Transmitting such a high number of views through traditional broadcasting channels increases the required bitrate significantly. Multiview extensions of existing stereo video codecs, while more efficient than transmitting the views individually, still require a bit rate which typically increases linearly with the number of views as shown in Figure 1.2.

1.2. View Synthesis and its Applications

Many of the technical problems which I've described so far can be solved by view synthesis. View synthesis allows the generation of an arbitrary number of views from varying viewpoints with only a limited set of input views. If done at the receiver, this enables the display to adapt the baseline on the fly to the required baseline at the location of the observer and, for multi-view autostereoscopic displays, to produce all required views from only a limited stream of information. One technique to generate such virtual views that has gained momentum in recent years is called Depth-Image-Based Rendering (DIBR). There, the virtual view is generated from the image of one or more cameras and corresponding depth maps.

1.3. Limitations of Existing View Synthesis Systems

A central problem in the generation of novel views lies in the handling of disocclusions. Background content, which was occluded in the original view by objects that were closer to the camera, may become unveiled in the virtual view. In a setup with two or more cameras, these so-called disocclusions may be partially filled with content from another camera, yet some disocclusions usually still remain [8]. Figure 1.3 shows a schematic example of how these holes emerge. Even more challenging, when there is only one view and a corresponding depth map, there is no other other information available to fill the holes in the resulting view and the holes may have to be filled with synthetically generated content. This is, for example, the case in 2-D to 3-D conversion or in transmission schemes with only one texture and depth map such as the one proposed in the Advanced Three-dimensional Television System Technologies (ATTEST) project [9, 10]. ATTEST was a project of the Information Society Technologies (IST) programme by the European Commission which task was to develop a commercially feasible 3D-TV system which should be backwards compatible to existing 2-D transmission schemes. The outcome was a transmission chain which combined a conventional MPEG-2 encoded 2-D base layer with an MPEG-2 or MPEG-4 encoded 3-D enhancement layer which basically consisted of the depth-map with some additional meta data. This makes it possible to either display the 2-D stream on a 2-D television screen or use a 3-D TV capable broadcast decoder which

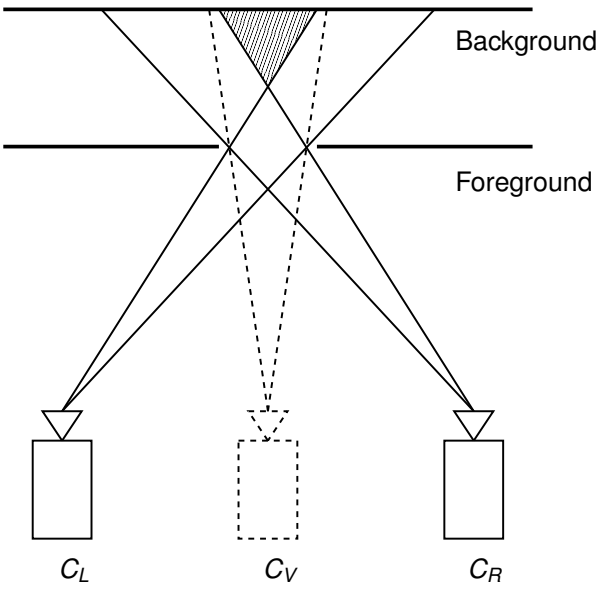


Figure 1.3.: The disocclusion appears in areas where the background is occluded by foreground content in the original view. Even with multiple views C_L and C_R, e.g. from a stereo camera setup, there can be areas in a virtual view C_V which are not covered by any of the two cameras.

would perform depth-image-based rendering to create the required views for single- or multi-user 3-D TVs.

A number of solutions have been proposed for the problem of handling disocclusions. Most of the algorithms can be classified into one of two categories. The first category sets out to mitigate the problem by eliminating the root cause. These algorithms modify the depth map in a way so that no holes in the synthesized views appear. Disocclusions appear at regions in the image where there is a steep gradient in the depth-map, i.e., at the borders of foreground objects. Zhang et al. [11] proposed a technique where the depth-map is filtered to remove these steep gradients. The result is a virtual view which doesn't contain any holes, at the cost of an incorrect reproduction of the depth which may lead to visible errors [12]. Fehn et al. [10] proposed a very similar method with a Gaussian low-pass filter to smooth the depth image. Another way is the use of so-called inpainting techniques. Inpainting describes a process where holes in images are filled with synthesized content in a visually plausible manner so that the viewer doesn't recognize that the content has been generated artificially.

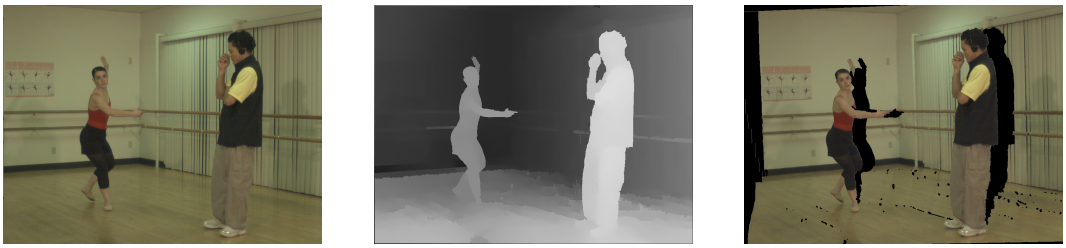


Figure 1.4.: Given an input image and a corresponding depth-map, one can generate virtual views, a process which is called depth-image-based rendering (DIBR). However, disocclusions appear where background is unveiled which was occluded by foreground objects in the original view.

1.4. Formulation of the Research Problem

Hole filling remains one of the most central problems in view synthesis. How a view synthesis algorithm deals with disocclusions is probably one of the most significant influencing factors of the quality of the synthesized output. While there are several very promising approaches in literature, many of those exhibit significant weaknesses in image quality, computational requirements or both, as I'm going to show in this thesis. How does one even evaluate the image quality of synthesized views? Is it possible to objectively quantify whether an image looks *natural* to an observer? In literature, there does not seem to be an agreement on how to answer this question except resorting back to PSNR. In this thesis, I therefore want to develop algorithms for hole filling in view synthesis and properly investigate methods to evaluate their performance in terms of visual quality.

1.5. Contributions

In my thesis, I'm presenting two novel approaches for handling the disocclusion problem in DIBR. Disocclusion handling is closely related to the image inpainting problem. However, no existing inpainting algorithm delivers satisfying results when directly applied to the disocclusion problem. Therefore, I'm going to give an extensive review of the image inpainting field, explaining the challenges in applying some of the most promising algorithms to DIBR and present novel solutions on how to extend these algorithms for the aforementioned research problems. As I'm going to present in more detail over the course of this thesis, state-of-the-art research in this field can be categorized into several classes of different al-

gorithms, e.g., exemplar-based techniques or learning algorithms. I've made contributions to two main classes and present the results of these algorithms, each surpassing the state of the art in DIBR algorithms. Furthermore, I'm going to review which metrics are available to quantify the visual quality of virtual views, evaluate their accuracy, and propose a new method to conduct subjective studies to evaluate the quality of virtual views with the online crowdsourcing tool QualityCrowd, thereby reducing the effort and time required to evaluate these algorithms.

1.6. Outline

Part I of this thesis covers the fundamentals of view synthesis. In Chapter 2 I'm going to show the process of generating novels views with depth-image-based rendering and explain the preliminaries needed for this process. Part II then focuses on the problem of hole filling in detail. In Chapter 3 I'm going to provide an overview over the state of the art in hole-filling techniques with a survey of view synthesis algorithms which address this problem. Diving deeper into suitable inpainting algorithms to synthesize content and possible challenges that arise when applying them to the hole-filling problems, I'm going to develop and propose two novel methods for hole filling in Chapter 4. Finally, in Part III, I'm addressing the problem of the evaluation of these algorithms. I'm starting with a review of existing objective metrics and evaluate their suitability for the kind of artifacts which can be introduced by the view synthesis process in Chapter 5. In Chapter 6 I'm presenting an online subjective study on the image quality, comparing its results to a traditional lab study and, of course, evaluate the how the proposed algorithms perform. Lastly, I'm going to wrap up this thesis with a conclusion in Chapter 7.

Part I.

View Synthesis

2. Depth-Image-Based Rendering

2.1. Introduction

Image-Based Rendering (IBR) is a result of the convergence between the two classical fields of Computer Vision and Computer Graphics which started more than 20 years ago [13]. Instead of trying to model a virtual scene in 3-D and projecting two-dimensional views, image-based rendering is a process where novel views are generated from one or more views which have been captured by cameras. Because photorealistic rendering of computer-generated content requires significant modelling efforts and can only be achieved with raytracing techniques which incur tremendous computational costs, it seems appealing to use captured images as the basis for the rendering of virtual views. Depth-Image-Based Rendering (DIBR) amends the captured image by a depth image (often called a depth map) to get the necessary depth information for the synthesis of novel views.

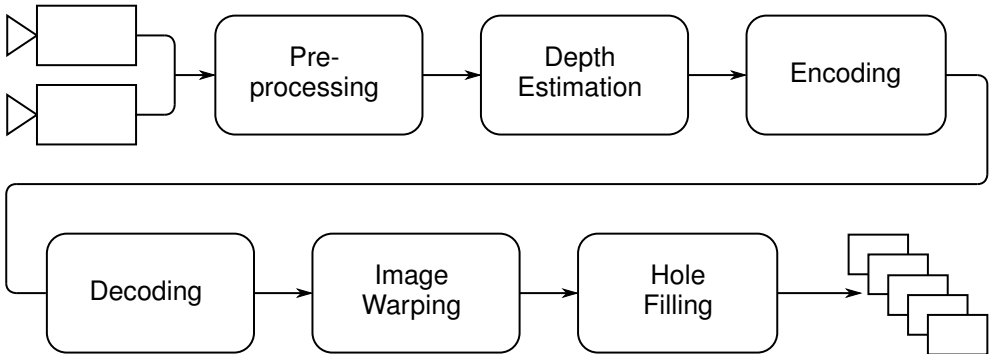


Figure 2.1.: Signal flow from the capture of the content with a stereo camera pair, the pre-processing steps, the depth estimation, encoding, transmission, decoding, and finally the DIBR step.

Figure 2.1 shows a very abstract and simplified signal flow for a depth-image-based ren-

dering chain, including the transmission from the site where the scene is recorded to the receiver where different views of the scene are reproduced. In the first step, the scene is recorded by two or more cameras from different viewpoints which ideally have been synchronized to capture the images at identical time instances. The captured images then get pre-processed which includes color matching to deal with differences in color rendition of the cameras and rectification if the camera setup is approximately parallel. In the next step, the depth map, if it has not been captured directly by the cameras through a time-of-flight measurement, is estimated from the disparity between the different images. Typically, further processing of the depth map is then necessary to get the required dense depth maps of sufficient quality for depth-image-based rendering. The images and depth maps then get encoded, transmitted and decoded at the receiver. There, the actual DIBR step takes place with image warping and hole filling to render the desired views from the captured scene which can then be displayed.

In the following chapters of this thesis, I want to give a brief overview over the steps in this chain, in every step explain the techniques that I've applied to create the views that are used during this thesis, give a short overview into the state of the art in the respective research field of each step and of course introduce my contributions in the hole-filling step.

2.2. Image Warping

2.2.1. Fundamentals

Image Warping describes the process of transforming the image of one camera into a virtual camera at a different viewpoint. To this end, the points in the image of the original camera have to be mapped into 3-D space (Mark calls this process an "un-projection" [14]) and then projected back into the 2-D image plane of the virtual camera.

For a mathematical description of this process, I have to take a short excursion into the field of projective geometry because it allows for a rather elegant representation of the underlying mechanisms. Figure 2.2 shows a typical setup with two cameras oriented such

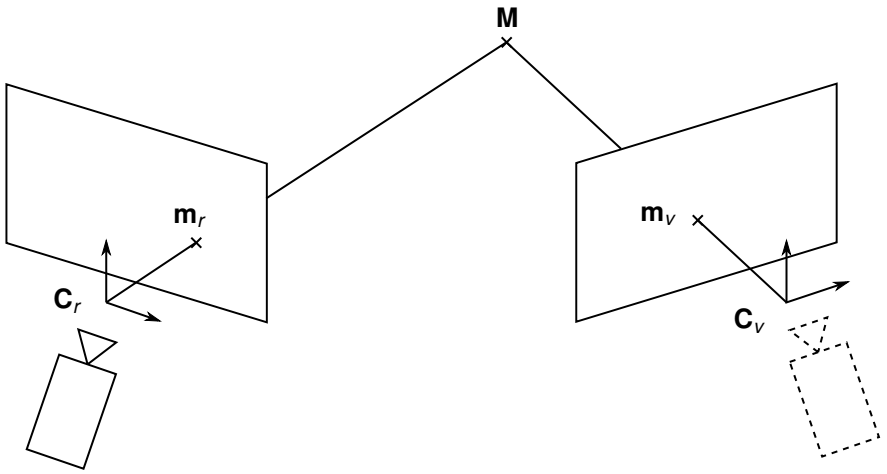


Figure 2.2.: Geometry of a setup with two cameras. The left camera is the reference camera; the right one is the virtual view. The camera centers are at $\mathbf{C}_r$ and $\mathbf{C}_v$, respectively. Both cameras are looking at the same point $\mathbf{M}$ in 3-D space which leads to the image points $\mathbf{m}_r$ and $\mathbf{m}_v$ in the camera planes.

that both cameras can see a point

$$\mathbf{M} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

in the world coordinate system and its projection

$$\mathbf{m}_r = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

in the image plane of the reference camera.

Using projective geometry, the relationship between both points can be formulated as a linear equation

$$\mathbf{m}_r \simeq \mathcal{P}_r \mathbf{M} \tag{2.1}$$

where $\mathcal{P}_r$ is the projection matrix of the reference camera and $\simeq$ denotes equality up to a constant scaling factor. In general, the projection matrix

$$\mathcal{P}_r \simeq \mathbf{K}_r \begin{bmatrix} \mathbf{R}_r & \mathbf{t}_r \end{bmatrix}$$

is composed of the intrinsic camera parameters

$$\mathbf{K} = 1 \begin{bmatrix} \alpha_u & \gamma & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

with the principal point $[\,u_0\ v_0\,]^T$, the scale factors α_u and α_v, the skew factor γ, and the extrinsic parameters **R** and **t** which define the orientation (rotation and translation) with respect to the world coordinate system which are sometimes combined to a displacement matrix

$$\mathbf{D} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}.$$

Most often, it is more convenient to directly specify the point in space **C** where the camera is located and the camera's orientation $\mathbf{R}_C$. The relationship between both representations can simply be deduced as

$$\begin{aligned} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_3^T & 1 \end{bmatrix} &= \begin{bmatrix} \mathbf{R}_C & \mathbf{C} \\ \mathbf{0}_3^T & 1 \end{bmatrix}^{-1} \\ &= \left[\begin{bmatrix} \mathbf{I}_3 & \mathbf{C} \\ \mathbf{0}_3^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_C & \mathbf{0}_3 \\ \mathbf{0}_3^T & 1 \end{bmatrix} \right]^{-1} \\ &= \begin{bmatrix} \mathbf{R}_C & \mathbf{0}_3 \\ \mathbf{0}_3^T & 1 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{I}_3 & \mathbf{C} \\ \mathbf{0}_3^T & 1 \end{bmatrix}^{-1} \\ &= \begin{bmatrix} \mathbf{R}_C^T & \mathbf{0}_3 \\ \mathbf{0}_3^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I}_3 & -\mathbf{C} \\ \mathbf{0}_3^T & 1 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{R}_C^T & -\mathbf{R}_C^T\mathbf{C} \\ \mathbf{0}_3^T & 1 \end{bmatrix}, \end{aligned} \tag{2.2}$$

where $\mathbf{I}_3$ denotes the 3×3 identity matrix and $\mathbf{0}_3$ the 3×1 zero-element vector.

When I combine equations 2.1 and 2.2, I get the following formula for the projection

$$\lambda \mathbf{m}_r = \begin{bmatrix} \mathbf{K} & \mathbf{0}_3 \end{bmatrix} \begin{bmatrix} \mathbf{R}_C^T & -\mathbf{R}_C^T\mathbf{C} \\ \mathbf{0}_3^T & 1 \end{bmatrix} \mathbf{M} = \mathbf{K}\mathbf{R}_C \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} - \mathbf{K}\mathbf{R}_C\mathbf{C},$$

where λ denotes the homogeneous scaling factor.

From here, I can derive the back-projection, which is the inverse transformation from the camera plane to world coordinates, as

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \mathbf{C} + \lambda \mathbf{R}_C^{-1} \mathbf{K}^{-1} \mathbf{m}_r, \tag{2.3}$$

which provides a good intuition for the process of the back-projection: from the origin of the camera a ray is cast in the direction of the pixel $\mathbf{m}_l$ with a scaling factor λ which depends on the distance of the object from the camera or the depth of that point.

Similarly, the same applies to the virtual view (in this case the right-hand camera)

$$\mathbf{m}_v \simeq \mathcal{P}_v \mathbf{M}.$$

2.2.2. Forward and Backward Mapping

Applying Equation 2.3 for the reference and the virtual view provides the following equation for mapping a pixel in the reference view $\mathbf{m}_r$ into the virtual view $\mathbf{m}_v$

$$\begin{aligned} \mathbf{C}_v + \lambda_v \mathbf{R}_v^{-1} \mathbf{K}_v^{-1} \mathbf{m}_v &= \mathbf{C}_r + \lambda_r \mathbf{R}_r^{-1} \mathbf{K}_r^{-1} \mathbf{m}_r \\ \mathbf{m}_v &= \frac{\lambda_r}{\lambda_v} \mathbf{K}_v \mathbf{R}_v \mathbf{R}_r^{-1} \mathbf{K}_r^{-1} \mathbf{m}_r + \frac{1}{\lambda_v} \mathbf{K}_v \mathbf{R}_v \left(\mathbf{C}_r - \mathbf{C}_v \right) \end{aligned} \tag{2.4}$$

For the special case of a lateral motion of the camera, only, with a rectified geometry, this equation gets significantly simpler. As both cameras are parallel to each other and facing the same direction, $\mathbf{R}_v = \mathbf{R}_r$, and because both cameras have identical intrinsic properties, $\mathbf{K}_v = \mathbf{K}_r$. With $\lambda = \lambda_r = \lambda_v$, I get

$$\mathbf{m}_v = \mathbf{m}_r + \frac{1}{\lambda} \mathbf{K}_v \mathbf{R}_v \left(\mathbf{C}_r - \mathbf{C}_v \right).$$

I also know that the lateral motion only changes the x-component of $\mathbf{C}_r$ and $\mathbf{C}_v$; the y- and

(a) **(b)**

Figure 2.3.: Two examples of how the forward mapping strategy leads to tiny holes which can be seen all over the warped images (a) and (b). In contrast to the large holes which stem from the disocclusion, these are sampling artifacts due to rounding to the nearest discrete pixel location.

z-component remain the same, which simplifies the equation further to

$$x_v = x_r + \frac{1}{\lambda}\mathbf{K}_v\mathbf{R}_v\left(\mathbf{C}_r - \mathbf{C}_v\right) = x_r + u(x_r), \tag{2.5}$$

where $u(x_r)$ denotes the disparity associated with pixel x_r.

In either case, if Z is known, I can calculate λ as

$$\lambda = \frac{Z - C_z}{\tau_3} \quad \text{with} \quad \boldsymbol{\tau} = \mathbf{R}_v^{-1}\mathbf{K}_v^{-1}\mathbf{m}_v,$$

where C_z is the third component of $\mathbf{C}_v$ and τ_3 denotes the third component of $\boldsymbol{\tau}$.

Given a certain discrete point on the reference image $\mathbf{m}_r$ or x_r, respectively, the resulting point $\mathbf{m}_v$ or x_v. will most probably land at a subpixel location, which requires rounding to get the nearest full-pixel location in the target image. Therefore, this forward mapping of the image into the target image will likely introduce holes because some areas in the target image are not covered when scanning over the reference image. Figure 2.3 shows the sampling artifacts that get introduced by this forward mapping technique.

On the other hand, because I have to round to the nearest target pixel location, it can happen that two or more reference pixels become mapped to the same target pixel location. It is therefore advantageous to store the target depth value in a depth map at the same time as I create the target image, a process which in computer graphics is called z-buffering. I

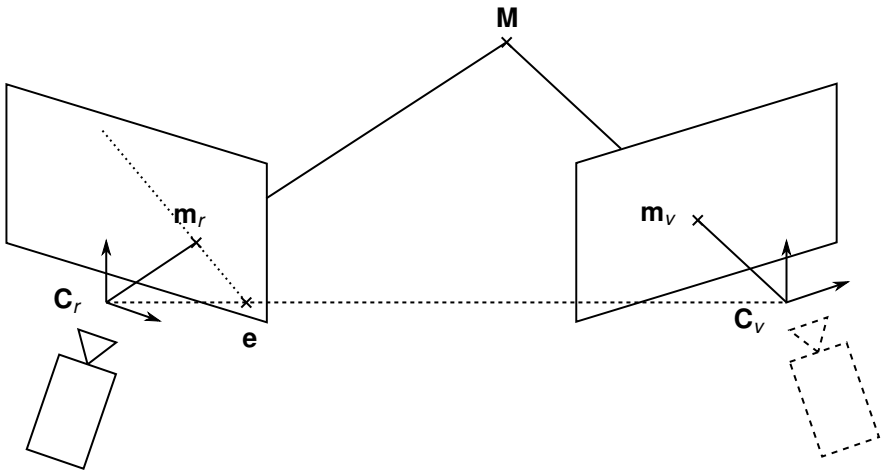


Figure 2.4.: The backward mapping strategy makes use of the epipolar geometry in a two-camera setup. Point **e** is the epipole [15], i.e. the imaginary point of the center of the virtual camera $\mathbf{C}_v$ in the reference camera image. $\mathbf{m}_v$ can be found by searching along the epipolar line $d\mathbf{e}$.

can then compare the two depth values of the pixels that are being mapped to the same location and choose the one which is closer to the virtual camera because this pixel would occlude the the pixel which is further away.

It is possible to avoid the creation of the holes due to the sampling problem of the forward mapping approach by using a technique called backward mapping. There, I'm not scanning over the reference image but rather over the virtual image and search for the respective location in the reference image.

I can rewrite Equation 2.4 to

$$\mathbf{m}_r = \frac{\lambda_v}{\lambda_r}\mathbf{K}_r\mathbf{R}_r\mathbf{R}_v^{-1}\mathbf{K}_v^{-1}\mathbf{m}_v + \frac{1}{\lambda_r}\mathbf{K}_r\mathbf{R}_r\left(\mathbf{C}_v - \mathbf{C}_r\right) \tag{2.6}$$

which is equivalent to

$$\mathbf{m}_r = \mathbf{H}\mathbf{m}_v + d\mathbf{e},$$

where $\mathbf{H} = \mathcal{P}_r^{-1}\mathcal{P}_v$ which defines the homography between the screen plane of the virtual view and the reference camera plane, **e** is the epipole (the point where the line $\mathbf{C}_v - \mathbf{C}_r$ intersects with the reference plane) and d is a scaling factor. Therefore, if I want to find the pixel in the reference image $\mathbf{m}_r$ which corresponds to the pixel in the virtual view $\mathbf{m}_v$, I have to search along the epipolar line $d\mathbf{e}$ for the value which has the minimum depth to

the reference camera. This strategy guarantees that there will be no holes in the virtual view due to the sampling artifacts described in the forward mapping strategy. However, this comes at the computational cost of having to search along the epipolar line.

Applying either of the techniques just presented allows me to create an arbitrary view of a scene as long as the camera parameters are known and as long as the distance of each pixel from the origin of the camera is known.

2.3. Depth Map Generation

A crucial prerequisite for the image warping process is knowing the depth Z at every pixel $\mathbf{m}_r$ of the reference image. Typically, this depth is encoded into a depth map which is a single-channel grayscale image where the depth at each pixel is encoded into the brightness of that pixel. Before I go into detail of how this information is encoded, I want to give a short overview of techniques for the acquisition of this depth information. The quality of this process for the generation of the depth map highly influences the quality of the view synthesis process. Incorrect depth data will necessarily lead to an incorrect generation of the novel view with easily recognizable artifacts. However, due to the numerous challenges in the acquisition of a correct depth map, I consider this to be a research problem on its own (and a still a very active field in that) which lies outside of the scope of this thesis. I will simply consider the depth maps to be given and accept the limitations of the underlying techniques. Nevertheless, I want to briefly give an overview over the different forms of acquisition that exist and present the techniques which have been used to generate the depth maps which I'm using.

2.3.1. Depth Estimation

The most commonly used way to acquire the depth information is to estimate the depth from multiple images which have been recorded from different viewpoints. There, the depth is determined by the disparity between common features of the scene in different images. As I need to obtain a dense depth map (i.e. there is a depth value for every single pixel), the challenge here lies in finding enough correspondences between the pictures

and resolving ambiguities, e.g., if the image content is very homogeneous or consists of repeating structures. Therefore, typically a number of techniques are applied for the regularization of the estimation problem and the post-processing of the depth map.

The latest evaluation on the Middlebury version 3 stereo data set [16] provides a comprehensive overview over different algorithms for creating dense depth maps from stereo pairs. In this thesis, I'm evaluating my algorithms with two sequences which I will present in detail in Section 6.2.2. For the sake of easier comparability against published algorithms, I'm using the depth maps which were published with these two test sequences. For the *Lovebird* sequence, these depth maps were generated using the MPEG Depth Estimation Reference Software (DERS) [17] and in the *Ballet* sequence, an iterative refinement algorithm published by Zitnick et al. [18] was used to calculate the depth maps.

2.3.2. Depth Acquisition

Instead of estimating the depth from stereo pair, there exist a number of techniques to directly capture the depth of a scene. Structured light or coded light methods project patterns of light onto the scene or object and measure the deformation of these patterns with a camera. By using infrared light, these methods can be used invisible to the eye or in conjunction with an RGB camera. The captured depth maps can be of very high quality and are even used for measurement applications, however, the success depends on how well the reflection of the projected pattern can be picked up by the camera, making it susceptible to the influence of ambient light and to the properties of the object itself. Also, this technique is not very well suited for moving objects because a number of different patterns have to be projected in succession to facilitate the measurement of depth maps of sufficient resolution and density.

Time-of-flight cameras don't measure the shape of the reflected pattern, but instead measure the round-trip-time from emitting the light rays until being sensed by the camera. While these systems can capture the depth map very quickly with frame rates of up to 160 Hz, they still suffer from low depth and spatial resolution, low accuracy and relatively high noise [19]. These, too, are susceptible to ambient light [20].

Light field cameras or plenoptic cameras capture not only the intensity of light rays entering the camera but also the direction of these rays. This knowledge enables the calculation

of the depth map [21, 22]. In practice, available light field cameras are using micro-lens arrays in front of the sensor and therefore the task of finding correspondences in these micro-apertures remains the same as in a stereo camera setup, yet, with a very small baseline.

Lastly, laser scanning techniques can also be used for capturing depth maps accurately. However, due to the scanning nature of these methods, there is a trade-off between the capturing speed and the density of the acquired depth map.

2.3.3. Depth Encoding

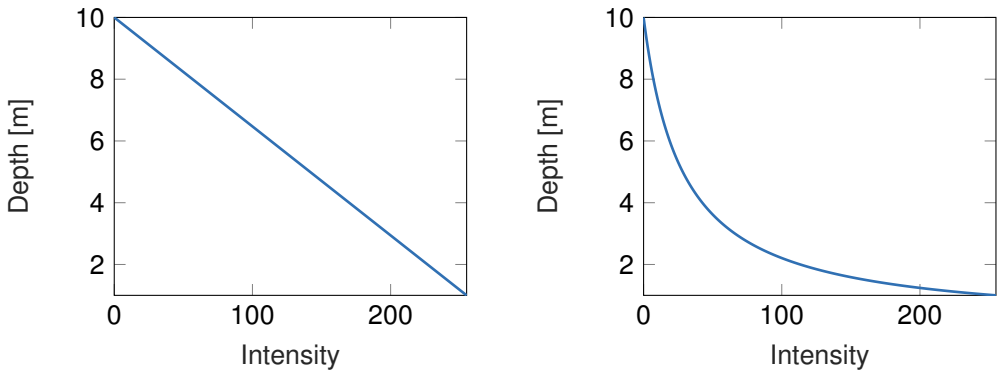


Figure 2.5.: Different mappings of the depth image intensity values I_D to the depth value Z. Plot (a) shows the linear mapping from $z_{max} = 10\,m$ to $z_{min} = 1\,m$ over the intensity range $[0 \ldots 255]$ and Plot (b) shows the inverse mapping. Inverse mapping has the advantage of a finer depth resolution in the near field.

There are two commonly used encoding techniques for mapping the real-world depth Z to the intensity values in the depth map. By convention, bright pixels are typically used to indicate objects closer to the camera and dark pixels are used for the background. The range which is being represented by the depth map needs to be specified by providing a minimum and maximum distance z_{min} and z_{max}, respectively.

The linear mapping strategy maps the intensity values linearly to the range of depth values between z_{min} and z_{max}. An intensity of the depth map $I_D = 255$ therefore gets mapped to the minimum depth z_{min} while the maximum depth z_{max} is mapped to the lowest intensity

$I_D = 0$ according to

$$Z = \left(1 - \left(\frac{I_D}{255}\right)\right)(z_{\max} - z_{\min}) + z_{\min}.$$

This mapping, however, has the disadvantage of exhibiting a constant resolution over the depth range. Depth maps are typically encoded as grayscale images with a resolution of 8 bits, which means that the depth has a resolution of $(z_{\max} - z_{\min})/255$. In the example shown in Figure 2.5 with a presumed $z_{\min} = 1$ m and $z_{\max} = 10$ m, this would lead to a depth resolution of $\Delta Z = 3.5$ cm. For a precise representation of fine details, this is unacceptably coarse. While a simple solution for this problem would be to increase the dynamic range of the depth image to 16 bits, unfortunately most of the commonly used transmission pipeline for broadcasting video is still accustomed to 8-bit images with only recent advances in the transmission of HDR content with typically used resolutions of 10 bits [23, 24].

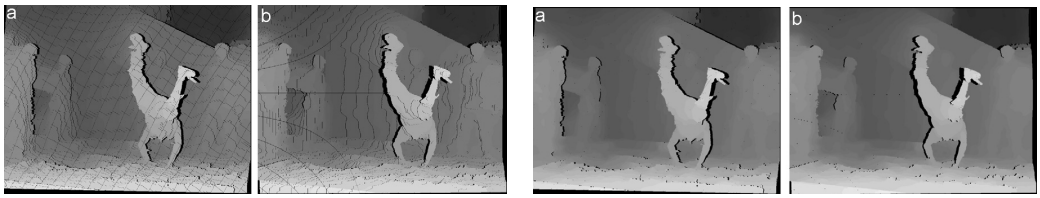
Because of this resolution problem, typically a so called *inverse mapping* is employed. There, the depth

$$Z = \frac{1}{\frac{I_D}{255}\left(\frac{1}{z_{\min}} - \frac{1}{z_{\max}}\right) + \frac{1}{z_{\max}}}$$

is encoded with a hyperbolic function which ensures a much finer depth resolution for objects which are closer to the camera.

2.4. Post-Processing of the Depth Map

Warping the depth map into the new view introduces two main artifacts as shown in Figure 2.6a. Because a forward mapping strategy is employed, we see the same pixel-sized holes which I've shown in Figure 2.3 which are introduced due to rounding errors. Also, one can see stepping artifacts in the background which are a result of the limited number of gray levels of the depth map. These artifacts of the depth map would of course introduce artifacts in the synthesized view as well. In the View Synthesis Reference Software (VSRS), Mori et al. [25] therefore proposed the following filtering techniques to deal with these kinds of errors. Instead of resorting to a backward-mapping strategy, they propose to fill in the pixel-sized holes with a median filter. To get rid of the stepping artifacts, they propose to smooth out the depth map. However, because they don't want to smooth away edge features or depth discontinuities, Mori et al. propose to apply the bilateral filter by Tomasi and



(a) (b)

Figure 2.6.: The warping process introduces two main artifacts into the depth map. Post-processing can eliminate the pixel-sized holes which stem from rounding errors and the stepping artifacts which are a result of the limited resolution of the depth map [25].

Manduchi [26] on the depth map which has the property to smooth the depth map while preserving the edges.

2.5. Rectification

For depth estimation as well as view synthesis, it may be beneficial to have rectified camera images. View synthesis then becomes much simpler as the depth can directly be mapped to a horizontal shift of the pixel. For the depth estimation, the disparity search can be constrained to horizontal lines in the rectified images.

2.6. Color Matching

When using the images of multiple cameras for virtual view synthesis, it is crucial that the color profiles of these cameras are matched. Otherwise, when filling in holes in the warped view of one camera with content of another camera, changes in color temperature, white balance, brightness or similar differences become obvious. While the best way to match the colors would be to calibrate the cameras prior to the recording, some minor differences in between the views typically can't be avoided. Therefore, several algorithmic methods have been proposed to perform this color matching post-recording. These algorithms can be classified into three categories: global, local or hybrid. A global color matching algorithm compares the histograms of both images in all color channels and tries to match these histograms, regardless of the pixel positions. Local color matching tries to find cor-

respondences in the pixels of the image, typically by using some form of block matching to ensure that only same content is compared to each other. These algorithms can therefore create transformation rules which could change for different pixel so that these can also deal with local color inconsistencies. Hybrid methods combine aspects of both global and local methods. [27, 28]

Part II.

Hole Filling

3. Related Work

3.1. Introduction

As mentioned in the introduction, there are two categories of how view synthesis algorithms deal with the disocclusion problems – the ones that avoid producing disocclusions in the synthesis step such that no filling is needed, and those that fill in the holes with known or synthesized content. The algorithms that I'm going to propose in this work are of the latter type. However, for the sake of completeness, I still want to provide a survey of the state of the art in both types of view synthesis algorithms.

3.2. Survey of View Synthesis Algorithms

Algorithms of the first category are typically smoothing the depth map so that no steep depth gradients in depth image appear. This in turn leads to a virtual view which doesn't contain any holes after warping, however, at the cost of a geometric distortion of both foreground and background objects. The algorithm proposed by Fehn et al. [10] is using a simple Gaussian filter to smooth the depth map. They are aware that "this must obviously lead to some geometric distortions" [10], however, they claim that an advantage would be that the depth map could be compressed more efficiently due to the lack of high frequency components. I will have a look at the subjective quality of two variants of this algorithm in Chapter 6. Tam et al. [29] investigated the effect of smoothing the depth map on the subjective visual quality and found that increasing levels of Gaussian blur increased the subjective visual quality while not degrading the perceived depth quality. Zhang et al. [11] extended this concept by introducing an asymmetric smoothing filter which applies stronger smoothing in the vertical direction than in the horizontal direction, claiming that this would

improve binocular depth perception which would rely mostly on horizontal cues. Chen et al. [30] and Horng et al. [31] suggested the use of directional filters in the edge direction of the depth map to better preserve the edges of objects.

As described in the previous chapter, the algorithm proposed by Mori et al. [25] is the basis of the View Synthesis Reference Software (VSRS) of the MPEG working group on Free-viewpoint Television (FTV), which I'm also using in the warping step of the virtual view generation process in the remainder of this thesis. They are proposing a filtering step to smooth out the projected depth-map to reduce depth discontinuity and a boundary matting technique to avoid ghosting artifacts where traces of background content remain at the border of foreground objects. For hole-filling, they are using the inpainting algorithm by Telea [32], which I'm going to have a closer look at in the next chapter.

The fairly recent algorithm proposed by Li et al. [33] also falls into the category of reducing the size of holes instead of filling them. Typically, only two views are used for generating a virtual view and the information of both views is merged to fill in the holes that originated in the warping of the respective other view. They've investigated whether in a multi-view recording with more than two cameras the information of what they call complementary views in addition to the two stereo views can be utilized to reduce the hole size even further and described an algorithm to decide on which information to use for merging the views. They showed that the holes can be reduced in size significantly by using more cameras, yet, in all cases they analyzed, holes remained in the virtual view which needed to be filled. This has been proposed several times in literature. Kauff et al. [34] also suggested to transmit multiple video-plus-depth streams to have sufficient material to fill in the disocclusions. Zinger et al. [35] were combining both views of a stereo system to reduce the hole size and filled in the remaining holes with a modification of Telea's [32] inpainting algorithm to work only on background pixels with the drawback "that the inpainted region becomes a low frequency patch, when the disoccluded region is very large" [35]. Müller et al. [36] use a hole-filling algorithm which fills the holes line-wise. Each horizontal line of the hole is filled with the color of the nearest background pixel which is horizontally extrapolated into the hole. They claim that "this simple constant-color extrapolation of the background pixel leads to better results, than an unconstrained linear interpolation between both values" [36]. However, this method obviously fails to reproduce texture or any kind of linear structure which is not horizontal.

The second category of view synthesis algorithms is filling the holes which remain after warping with artificially generated content, usually by means of interpolation, texture synthesis or inpainting algorithms, or with content from other frames along the temporal dimension. One of the most popular inpainting algorithms used for hole filling is the one proposed by Criminisi et al. [37]. There is a huge number of adaptations of this algorithm for disocclusion filling. Criminisi's algorithm can be categorized into the group of so-called exemplar-based techniques, i.e., the algorithm uses patches of the image itself and copies these into the hole, thus exploiting the redundancy of natural images. I will explain his algorithm in more detail in the next chapter as well, for now, suffice to say that Criminisi discovered that the order in which this filling process is executed determines the quality of the output image. He therefore introduced a confidence and a priority term with the intention to steer the filling process into the direction of isophotes, i.e., lines with constant luminance. However, using this algorithm directly for disocclusion filling in the context of view synthesis leads to very poor results [38]. Therefore, several modifications have been proposed.

Oh et al. [39] explicitly modified the boundaries of the holes to only incorporate background pixels. On the side of the hole where the foreground object is located, the pixels at the border are simply replaced by the background pixels from the opposite side before the hole is filled. Daribo and Saito [38] proposed a depth-based modification to Criminisi's priority term to prioritize background pixels over foreground pixels. Gautier et al. [40] replaced the color gradient in the priority term with a structure tensor based on the color of the texture and the structure of the depth map. Ndjiki-Nya et al. [41] steered the filling order from the background towards the foreground by first initializing the holes in the depth map with an estimate for the depth at the unknown regions. It is important to note that some of these methods actually require the depth-map of the virtual view, i.e, the correct depth at the region of the disocclusion, which is not available in most view synthesis use-cases. Ahn and Kim [42] and Buyssens et al. [43] therefore proposed algorithms which perform depth-map filling and disocclusion filling consecutively. Zhu and Li [44] created an analytical model to determine the size of the holes and proposed an additional step during the warping process to preserve occluded depth information which can then be used to guide Criminisi's inpainting algorithm.

Criminisi's inpainting technique is a greedy algorithm, i.e., once a patch has been copied

into the hole, it won't be changed regardless of the patches that follow in its neighbourhood. Komodakis and Tziritas [45] recognized this as a potential drawback and therefore introduced an inpainting algorithm based on the solution of a Markov random field. They demonstrate that this technique has the potential to significantly outperform the method of Criminisi in terms of visual quality of the inpainting result. In this thesis, I therefore propose an adaptation of the algorithm of Komodakis and Tziritas for disocclusion filling for view synthesis.

The last class of algorithms exploit temporal or spatio-temporal correlation between consecutive frames of a video sequence to fill in holes. While it may seem obvious to use the content of neighboring frames which may not be occluded for hole filling, the challenge here lies in the circumstance that the content of a scene may change rapidly, either in the background or in the foreground or both, which renders the content unusable very quickly.

Schmeing and Jiang [46] proposed an algorithm which is trying to fill in the holes using *faithful* pixels, in contrast to the *plausible* pixels that are used in inpainting. Faithful pixels are ones which are taken from temporally adjacent frames. The problem with filling in content from different frames is that the content is likely to change over time which makes it difficult to identify valid information. While Cheng et al. [47] try to achieve this goal by simply extending the source region of the inpainting algorithm of Criminisi [37] to not only use patches from the current image but also temporally adjacent frames, Schmeing and Jiang extend this idea by using SLIC superpixels [48] instead of patches. While their approach to extend the search region by another dimension is valid to improve the image quality, it also increases computational cost significantly.

Hsu et al. [49] and Kim et al. [50] proposed a global energy minimization approach with a Markov Random Field (MRF) to ensure spatial and temporal consistency in the disoccluded area.

Recently, machine learning methods with deep neural networks are gaining increasing attention for view synthesis and hole filling for depth-image-based rendering. Zhou et al. [51] presented a technique to increase the baseline of a small baseline stereo pair typically found in recent smartphones by training a deep neural net to infer the global scene representation described by fronto-parallel planes at fixed depths. Flynn et al. presented "Deep Stereo" [52] which trains a network to infer the depth and the color of a new view sep-

arately, and, more recently "DeepView" [53], a method for estimating multi-plane images from sparse views using learned gradient descent. Ali Eslami et al. [54] presented the Generative Query Network, a framework which can construct an internal representation of a scene and can predict views which were not seen during training.

Many of these methods either utilize an enormous amount of training data with thousands of images from the same scene to train the network or introduce significant artifacts in the hole regions when the baseline becomes larger [51]. Also, many of these methods need to train the networks on each scene individually [52, 55], resulting in an enormous computational complexity for each frame of a video. Others, such as the work by Zhou, can predict a the virtual view via a learned feed-forward network, thereby reducing this effort, however, they become ineffective for dealing with disocclusions, as "the number of network connections required to effectively model this visibility can become prohibitively large" [53]. Occluded content which is not visible in any of the views typically can't be constructed by these algorithms because it can't be trained to the network. DeepStereo for example is "unable to render surfaces that appear in none of the inputs" [52]. While the work of Ali Eslami et al. [54] is able to predict unseen views from a small set of input views and correctly handle the occlusions that appear in the scene, they demonstrate this only for very low resolution (64×64 pixels), extremely simple synthetic rendered environments on a network that has been trained on 2 million scenes. These toy examples already have a prohibitively high computational cost, which makes rendering natural scenes with medium to high resolution intractable using their technique. Despite all of these drawbacks, I'm confident that we will see significant advancements out of that research direction in the near future.

4. Algorithms for Hole Filling

4.1. Introduction

The terms hole filling and inpainting are used somewhat synonymously in literature, however, I chose to use hole filling in the title of this thesis because of its broader scope. Hole filling in the case of DIBR is more than just inpainting; it consists of the reconstruction of the view geometry, the reconstruction of the depth information and, as the goal, the synthesis of the image information that is missing. From the very beginning of digital image processing, researchers have also addressed the hole-filling problem [56], with early applications e.g. in the reconstruction of satellite images [57] or the restoration of telefaxes which were damaged due to transmission errors [58]. Early research has focused on texture synthesis, where statistical models were investigated to synthesize *texture*, usually describing some kind of regular, often repeating pattern which can be modeled sufficiently well using the wealth of experience gained with time series modelling, applied in a spatial instead of a temporal dimension. However, texture synthesis in this form is limited to very regular images. *Natural* images of sufficient complexity can typically no longer be described by a single mathematical model. In this chapter, I will therefore review the most relevant techniques to hole filling and describe the algorithms I have developed for this task.

4.2. Interpolation, Diffusion and Partial Differential Equations

One of the simplest and maybe most intuitive approaches to hole filling is interpolation. There, the pixels around the hole are used as boundary values for interpolating functions, which then calculate intermediate values for the missing pixels inside the hole. Commonly used interpolation functions in 2-D are nearest neighbor, bilinear and functions of higher

order, e.g. bicubic interpolation. When using interpolation for hole filling, one draws the implicit assumption that the missing content is appropriately described by its surrounding environment. This assumption will typically only hold true for relatively small holes or very homogeneous image contents. Also, the maximum spatial frequency of the synthesized content will directly depend on the interpolating function and the size of the hole. By definition, interpolation kernels are low-pass filters. When only the boundary pixels are used for the interpolation, the size of the hole defines the sampling rate of the input of the interpolating function. The maximum spatial frequency of the output signal will therefore depend on the size of the hole, with a larger hole resulting in lower spatial frequency. In practice, this leads to interpolation results with very little detail and often very smooth regions. Depending on the size of the hole and the spatial frequency of the content around the hole, these changes in spatial resolution may become very apparent [59, 60].

Researchers have recognized these shortcomings of basic interpolation kernels. Depending on the implementation of the interpolation algorithm, typically only one single layer of pixels around the hole will be used as boundary values. However, it is beneficial to extend this source region to a wider area around the hole, possible weighting it with the distance from the hole. Kaup and Aach [61], for example, proposed to extrapolate the background content into an uncovered region by evaluating the prevailing spectral components in a discrete Hartley transform and extrapolating those into the unknown region using a technique called successive extrapolation, which leads to a much more natural continuation of the background texture.

It was also discovered that it may be important to continue lines arriving at the boundary of the hole throughout the hole and connect them where possible to other lines. One of the most fundamental techniques exploiting that insight was proposed by Bertalmio et al. [62]. They formulated an analogy between the inpainting process and fluid dynamics, with the image being regarded as a stream function and the isophotes arriving at the border of the hole as fluid streams with a certain direction of the flow velocity. They were then able to derive a set of partial differential equations based on the Navier-Stokes equations, which when being solved generate the inpainted images. They showed that this technique can be applied to images and videos, and it is also being used as the algorithm for inpainting in the VSRS of the Moving Picture Experts Group (MPEG) FTV project. However, it suffers from the drawbacks which I've already insinuated, namely creating very smooth in-

painting results which when applied to large holes tend to lose the structure of the original image. I've included the results of the VSRS reference implementation for comparison in the evaluation of my algorithms in Chapter III.

4.3. Exemplar-Based Techniques

Exemplar-based techniques are a class of algorithms which use patches directly from a source image to fill in the missing regions. The term has been introduced by the paper of Criminisi et al. [63]. They were one of first to show an algorithm which manages to fill large holes in *natural* images, i.e. images which don't consist purely of one single texture, in a "visually plausible way" [63]. As this algorithm lies the foundation of many algorithms which were developed on its basis, and also because I'm going to adapt their notation, I'm going a little bit more into detail on how it works.

The basic idea of sampling directly from a source image was introduced by Efros and Leung [64]. They recognized that the most promising approaches at that time treated the texture synthesis problem as one of sampling from a probability distribution, e.g. modelling the texture as a Markov Random Field and using Gibbs sampling for synthesis [65]. However, instead of constructing a probability distribution drawing assumptions about the statistical properties, they introduced the idea of querying samples directly from a source image, a technique which they call *non-parametric sampling*. This idea had been introduced as early as 1948 by Claude Shannon for natural text synthesis [66] by modelling language as a Markov chain where consecutive words are being sampled from a large distribution such as a book.

Extending this idea to 2-D, the image I is synthesized from a texture sample $I_{smp} \subset I_{real}$ which is a section of the real infinite texture image I_{real}. From this texture sample, Efros and Leung construct a conditional probability distribution $P(p|\omega(p))$ to sample from with p being a single pixel which is to be reconstructed and $\omega(p)$ being a patch around the central pixel p with dimension $w \times w$. Modelling the texture as a Markov Random Field (MRF), they assume that the brightness value (or color for RGB images by simple extension of their algorithm to a color space) given its surrounding neighborhood $\omega(p)$ is independent of the rest of the image I, i.e. p is independent of $I \setminus \omega(p)$ given $\omega(p)$.

Using a perceptual distance metric $d_{perc}(\omega_1, \omega_2)$ between two patches ω_1 and ω_2, they then define a set $\Omega(p) = \{\omega' \subset I_{real} : d_{perc}(\omega', \omega(p)) = 0\}$ which contains all occurrences of the patch $\omega(p)$ in I_{real}, which can be used to produce a histogram of all pixel values p as an approximation for the conditional PDF $P(p|\omega(p))$. This distance metric could be a normalized sum of squared differences (SSD), however they chose to include a gaussian kernel G because they want to give a higher weight to pixel values closer to the center pixel, which leads to $d_{perc} = d_{SSD} * G$.

Because it is likely that there are no direct occurrences of $\omega(p)$ in the finite sample image I_{smp}, Efros and Leung are weakening the condition in $\Omega(p)$ to get an approximative set $\Omega'(p)$ based on a variation of the k-Nearest-Neighbour algorithm. They search for the closest match which minimizes the perceptual distance $\omega'_{best} = \arg\min_{\omega} d_{perc}(\omega(p), \omega) \subset I_{smp}$ and then include all patches ω' into the set $\Omega'(p) \approx \Omega(p)$ which have a perceptual distance $d_{perc}(\omega'_{best}, \omega') < \epsilon$ smaller than a threshold ϵ.

As the holes to be filled are typically larger than one single pixel, not the entire neighborhood of the center pixel is known. Therefore, ideally a joint probability of all pixels would be constructed, which they claim to be "intractable for images of realistic size" [64]. While I'm going to present a solution for that problem later on, Efros and Leung are proposing another heuristic in which they are filling the hole in concentric circles from the edge of the hole. There, the algorithm needs to be modified to ignore the unknown pixel values and only use the known values at the edge to construct the conditional PDF which can easily be done by ignoring the unknown pixel values and normalizing the distance metric for the number of known pixels.

A number of variants of that algorithm have been proposed, all relying on that same principle of querying single pixel values by exhaustively searching fitting patches in a query image with the goal of synthesizing a more or less regular texture [67, 68].

4.3.1. Structure Propagation

A rather large step into the direction of natural image synthesis was then undertaken by Criminisi et al. [63]. They noticed that the quality of the inpainting result depends heavily on the order in which the filling process is executed. Simple concentric filling of the hole

region may lead to artifacts which originate from the structure of the hole rather than the structure of the original image.

Their contribution was an extension of the algorithm just described with an idea of the seminal paper *Image inpainting* by Bertalmio et al. [69]. They were one of the first to introduce the idea of isophote continuation to fill holes. The basic principle is to continue isophotes (i.e. lines of equal brightness) in the direction of their arrival at the border $\delta\Omega$ of a hole region Ω, much like traditional restaurateurs are working when manually repairing damaged pictures. The hole is then filled with the same color as the contour line which is arriving at the hole. While this algorithm works quite well even on natural images, it has the drawback of smoothly continuing the isophotes into the hole, thus losing the original structure of textured regions, a disadvantage that the authors mention themselves in their paper.

Criminisi et al. combine these two ideas. They use the exemplar-based approach for its property of accurately capturing the texture of the image to be synthesized, while steering the filling process into the direction of isophotes, thus taking into account the dependency of the result on the filling order.

They separate their image to be inpainted $I = \Phi \cup \Omega$ into a source region Φ and the hole Ω, with no constraints on the shape and topology of the hole, and define patches Ψ with a typical size of 9×9 pixels, depending of the largest texture element in the pictures. Each pixel on the fill front gets assigned a confidence value

$$C(\mathbf{p}) = \frac{\sum_{\mathbf{q} \in \Psi_\mathbf{p} \cap \Omega} C(\mathbf{q})}{|\Psi_\mathbf{p}|},$$

with $\Psi_\mathbf{p}$ being a patch around pixel $\mathbf{p}$ and $|\Psi_\mathbf{p}|$ the area of this patch. The confidence describes how much information in the neighborhood of this pixel is already known. This confidence is then used to calculate a priority

$$P(\mathbf{p}) = C(\mathbf{p}) \cdot D(\mathbf{p}) \tag{4.1}$$

by multiplying the confidence term with a so called data term

$$D(\mathbf{p}) = \frac{\left|\nabla I_\mathbf{p}^\perp \cdot \mathbf{n}_\mathbf{p}\right|}{\alpha},$$

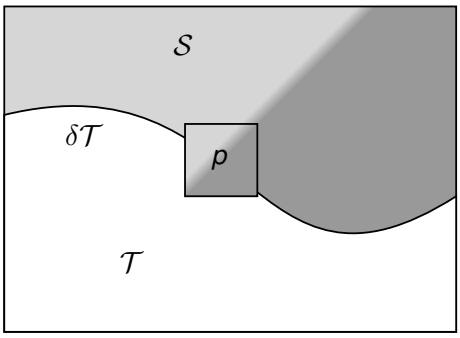


Figure 4.1.: This illustration shows the inpainting step in isophote-driven exemplar-based algorithms such as the one of Criminisi. The hole is the target area $\mathcal{T}$ and the border of the hole is denoted by $\delta\mathcal{T}$. A suitable patch is copied from the source region $\mathcal{S}$ to the position *p*.

with $\nabla I_{\mathbf{p}}^{\perp}$ being the direction of the isophote in the point $\mathbf{p}$, $\mathbf{n_p}$ a unit vector orthogonal to the border of the hole $\delta\Omega$ at point $\mathbf{p}$ and α a normalization factor.

Once the priorities of all pixels around the hole have been computed, the patch with the highest priority $\Psi_{\hat{\mathbf{p}}}$ will be filled in. Therefore, in the source region the patch which is most similar to the known pixels of $\Psi_{\hat{\mathbf{p}}}$

$$\Psi_{\hat{\mathbf{q}}} = \arg\min_{\Psi_{\mathbf{q}} \in \Omega} d(\Psi_{\hat{\mathbf{p}}}, \Psi_{\mathbf{q}}) \tag{4.2}$$

according to a suitable distance metric $d(\Psi_{\hat{\mathbf{p}}}, \Psi_{\mathbf{q}})$ such as SSD in the Lab color space, is being searched and the unknown pixels which are part of the hole are then updated with the best matching source patch. Unfortunately, this exhaustive search of patches in the source region is computationally extremely expensive, making the whole algorithm relatively slow. I will show improvements to this strategy in Chapter 4.4.3.

Finally, when the patch has been filled, the confidence at that point will be updated according to

$$C(\mathbf{q}) = C(\hat{\mathbf{p}}) \quad \forall \mathbf{q} \in \Psi_{\hat{\mathbf{p}}} \cap \Omega.$$

Figure 4.2.: Result of Criminisi's algorithm when applied to the hole-filling problem

4.3.2. Criminisi's Algorithm for Hole Filling

Figure 4.2 shows the result of Criminisi's algorithm when directly applied to the hole-filling problem. The result is rather unsatisfactory as the inpainting result exhibits several obvious problems. The disoclussion is partially filled with foreground content and the structure propagation seems to be working poorly in this scene as revealed by the discontinuation of the bars in the background.

Several researchers have identified these problems and proposed solutions for these drawbacks. Oh et al. [39] explicitly modified the hole at the border close to foreground objects and replace the missing content there with background content from the opposite side of the hole. This ensures that the inpainting algorithm is only operating on background content. Cheng et al. [70] proposed to constrain the search region to the background, therefore avoid having to manipulate the hole geometry first. Advancing these two approaches, Daribo et al. [38] proposed a modification of Criminisi's algorithm as follows. They extend the priority computation term of Equation 4.1 with a so called level regularity term

$$P(\mathbf{p}) = C(\mathbf{p}) \cdot D(\mathbf{p}) \cdot L(\mathbf{p})$$

which considers the variance of the corresponding depth-image patch according to

$$L(\mathbf{p}) = \frac{|Z_\mathbf{p}|}{|Z_\mathbf{p}| + \sum_{\mathbf{q}\in\Psi_\mathbf{p}\cap\Omega} \left(Z_\mathbf{p} - \overline{Z_\mathbf{p}}\right)^2}$$

with $\overline{Z_\mathbf{p}}$ being the mean depth value of that patch. This gives priority to background patches simply because they don't exhibt such strong depth discontinuities as the patches on the border of foreground objects. Daribo et al. also extend the search strategy for suitable patches of Equation 4.2

$$\Psi_{\hat{\mathbf{q}}} = \underset{\Psi_\mathbf{q}\in\Omega}{\arg\min} \left\{ d(\Psi_{\hat{\mathbf{p}}}, \Psi_\mathbf{q}) + \beta \cdot d(Z_{\hat{\mathbf{p}}}, Z_\mathbf{q}) \right\}$$

by a term which takes the corresponding depth of that patch into account, weighted by a factor β whose selection they unfortunately don't mention in their paper. Also, because their algorithm relies on the same exhaustive search strategy, this algorithm shares the disadvantage of being computationally extremely expensive.

Gautier et al. [40] made some further modifications of Criminisi's and Daribo's algorithm. They replace the data term $D(\mathbf{p})$ by a "more robust structure tensor" [40] based on the Di Zenzo matrix [71] of a local spatial gradient in a 3×3 window $\nabla I(\mathbf{p})_I$

$$J(\mathbf{p}) = \sum_{I=R,G,B} \nabla I(\mathbf{p})_I \nabla I(\mathbf{p})_I^T.$$

By calculating the eigenvalues $\lambda_{1,2}$ of $J(\mathbf{p})$, they derive a measure for variation in the structure and use that to get a new data term

$$D(\mathbf{p}) = \alpha + (1-\alpha) \exp\left(\frac{-C}{(\lambda_1 - \lambda_2)^2}\right).$$

Furthermore, they don't just use one single exemplar patch to fill in the target but a combination of the K best patches from a K-nearest-neighbor search, where they chose $K = 5$. Josselin Gautier was kind enough to provide me with his implementation of the algorithm so that I could use it to compare the results of his algorithm with mine. The results of these comparisons can be found in Section III.

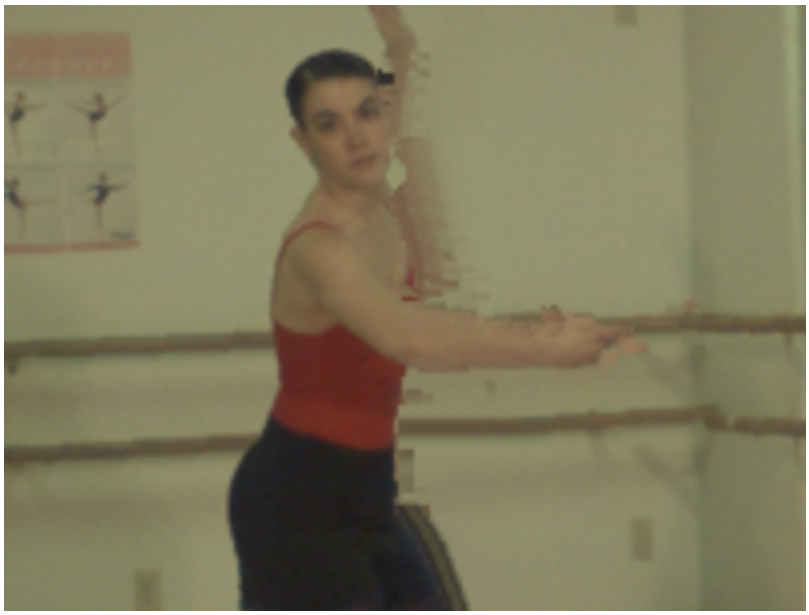


Figure 4.3.: Result of Gautier's algorithm when applied to the hole-filling problem

4.3.3. Propagation with Graphical Models

Parts of the following section have already been published in [72].

Introduction to Komodakis and Tziritas' Algorithm

Criminisi's algorithm uses the isophote continuation as an heuristic to guide the filling process. Komodakis and Tziritas [45] identified this as a potential drawback and instead of the greedy synthesis method (i.e. one patch at a time), they propose to pose the image completion problem as a discrete global optimization problem. That way, they want to make sure that the image doesn't contain any local inconsistencies. Also, instead of building the image out of single patches, their algorithm maintains several candidate patches for each location which are then iteratively selected to form the complete image. As I've developed a hole-filling algorithm extending on their idea, I'm going to explain it more in detail. I will

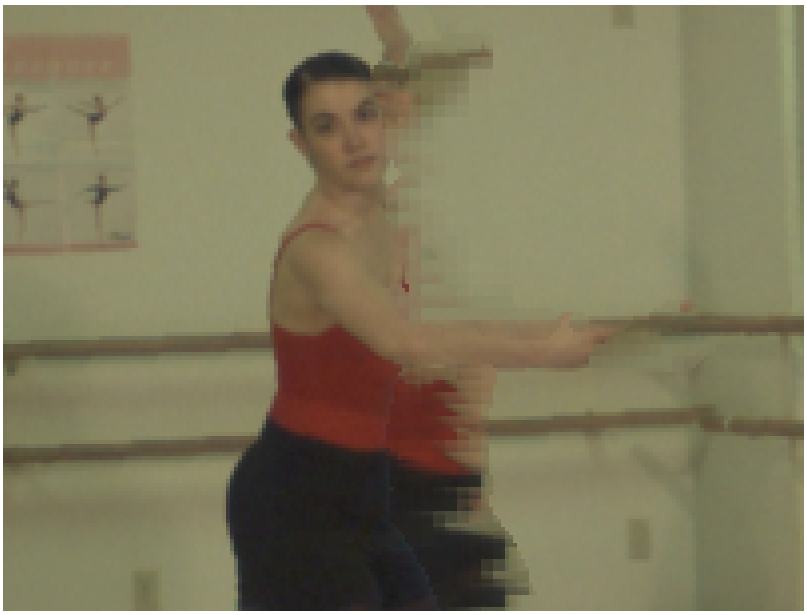


Figure 4.4.: Komodakis and Tziritas' algorithm cannot be used directly to fill disocclusions. The most obvious problem is bleeding of foreground objects into the background.

explain my modifications in the next section of this thesis. For clarity, I try to adapt the notation of [45] as closely as possible.

Prerequisites

Again, I'm separating the image I into a source region $\mathcal{S}$ and a target region $\mathcal{T}$, i.e., the holes to be filled. In the view-synthesis problem, the location of the holes is determined by the scene geometry, and during the process of mapping the texture of the original view to the virtual view, I can simultaneously generate a mask which specifies the location of the holes. I am using the MPEG View Synthesis Reference Software (VSRS) [73] to conduct this mapping and to generate the virtual view from the texture and the depth map. The results of this warping process can be seen the illustration in the introduction of this thesis in Figure 1.4. I've modified the source code of VSRS to output the intermediate products of the view synthesis process, e.g. the map of holes and the unfilled warped view which I will need for the hole filling.

The image is then partitioned into small, overlapping patches of size $w \times h$ with a spacing

of gap_x and gap_y, respectively. As the patches need to overlap each other, $\text{gap}_x < w$ and $\text{gap}_y < h$. The goal of the inpainting algorithm is then to find suitable patches from S which can be filled into the holes T. To this end, Komodakis and Tziritas proposed a Markov network which consists of nodes $\mathcal{V} = \{1, 2, \dots, N\}$ at the positions of the patches inside and at the border of the hole, each of which is associated with a random variable X_j out of the set of random variables $\mathbf{X} = \{X_1, X_2, \dots, X_n\}$ and a set of edges $\mathcal{E}$ which connect these nodes to their four neighbors. Figure 4.6a shows an exemplary distribution of the nodes over the holes in the view synthesis setting.

The resulting graph defines a Markov Random Field (MRF). An MRF is an undirected graphical model $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where the random variables exhibit a Markov property, i.e. their probablity depends only on their direct neighbors, not on the rest of the random variables of the network

$$Pr(X_i | \{X_j\}_{j \in \mathcal{V} \setminus i}) = Pr(X_i | \{X_j\}_{j \in \mathcal{N}_i})$$

where $\mathcal{N}_i$ is the neighborhood of node i, i.e., $j \in \mathcal{N}_i \iff (i, j) \in \mathcal{E}$.

Each of the nodes has a set of labels $\mathcal{L} = \{l_1, l_2, \dots, l_M\}$ associated with it which comprises candidate patches from S to be inserted at the position of the node. The assignment of a specific label to a certain node thus would result in a specific patch being copied to that location. I denote the probability for a node p_i taking on a label l_i as $Pr(X_i = x_i)$. The vector $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$ thus describes a labeling of the random field with the joint probability $Pr(\mathbf{X} = \mathbf{x})$ or $Pr(\mathbf{x})$.

According to the Hammersley-Clifford theorem [74], one can define the joint probability as a factorization of the set of cliques (i.e. fully connected subgraphs) $\mathcal{C}$

$$Pr(\mathbf{x}) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \exp(-\phi_C(\mathbf{x})), \tag{4.3}$$

where the functions $\psi_C(\mathbf{x})$ are potential functions and

$$Z = \sum_{\mathbf{x}} \prod_{C \in \mathcal{C}} \phi_C(\mathbf{x})$$

is a normalization constant, often called the partition function which ensures that the dis-

tribution sums to 1. Equation 4.3 can be rewritten as a Gibbs distribution

$$Pr(\mathbf{x}) = \frac{1}{Z} \exp\left(-\sum_{C \in \mathcal{C}} (-\psi_C(\mathbf{x}))\right), \tag{4.4}$$

where $\psi_C(\mathbf{x}) = -\log(\phi_C(\mathbf{x}))$. Given that I want to find the labels for each patch, I'm looking for $\hat{\mathbf{x}}$, the maximum a-posteriori estimate

$$\begin{aligned}\hat{\mathbf{x}} &= \arg\max_{\mathbf{x}} Pr(\mathbf{x}) \\ &= \arg\max_{\mathbf{x}} \frac{1}{Z} \prod_{C \in \mathcal{C}} \exp(-\phi_C(\mathbf{x}))\end{aligned}$$

or when applying Equation 4.4 one can formulate this as an energy minimization problem

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} E(\mathbf{x})$$

with

$$\begin{aligned}E(\mathbf{x}) &= -\log\left(Pr(\mathbf{x})\right) \\ &= -\log\left(\frac{1}{Z} \prod_{C \in \mathcal{C}} \exp(-\phi_C(\mathbf{x}))\right) \\ &= \sum_{C \in \mathcal{C}} \phi_C(\mathbf{x}) + const.\end{aligned}$$

In a first order random field, this energy

$$E(\mathbf{x}) = \underbrace{\sum_{i \in \mathcal{C}} \phi_i(x_i)}_{\text{unary pot.}} + \underbrace{\sum_{(i,j) \in \mathcal{C}} \phi_{ij}(x_i, x_j)}_{\text{pairwise pot.}}$$

can be written as a sum of unary and pairwise potentials. In the following, I'm going to adapt the notation of Komodakis and Tsiritas and denote the unary potential as the node potential V_p and the pairwise potential as V_{pq}. The node potential

$$V_p(x_p) = \sum_{dp \in \left[-\frac{w}{2} \frac{w}{2}\right] \times \left[-\frac{h}{2} \frac{h}{2}\right]} \mathcal{M}(p + dp)(I(p + dp) - I(x_p + dp))^2$$

defines the specific cost which the association of any of the labels to a node p incurs, i.e., how well the patch x_p matches any available content from S. $\mathcal{M}$ denotes a mask which is zero inside $\mathcal{T}$, 1 else. Similarly, a pairwise potential $V_{p,q}(x_p, x_q)$ is constructed which states how well the patch matches the other patches in its 4-connected neighborhood where they are overlapping. This pairwise potential has to consider the respective orientations of the pair of patches because the overlapping area is different, e.g., for a patch above the current position than below.

Solving the optimization problem defined by an MRF can be shown to be NP-hard [75], however, especially in the field of image processing, algorithms like belief propagation [76] or graph cuts [77] have demonstrated to provide good approximate solutions to these problems [78, 79].

Priority Belief Propagation

The goal of the optimization problem is to minimize the total energy of the MRF

$$\mathcal{F}(\hat{x}) = \sum_{p \in \nu} V_p(\hat{x}_p) + \sum_{(p,q) \in \varepsilon} V_{pq}(\hat{x}_p, \hat{x}_q), \tag{4.5}$$

by assigning labels $\hat{x}_p \in \mathcal{L}$ to each node p for which Komodakis and Tziritas proposed a *priority-belief propagation* algorithm. The classical belief propagation algorithm was introduced by Pearl [76] and soon became a standard algorithm for the inference on MRFs.

In belief propagation, messages are exchanged between the connected nodes about the confidence in the association of a patch to a neighboring node, which then in turn defines the belief $b_p(x_p)$ each node has in its own set of labels. A message $\{m_{pq}(x_q)\}_{x_q \in \mathcal{L}}$ denotes a message sent from node p to node q about the confidence node p has about assigning label x_q to node q. Thus, in every iteration of the belief propagation algorithm there are $|\mathcal{L}|$ message sent from one node to each of its neighboring nodes. This message consists of three components

$$m_{pq}(x_q) = \min_{x_p \in \mathcal{L}} \{ V_{pq}(x_p, x_q) + V_p(x_p) + \sum_{r : r \neq q, (r,p) \in \mathcal{E}} m_{rp}(x_p) \}. \tag{4.6}$$

For one message about one label to be sent to node q, node p must first check each of

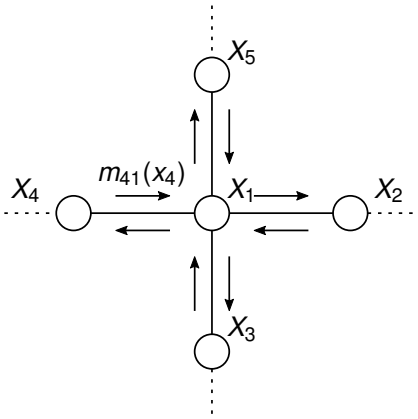


Figure 4.5.: Illustration of the basic message passing scheme in belief propagation. A central node can exchange messages with its four-connected neighborhood.

its own labels $x_p \in \mathcal{L}$ about how well labels x_q and x_p match (i.e., calculate the pairwise potential), check how well its label x_p matches at its own position and gather the messages from all surrounding nodes of p about their belief in the association of label x_p to node p. It becomes clear that for the exchange of information between nodes, all nodes have to cooperate. This message exchange formulated in Equation 4.6 is therefore conducted iteratively until all nodes agree about the labels which should be assigned to them. Once the nodes have converged to a specific labeling, a set of beliefs

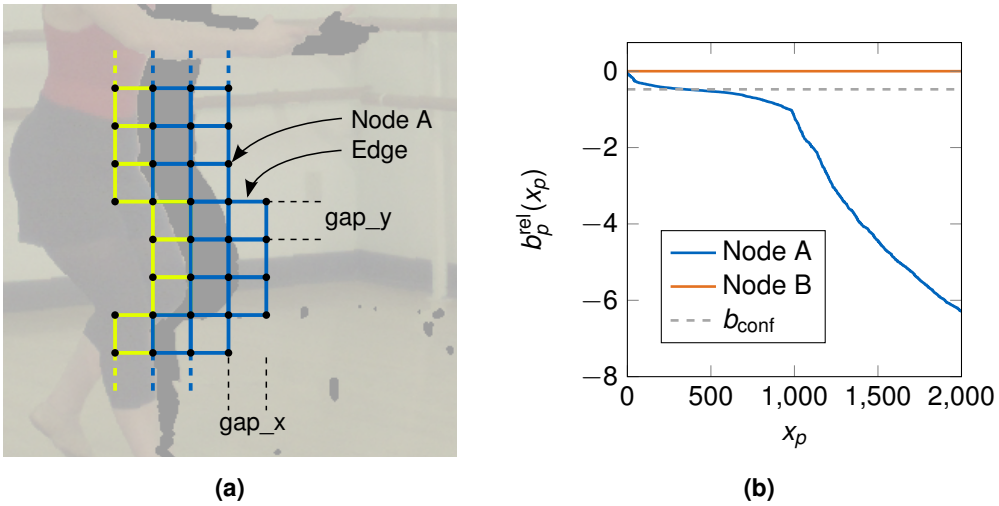
$$b_p(x_p) = V_p(x_p) - \sum_{r:(r,p)\in\mathcal{E}} m_{rp}(x_p)$$

is computed for each node. This belief combines the confidence a node has in the assignment of a label to itself and the information it got from all neighboring nodes. To find the label with the maximum likelihood, the label with the highest belief

$$\hat{x}_p = \arg\max_{x_p\in\mathcal{L}} b_p(x_p)$$

is found for that node.

In the setting of image completion, the number of labels represents the number of patches which can be created out of that image. Depending on the image size and the size of the patches it may easily consist of hundreds of thousands of patches. As the message passing step needs to consider all labels for one message to be sent for one label, its



(a) **(b)**

Figure 4.6.: Figure (a) shows a schematic visualization of the distribution of the nodes over the disocclusion. Nodes marked with yellow edges lie over foreground content and will be assigned a node potential $V_p = 0$. Figure (b) showcases the relative beliefs of a node at the border of the hole (Node A) and of any interior node (Node B) before the message passing step.

complexity is $\mathcal{O}(|\mathcal{L}|^2)$. The computational cost of the standard BP-algorithm therefore would be prohibitive in this setting. Komodakis and Tziritas thus added a method called *dynamic label pruning* based on priority. If a node has only a small set of labels in which it has a belief higher than a given confidence threshold, i.e.,

$$b_p^{\text{rel}}(x_p) \geq b_{\text{conf}}$$

with

$$b_p^{\text{rel}}(x_p) = b_p(x_p) - b_p^{\max}(x_p),$$

it will be assigned a high priority, that means it is quite confident about the assignment of its patch. On the other hand, if a node has similar beliefs in all of its labels, it may be considered indetermined and will be given a low priority. Nodes with high priority will be the ones to first get rid of all labels in which they have a low belief and then send efficient messages. Figure 4.6b shows the distribution of the relative beliefs of a node with high priority which is usually located at the border of the hole. An interior node has a low priority as its node potential is zero and therefore has the same belief in all of its labels.

4.3.4. Extension for View Synthesis - VS-BP

Komodakis and Tziritas's algorithm is not directly applicable to the disocclusion problem in view synthesis as a naïve application leads to very poor results, as shown in Figure 4.4. The most obvious problem is that there occurs bleeding of foreground objects into the background, which should be avoided. I therefore present extensions which will deal with this problem. Disocclusions occur at steep depth gradients, where there is a jump between a foreground object to the background of a scene. When I move the virtual camera to the right, disocclusions will appear on the right side of foreground objects. I therefore adapt the idea of Gautier [40] and others to steer the filling process into the opposite direction of the camera displacement. In this setting, I achieve this by modifying the node potential of all nodes that are on the side of the disocclusion opposite to the camera displacement, e.g., on the left side. These nodes, in Figure 4.6a marked as yellow, are given a node potential $V_p = 0$. The algorithm thereby treats them just like interior nodes and they will get the lowest priority. As the MRF now doesn't have any support on the left side of the hole, the inpainting task has become somewhat similar to the texture synthesis task described in [45]. It is therefore necessary to introduce another term

$$V^0_{pq}(x_p, x_q) = \begin{cases} w_0, & \text{if } x_p - x_q \neq p - q \\ 0, & \text{otherwise} \end{cases}$$

to the cost function which enforces the coherence of the image by penalizing the filling of non-adjacent patches.

Furthermore, I modify the node potential

$$V_p(x_p) = V_{I,p}(x_p) + \lambda_D V_{D,p}(x_p) \tag{4.7}$$

and the pairwise potential to not only accommodate for visual similarity between neighbouring nodes but also for similarity in depth. To this end, I add another term to both potentials which calculates the SSD

$$V_{D,p}(x_p) = \sum_{dp \in [-\frac{w}{2}\,\frac{w}{2}] \times [-\frac{h}{2}\,\frac{h}{2}]} \mathcal{M}(p + dp)(\mathcal{D}(p + dp) - \mathcal{D}(x_p + dp))^2 \tag{4.8}$$

Figure 4.7.: Result of my VS-BP algorithm when applied to the hole-filling problem

in the depth map $\mathcal{D}$, weighted by a factor λ_D. Thereby, I make sure that candidate patches are selected from similar depth ranges as the surrounding of the nodes which ensures consistency of the image and also improves the efficiency of the algorithm because it dramatically reduces the number of contemplable labels for each node.

Results

In the remainder of this thesis I will identify the algorithm just described as *VS-BP*, short for View Synthesis with Belief Propagation. Figure 4.7 shows a crop of the result of my *VS-BP* algorithm when applied to the first frame of the *Ballet* sequence. One can see that this algorithm no longer fills in foreground texture into the disocclusion and that the structure of the elements in the background has been propagated much better, as can be seen in the continuation of the bars in the background. Figure A.1 shows a visual comparison of the results of VS-BP with two other state-of-the-art algorithms. In Chapter III, I will compare the performance of this algorithm against the state of the art in disocclusion filling algorithms and show that it outperforms even recently published algorithms in terms of visual quality. However, even with the performance optimizations described in this section, it is still one

of the slowest algorithms in that comparison (see Section 5.5.3). With the algorithm I will describe in Section 4.4, I want to tackle this problem.

Temporal Consistency

The VS-BP algorithm can very simply be extended to the temporal dimension by adding a third term to the node potential of Equation 4.7

$$V_p(x_p) = V_{I,p}(x_p) + \lambda_D V_{D,p}(x_p) + \lambda_T V_{T,p}(x_p)$$

with

$$V_{T,p}(x_p) = \sum_{dp \in [-\frac{w}{2}\frac{w}{2}] \times [-\frac{h}{2}\frac{h}{2}]} \mathcal{M}_t(p+dp)(I_t(x_p+dp) - I_{t-1}(\hat{p}+dp))^2$$

to improve the temporal consistency of the output with I_t and I_{t-1} being consecutive frames and where points p and $\hat{p}$ are the corresponding points in both frames which have been found when estimating the motion vectors between these two frames. This extension of my algorithm has been proposed by Kim et al. [80]. They show that this extension improves the temporal consistency in terms of flicker between frames (see Chapter 5.5.2 for more information of the evaluation of temporal consistency and flicker). They are also showing how this method can be used to propagate the labeling to the next frame and thereby reduce the computation time by initializing the labels with the previous result. However, as mentioned in the introduction, propagating information of temporally adjacent frames bears the risk to introduce outdated content into the hole because the background image could be changing. Kim et al. [80] avoid this problem by evaluating their algorithm only on scenes with static background.

4.4. Approximate Nearest Neighbour Methods

One of the major drawbacks of exemplar-based techniques is their exhaustive search effort for matching patches. Finding the best matching patch of a set consisting of all patches of a source image, much less of a source video, quickly becomes infeasible. The number of patches in an image which are overlapping by one pixel is roughly proportional to the

number of pixels in this image, not considering the border of the image. As we will see in the runtime evaluation in Chapter III, this enormous search effort is the most important factor for the high computational complexity of these algorithms.

Therefore, for an efficient patch matching algorithm, there is a need for extremely efficient search strategies. As I am searching for matching patches via a similarity measurement, this problem reduces to a nearest neighbor search. The patch can be represented as a vector with matching patches represented as points in the space of all patches. The closest matching patch is the point which minimizes a given dissimilarity function, which is called the nearest neighbor. Donald Knuth describes this problem of searching for a nearest neighbor with the example of a post-office tree, which was first suggested by Bruce McNutt [81]. There, the task is to find the nearest post office given a destination address.

In this section of my thesis, I'm going to give a very brief overview about existing nearest neighbor algorithms which have been or could theoretically be used for DIBR and then present a novel DIBR algorithm based on a recent and very efficient nearest neighbor algorithm.

4.4.1. Search Trees

Linear search, i.e. the brute-force method, has a search complexity of $\mathcal{O}(dN)$, where N is the cardinality of my set of patches and d is the dimensionality of the patches. A more efficient way can be achieved by organizing the data into a search tree and partitioning that tree. Thereby, large portions of the search space can quickly be eliminated. In literature, such a tree is called a k-d-tree [82], short for k-dimensional tree. This representation partitions the data into a binary tree where each node divides the space into two parts. Such a representation can typically reduce the search effort to $\mathcal{O}(\log N)$. In the context of image inpainting, the usage of k-d-trees has been proposed e.g. by He et al. [83]. However, with increasing dimensionality d, the search effort increases rapidly. In literature, this problem is known as the *curse of dimensionality*, an expression coined by Richard E. Bennlan [84].

Fortunately, in my application it is not necessary to know the exact nearest neighbor in the search space, i.e. the one with the absolute smallest distance. Instead, it is sufficient to search for a matching patch that is reasonably close to the query patch $\mathbf{q}$. The assumption

here is that a patch which is very similar to the query region may fill the gap well enough. The patch is considered to be a $(1 + \varepsilon)$-nearest neighbor if the distances $D(\mathbf{p}, \mathbf{q})$ between a patch $\mathbf{p}$ and $\mathbf{q}$ and between $\mathbf{q}$ and its nearest neighbor have a ratio of at most $(1 + \varepsilon)$. A solution for this relaxation of the problem was first proposed by Arya and Mount [85] and is known as an Approximate Nearest Neighbour (ANN) query. The aforementioned example of the *k-d*-tree query can easily be made approximate by stopping the search after a predefined number of points in the tree have been visited or after a certain runtime. The algorithm of Arya and Mount can reduce the query time to $\mathcal{O}(1/\varepsilon)^d\mathcal{O}(\log N)$ and the pre-processing time (the tree has to be built first) to $\mathcal{O}(1/\varepsilon)^d\mathcal{O}(N)$.

4.4.2. Hashing Methods

The search for a matching candidate may be sped up even more by introducing a hash table. The idea is that when I use a suitable hash function and create a hash table for all the patches, any collision in the fingerprint of the patch may indicate a close match. I therefore only have to compare the patch to the ones in same bucket of the hash table, making the search effort sublinear [86]. The performance of the algorithm is therefore strongly dependent on the hash function. In the context of nearest-neighbor searches, I need to find a hash function that is sensitive to *locality*, which means that points that are close in space will have a high probability of collision, i.e. landing in the same bucket.

An algorithm for creating such a hash function was first proposed by Indyk and Motwani [86] and is known as Locality Sensitive Hashing (LSH). They have proven that such functions exist for any domain which leads to ε-NNS algorithms with preprocessing costs of $\mathcal{O}(d)$ and sublinear in N. I will present a rough sketch of their algorithm but refer the inclined reader to their paper. First, they partition the search space by random hyperplanes into different regions, with bits of the hash indicating on which side of the hyperplane the point resides. Points with the same hash therefore indicate points residing in the same region of the space, however, not necessarily the closest points as there may be nearer neighbors in a different region. This partitioning is therefore executed multiple times, each time with a different set of random hyperplanes. After all the candidate points in their respective region have been collected, we can then perform the search for the nearest neighbor.

While the algorithm propsed in [86] is limited to points in Hamming space, i.e. restricted to binary data, Datar et al. [87] presented an algorithm extending LSH to operate directly in Euclidean space without the need for embeddings of the l_2 space into Hamming space. I will explain this in detail in Section 4.4.3.

LSH and k-d-trees perform very well but are still not fast enough for my application. Another algorithm called *PatchMatch* [88] has been proposed by Barnes et al. for the task of finding matching patches. This algorithm exploits the fact that images are usually *coherent*. If a matching patch was found in an image, the coherency property indicates that patches in the close neighborhood of this pair of patches will also be similar. While the algorithm has the same complexity as k-d-tree-based algorithms of $\mathcal{O}(\log N)$, the authors show that in practical applications, their algorithm is faster by a factor of 20-100.

PatchMatch is employed for the hole filling in view synthesis in several algorithms [89, 90, 91], including very recent ones such as [92].

Korman and Avidan [93] proposed an algorithm called *Coherency Sensitive Hashing (CSH)*, combining the advantages of LSH and PatchMatch. As I've developed a hole-filling algorithm which is based on their hashing algorithm, I'm going to explain it in detail and then propose my hole-filling algorithm based on CSH.

4.4.3. Coherence Sensitve Hashing

CSH uses the set of locality sensitive hashing functions

$$h_{\mathbf{a},b}(\mathbf{v}) = \left\lfloor \frac{\mathbf{a} \cdot \mathbf{v} + b}{r} \right\rfloor \tag{4.9}$$

proposed by Datar et al. [87], where $\mathbf{a}$ is a random vector of dimension d drawn from a p-stable distribution and b is a random real value drawn uniformly of the range $[0, r]$. This hash function maps a d-dimensional vector $\mathbf{v}$ onto the set of integers $h_{\mathbf{a},b}(\mathbf{v}) : \mathbb{R}^d \rightarrow \mathbb{N}$. The input vector $\mathbf{v}$ is projected onto the random line $\mathbf{a}$, which is divided into equidistant segments of size r. The hash value is then the index of the bin into which the input is projected. b is used to offset the quantization effect of the binning.

Instead of projecting onto random vectors $\mathbf{a}$, Korman and Avidan suggest to use the most

significant 2-dimensional Walsh-Hadamard kernels [94] as a hash function

$$h_{\mathbf{j},b}(\mathbf{v}) = \left\lfloor \frac{WH_j \cdot \mathbf{v} + b_j}{r} \right\rfloor . \tag{4.10}$$

Their rationale for selecting a Walsh-Hadamard transformation lies in its descriptiveness for pattern matching in images as shown by Hel-Or and Hel-Or [95] and its fast computation (2 additions per patch and kernel [96]). They claim that a hash table built of these functions leads to local sensitivity in the appearance plane, a property I can exploit for finding matching patches for hole filling.

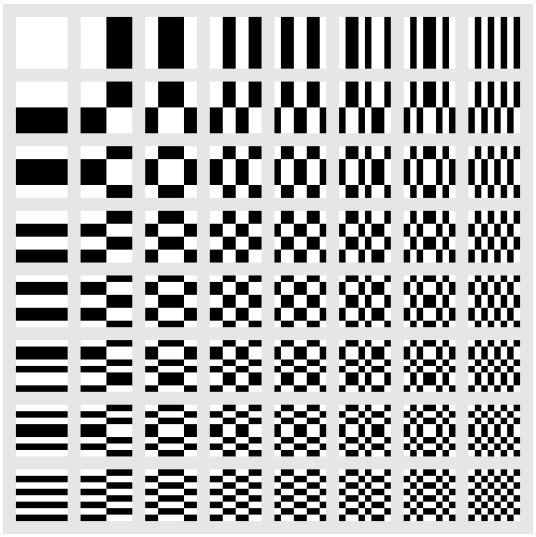


Figure 4.8.: Illustration of the 64 Walsh-Hadamard kernels of size 8 $\times$ 8 pixels.

After indexing all patches of the source and target image (which for image inpainting might very well be the same image), I should find patches of similar appearance with the same hash. However, this set of similar patches might be very small. Korman and Avidan thus combine this LSH approach with the findings of Barnes et al. [88] in their PatchMatch algorithm.

PatchMatch exploits the local coherency property of images. If the left neighbour of the candidate patch, that is the patch which is shifted by one pixel to the left, has a good match in our target picture, the right neighbour of this match will most likely be a good hit for the original candidate patch. The same holds true for neighbours on top, the bottom or the

right side of the candidate patch. Korman and Avidan extend the LSH algorithm with this finding and thus arrive with three types of candidate patches:

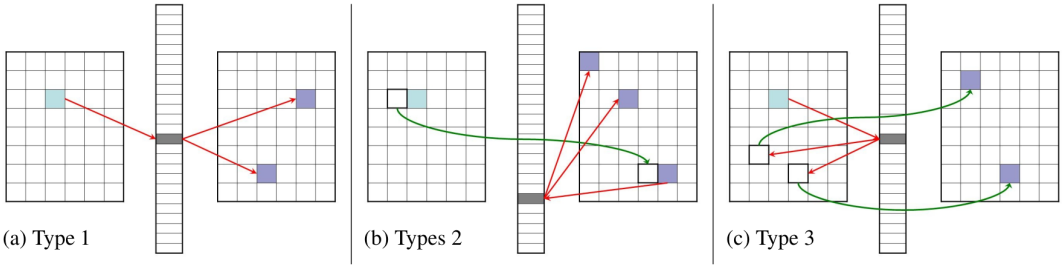


Figure 4.9.: The three different types of candidate patches which the CSH algorithm selects. Patches of Type 1 are those which have the same hash as the query patch. Patches of Type 2 are found by looking up hash collisions of the left neighbor of the query patch. Because of the coherence property, the right neighbor of this patch will also be a good candidate for the query patch. Patches of Type 3 are found by looking up the hash collisions in the query image and finding their neighbors in image B [93].

appearance-based

Type 1 patches are the ones which can be found by using the LSH method. If candidate patches b of image B have the same hash as the query patch a from image B, then these patches will be good matches for a: $g_B^{-1}(g_A(a))$

coherency-based

I search for the nearest patch in the target image B which has the same hash as the left neighbour of the candidate patch a. The local coherency property then suggests that the right neighbour of this patch will be a good candidate for the candidate patch. The list of candidates is further extended by looking up patches with the same hash as these candidate patches.

appearance-based

I search for patches in the query image A with the same hash as the query patch a. These patches will have good candidates for a in their matches in image B.

After this search, each query patch will typically have a set of $4k + 2$ candidate patches, with k being the number of entries in the hash table at each bin. Out of these candidates, the best matching one, i.e. the one with the lowest distance to the query patch, will be selected. As this task again would be computationally relatively expensive, Korman and Avidan suggest to use the Walsh-Hadamard transformation again for each candidate on

the kernels with the highest frequency and select the one which minimizes the sum of projected differences.

Once this algorithm has been applied to each patch of the query image, one can generate a dense approximate nearest neighbour field (ANNF) for one image. This field contains a matching patch to each query patch in the query image. In the following, I show how to modify this ANNF-algorithm to be suitable for the view synthesis problem, how to use the resulting ANNF for inpainting and propose the complete new algorithm for hole filling.

4.4.4. CSH for View Synthesis (VS-CSH)

(a) (b) (c)

Figure 4.10.: This comparison shows how incorporating the information of the depth map can significantly improve the inpainting result of a CSH-based inpainting step. Image (a) shows the input image with the disocclusion, Image (b) shows how the result looks when using CSH without any depth information and (c) shows the result when guiding the inpainting step with the depth information. Image (b) shows areas where foreground content has been filled into the disocculsion, leading to an unnatural result.

To this date, I'm not aware of any view synthesis algorithms using CSH in the inpainting step. I would therefore describe this algorithm as novel. Nevertheless, it does adopt some of the groundwork in the algorithms described by Barnes et al. [88], Fitzgibbon et al. [97] and Wexler et al. [98]. In the remainder of this thesis I will address the following algorithm as *VS-CSH*.

In the first step, I create a virtual image $\mathcal{I}_{\text{warp}}$ with the methods described in Chapter 2. The original view $\mathcal{I}$ is transformed using the depth map $\mathcal{D}$ into a virtual view $\mathcal{I}_{\text{warp}}$ and a mask

$\mathcal{M}$ is created which marks the holes $\mathcal{H}$ in the image. For a change, I took the the following examples from the sequence *Lovebird*, provided by Electronics and Telecommunications Research Institute (ETRI) and the MPEG Korea Forum [99]. I will provide more examples in the evaluation chapter where all algorithms will be evaluated on the same data set.

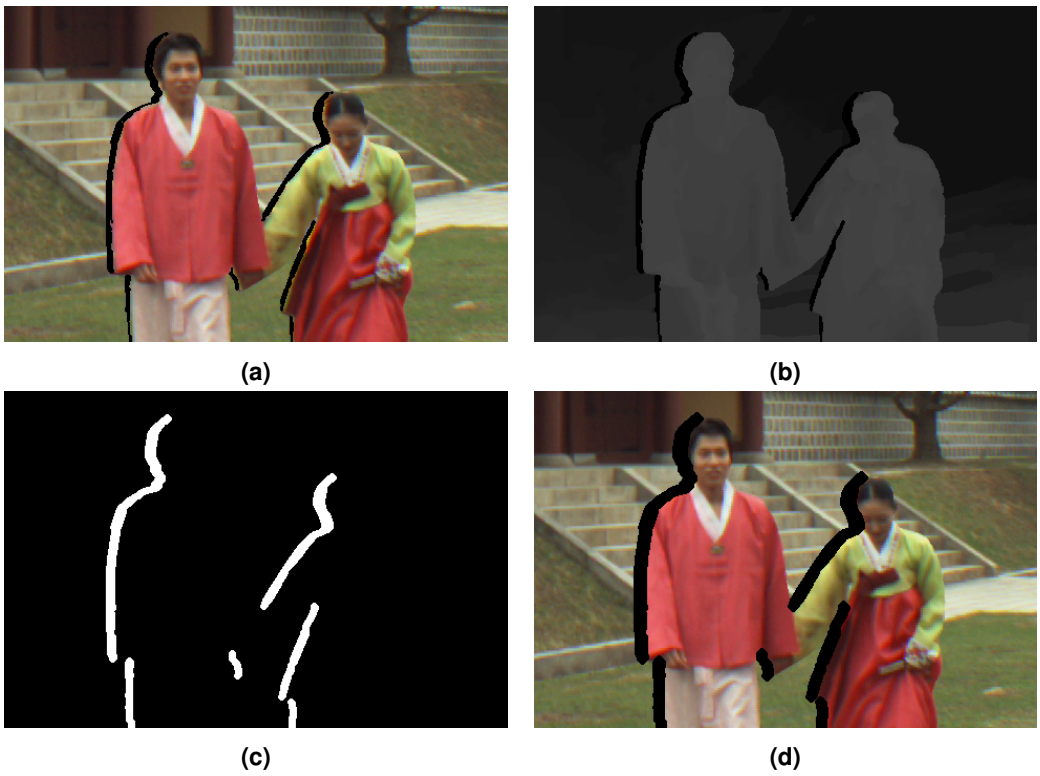


Figure 4.11.: Pre-processing steps for the VS-CSH algorithm of the first frame of the *Lovebird* sequence. (a) is the synthesized virtual view with holes. One can see remnants of the background at the outermost seam of the foreground objects due to imperfections in the depth map, depth quantization errors and bleeding of the colors. (b) shows the synthesized depth map, (c) is the resulting mask after the dilation operation, and (d) shows the final view with enlarged holes.

The pre-processing steps are essentially the same as in the previously described algorithms. The transition between the foreground object to the hole is not perfect due to imperfections in the depth map and quantization errors, so there are usually some background artifacts around the outermost seam of the foreground object (see Figure 4.11a). To avoid ghosting artifacts in the resulting inpainted image, I'm removing this seam from the foreground object using a simple binary morphological operation on the mask, namely

a dilation

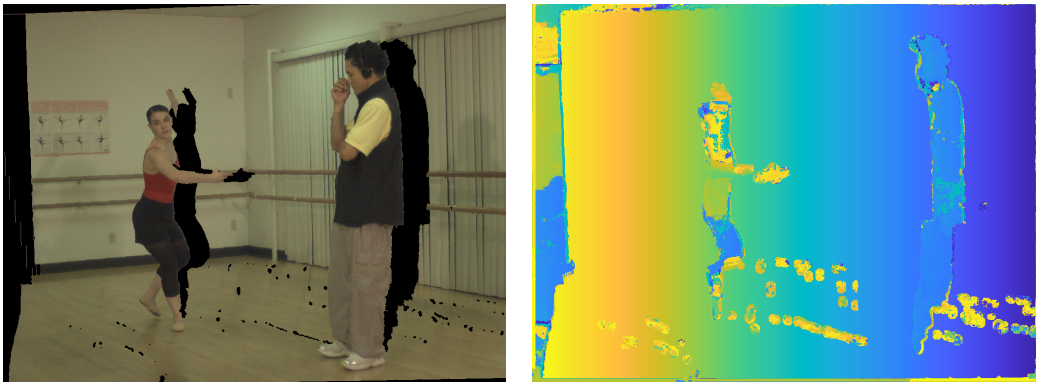
$$\mathcal{H} \oplus K = \bigcup_{k \in K} \mathcal{H}_k \tag{4.11}$$

with the filter kernel

$$K = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}. \tag{4.12}$$

This will essentially enlarge the mask on the left-hand side by one pixel, which will remove these seams. Please note that for the sake of brevity, and without loss of generality, I describe this algorithm for a virtual view which is located on the right-hand side of the original view. The algorithms can trivially be modified for views at other positions, in the simplest case by mirroring all input and output images.

Using the image, the mask and the depth map, I'm then calculating an ANN-field (short ANNF) which for each pixel p in the synthesized view $\mathcal{I}$ provides the location of the source patch. For known content, the source location is simply the location of the pixel itself, but in the regions of the disocclusion, the ANN field points to the location of the patch which fits best at that location. To create this ANN field, I'm using a modified version of the CSH algorithm, which I will describe shortly. In all following examples, the patch size was chosen to be $w = h = 8$. Note that the patch size has to be a power of 2 because the Walsh-Hadamard kernels need to have a scale of a power of 2.



(a) **(b)**

Figure 4.12.: Input image and the corresponding ANN field (only the horizontal component is shown here). The color map indicates the position of the source patch. The ANN field shows a mostly linear color gradient over the image, which means that the source position is the pixel itself. In the disoccluded region, the color indicates where the pixel which fills that hole is taken from.

To compute the ANNF, I'm creating a copy of the image $\mathcal{I}$ where I'm initializing the holes

with random content, i.e., drawing each pixel from uniformly distributed random color values. I then calculate the ANNF between the image with holes $\mathcal{I}$ and the randomly filled image $\mathcal{I}_{rand}$. This ensures that only the areas outside of the hole are contributing when matching to a suitable source patch as it is unlikely that the random pixels are found in a source region.

To calculate the ANNF, in the first step, both images are segmented into overlapping patches. All the patches from $\mathcal{I}$ and $\mathcal{I}_{rand}$ are then projected onto M Walsh-Hadamard kernels $\{WH_j\}_{j=1}^{M}$. In my examples, M was chosen as $\left(\log_2(w)\right)^2 \cdot n_{\text{Channels}} = 27$.

Using the hash functions from Equation 4.9, I create a hash table by concatenating M functions $\{h_j\}_{j=1}^{M}$ to a code

$$g_i(\mathbf{p}) = h_1(\mathbf{p}) \circ \cdots \circ h_M(\mathbf{p}).$$

Using this hash code, the signatures of all patches of both images are then inserted into a hash table T. Due to the local sensitivity of this approach, patches which are proximate in the appearance space, i.e., patches which are looking similar to each other will land in the same bin in this hash table. This indexing step therefore has reduced the number of contemplable patches significantly.

In the next step, for all these patches with colliding hashes the number of suitable patches is increased by the previously described coherency-based expansion to $4k+2$ candidates. This ensures that not only the patches which were found by the locality sensitive hashing step are considered but also those which are in a local proximity to the candidates, thereby increasing the set of candidates for one patch. I can then search for the one which matches best by comparing all cadidate patches to the query patch with a suitable measure such as SSD. While this is still a computationally expensive operation, the effort is much lower than having to compare against all possible patches.

This is also the step where I'm introducing the information of the depth map into the matching algorithm. As I've explained in Section 4.3.4, for a visually plausible completion of the holes introduced in the view synthesis, the algorithm is only considering patches of the background to fill the holes. Therefore, here the algorithm is not only comparing the patches of the image but also their corresponding patches of the depth map. If a candidate patch has on average a depth value which is smaller than the query patch, this patch will

not be considered for the filling. I can thereby ensure that only background content will be filled into the hole.

As a last step, the missing pixels $\mathbf{p} \in \mathcal{H}$ are filled using the scheme proposed by Wexler et al. [98] where the color of that pixel

$$c_p = \frac{\sum_{i \in \mathcal{N}_p} \alpha_p^i s_p^i c^i}{\sum_{i \in \mathcal{N}_p} \alpha_p^i s_p^i}$$

is calculated by a weighted mean of all ANN-patches which contain $\mathbf{p}$, with a weighting factor

$$w = \alpha_p^i s_p^i$$

which comprises a measure for the similarity

$$s_p^i = sim(W_p^i, V^i) = e^{-\frac{d(W_p^i, V_p^i)}{2\sigma^2}}$$

between the query patch and its AN-neighbour. α_p^i is a factor which denotes whether a point is part of the hole $\mathbf{p} \in \mathcal{H}$ or of the image area outside of the hole $\mathbf{p} \in \mathcal{I} \setminus \mathcal{H}$. Areas inside the hole will get a low confidence while areas outside of the hole will get a high confidence. Wexler suggests to use a distance measure $\alpha_p = \gamma^{-dist}$ to the boundary of the hole with $\gamma = 1.3$.

This filling is repeated iteratively. The image with the inpainted pixels is then again used to initialize the next iteration instead of the random initialization of the hole of the first step. In the next iteration, another ANNF between this inpainted image and the original image with the holes is created to refine the inpainted content. In each iteration, I'm using SSD to measure how much change there is in the picture. After a maximum number of iterations ($n_{Iter} = 30$) or when the change in the picture falls below a certain threshold ($\epsilon < 0.1 \cdot w \cdot h$, where w and h are the size of the image), the iterations are stopped. Typically, the algorithm converges relatively quickly to a suitably small threshold in about 3 iterations, however, there is no guarantee for a convergence which can sometimes be observed by an oscillation of the change in the SSD between consecutive iterations.

This algorithm as described so far can only be applied for very small holes because the size of the holes is limited by the patch size. Should the hole be larger than the patch

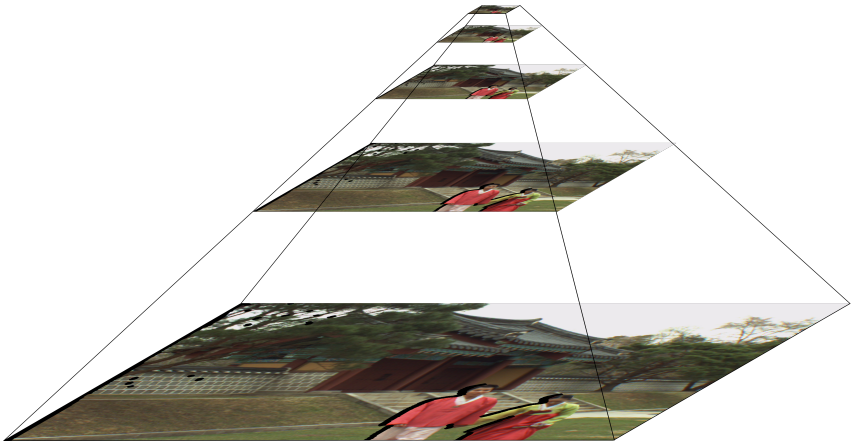


Figure 4.13.: Hole filling is started on the lowest resolution level of a log-spatial pyramid. This facilitates filling of holes which are larger than the patch size and enables faster convergence in the next level due to propagation of the result.

size, the query patch would consist of random content only. I'm therefore introducing a log-spatial pyramid. The image is scaled down by a factor 2^n where $n = -5 \dots 0$, which ensures that the holes in the lowest resolution level are smaller than the patch size. The lowest resolution scale depends on the size of the largest hole and the patch size. In my examples a starting scale of 2^{-5} has been sufficiently small. After the image has been inpainted in the smallest level, the image is scaled up to be used as the initialization in the next level. This guarantees global consistency across the hole and faster convergence in the high-resolution levels of the pyramid.

Algorithm 1 shows the complete *VS-CSH* algorithm in pseudo code.

4.4.5. Results

Figure 4.14 shows the result of the inpainting of the disocclusions of the last frame of the *Lovebird* sequence. I'm showing the last frame here because it has the largest holes of all frames due to the persons walking closer to the camera over the course of the sequence. Image 4.14a shows a warped virtual view as would be seen from the position of Camera 7; using the image of Camera 6 as the reference image. Image 4.14b shows the inpainted image. Especially when looking at the stairs in the background one can see how well the algorithm preserves the structure of the stones, while making sure that none of the

foreground is bleeding into the background. Figures (c) and (d) show some cropped parts of the image to better visualize the details.

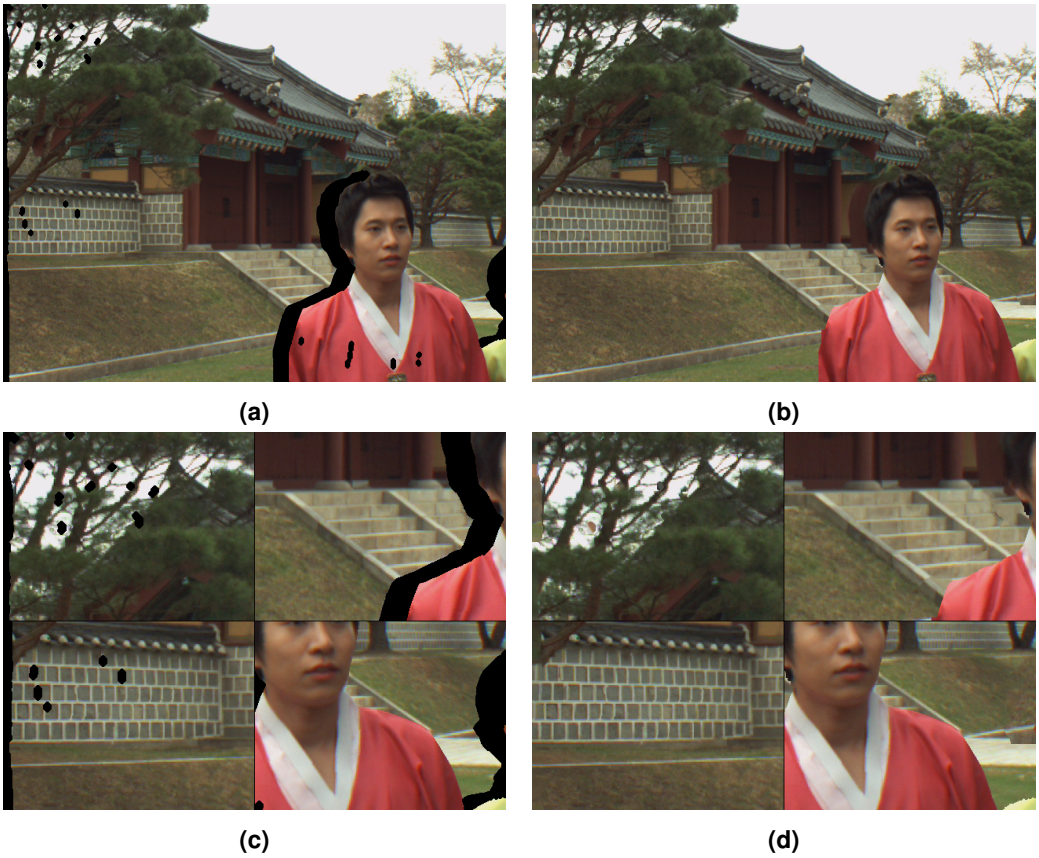
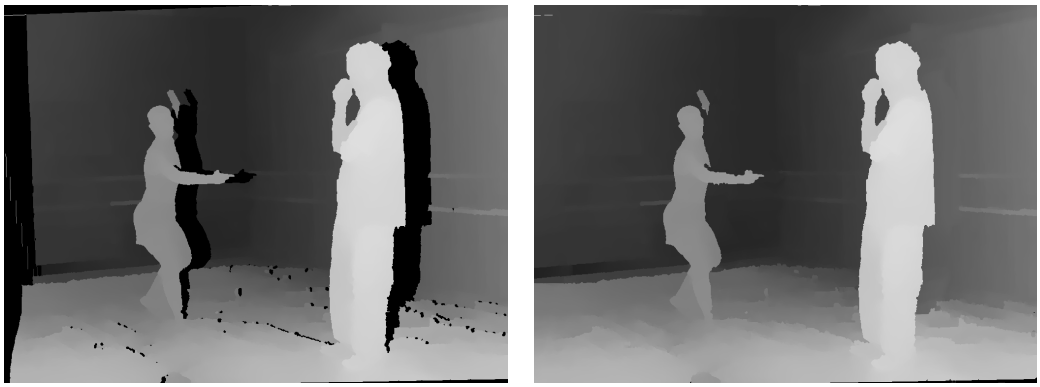


Figure 4.14.: Inpainting results for the last frame of the *Lovebird* sequence of the proposed algorithm. (a) shows the warped image with the holes in black, (b) shows the inpainting result of the VS-CSH algorithm. (c) and (d) show selected cutouts for improved visibility of the details.

Figure 4.15 shows the reconstruction of the depth map of the VS-CSH algorithm. The depth can be reconstructed at the same time as the image is being inpainted, simply by selecting the patches in the depth image which are at the same location as the patch of the visual image and filling these into the disoccluded depth map.



(a) **(b)**

Figure 4.15.: This picture shows the completion of the depth map by the VS-CSH algorithm. (a) shows the input depth map with holes, (b) the reconstructed result

Part III.

Evaluation

5. Objective Evaluation

5.1. Introduction

To evaluate the performance of the algorithms I've presented so far, suitable measures need to be found to compare the performance of the algorithms against other state-of-the-art algorithms. The goal of these algorithms as formulated in the course of this thesis was to produce images where the holes have been filled in a *visually plausible* manner. This already indicates that the synthesized content doesn't necessarily need to be exactly the same as the original view would be at that point. Measures which compare the image to the ground-truth image on a pixel-by-pixel basis such as PSNR are therefore only of limited expressiveness. Nevertheless, I've included them because they can be easily compared against published numbers of existing algorithms. More appropriate methods would be ones which don't compare pixel values but rather the structure of the image such as SSIM [60]. Even better, there exist metrics which take into account the processes occurring in the human visual system, which might be able to give us a good indication of how similar the synthesized image might be to the ground truth image in the eyes of a human observer.

Researchers have investigated the properties of objective image metrics extensively because they exhibit a number of advantages compared to subjective testing in a laboratory. Objective metrics allow for a quick analysis of the performance of algorithms already during the development stage without the need to conduct time-consuming tests associated with high effort and therefore high cost. Especially in the field of video coding, there exists a wealth of analyzes of different objective metrics for exactly this reason. In many recent publications, researchers were able to demonstrate that objective metrics for video coding artifacts can provide results which are as good as tests performed in a laboratory setting,

with deviations from the measured lab results which are well below deviations in between different labs [100, 101, 102].

5.2. 2-D Objective Metrics

In 2011, Bosc et al. [103] conducted a comprehensive analysis how well classical objective metrics used for assessing 2-D images may be applied to synthesis results of 3D-DIBR algorithms. In a subjective study, she generated four viewpoints of three different sequences using seven different view synthesis algorithms. 43 observers were then tasked to perform an absolute category rating (ACR [104]) on the generated views, which were then averaged to get the mean opinion score (MOS). These ratings were then compared to the result of 12 objective metrics, namely PSNR, SSIM [60], MSSIM [105], VSNR [106], VIF [107], VIFP [107], UQI [108], IFC [109], NQM [110], WSNR [111], $PSNR_{HVSM}$ [112] and $PSNR_{HVS}$ [113]. She found out that WSNR (weighted signal-to-noise ratio) had the highest correlation to the MOS, with a correlation coefficient of 0.423, followed by PSNR (0.386) and NQM (0.386, also). These low correlation values indicate that apparently none of these metrics is particularly well suited to the quality degradations introduced by the DIBR process.

5.3. Objective Metrics for DIBR

Battisti et al. [114] had a closer look at these specific artifacts. They identified six main types of DIBR-related distortions: object shifting, incorrect rendering of textures, blurring, flickering, geometry and depth distortions. Based on these findings, they developed an objective metric called 3DSwIM which is tailored to evaluate view synthesis results. Their metric is based on three assumptions. Given that the rendering process displaces elements of the image in the horizontal direction between different viewpoints, they introduce a block registration step in their metric, giving it a "shifting-resilience" [114]. This is one of the main differentiators to classic pixel-based metrics. Also, they assume that the image synthesis process would introduce artifacts which would be especially noticeable in the component of a Haar-wavelet transform related to the horizontal image features. The

idea is that the holes in the image are oriented vertically in the image due to the horizontal camera shift so that differences between the synthesized and the reference view should manifest in a change of the horizontal frequency spectrum. Lastly, they propose that a human observer would be specifically sensitive to changes in human skin, e.g. distortions of the faces of people in the images. Therefore, they incorporate a skin detection algorithm in their metric. They as well performed a comparison of the results of their metric with the subjective study performed by Bosc et al. [103]. They showed that 3DSwIM has a Pearson correlation coefficient of up to 0.7617 with the MOS values of the subjective study, indicating that this metric might be much better suited to replicate the rating of a human observer.

The fairly recently proposed DSQM metric by Farid et al. [115] achieves even higher coefficients of correlation on the dataset of Bosc et al. Their metric is not quite a full reference metric in that it doesn't use the ground truth reference image from a camera at the synthesized viewpoint but only the original image which has been warped as a reference. This metric uses a patch-based normalized cross correlation in a first step to find the location to where the patch has been shifted in the synthesized image. Then, they apply the phase congruency model [116] as a perceptual feature extractor. The phase congruency model states that perceptually important features such as edges, lines, mach bands etc. are located at points where all fourier components of the signal are in phase. This feature is extracted on the luminance channel of the image which has been converted from RGB to YIQ. Farid et al. then compare the mean value of the phase congruency of all blocks and use that as a measure for the distortion introduced by the DIBR process. Thereby, they achieve a Pearson linear correlation coefficient of 0.7895 and a Spearman's rank order correlation coefficient of 0.7151 with the MOS values of the IRCCyN/IVC DIBR image data base [103]. This outperforms not only 2-D-IQA algorithms such as SSIM, Multi Scale SSIM (MS-SSIM) [105], PSNR or VSNR [106], but also metrics tailored especially for the assessment of synthesized 3-D content such as 3DSwIM, Tsai [117], PQM [118], SIQE [119], MW-PSNR [120] and MP-PSNR [121].

5.4. Test Sequences

To evaluate the performance of my algorithms, I use two well-known Multiview Video-plus-Depth sequences which seem to have become the de-facto standard sequences in literature to evaluate view synthesis hole-filling algorithms. This allows for a fair comparison against published results of the state-of-the-art algorithms. The first one is the *Ballet* sequence from the Interactive Visual Media Group at Microsoft Research [18] which I've chosen to include in the comparison because of its large baseline. This leads to comparably large holes and therefore poses a unique challenge for hole-filling algorithms. Furthermore, the background has very clearly defined structures such as the bars and the regular curtain which makes it easy to visually compare the results.

The scene was filmed with 8 cameras mounted on an arc with 30° angular separation and the images have a resolution of 1024×768 pixels. Zitnick et al. [18] developed a novel iterative algorithm to calculate the depth-map based on the disparity between two cameras and I've used their published depth-maps in the warping step of the algorithm. They've also published intrinsic camera parameters and a rotation matrix for each camera.

For my evaluation I take the view from Camera No. 5 and create a virtual view which would be seen from Camera No. 4. I can therefore use the image from Camera No. 4 as a ground truth reference. I use the MPEG View Synthesis Reference Software (VSRS) [73] in version 3.5 to generate the virtual view and use my algorithms to fill the disocclusions.

Unfortunately, Zitnick et al. used different conventions for their transformation from world to camera coordinates than the MPEG VSRS software expects so their published matrices can't be used in the VSRS software. Zitnick et al. provided the rotation matrices and translation vectors according to

$$\mathbf{x}_c = \mathbf{R}\mathbf{x}_w + \mathbf{t}$$

while the VSRS software expects an inverse external transformation

$$\mathbf{x}_w = \mathbf{R}\mathbf{x}_c + \mathbf{t}.$$

In Chapter 2, I provide the derivation of the conversion of one representation into the other in Equation 2.2. It is therefore necessary to convert the parameters provided by the

Microsoft Research (MSR) team according to

$$\mathbf{R}_{\text{VSRS}} = \mathbf{R}_{\text{MSR}}^{-1}$$

and

$$\mathbf{t}_{\text{VSRS}} = -\mathbf{R}_{\text{MSR}}^{-1}\mathbf{t}_{\text{MSR}}.$$

This conversion is also described in detail in an MPEG document by Smolic et al. [122].

The other sequence I've used for the evaluation is the *Lovebird* sequence from the Electronics and Telecommunications Research Institute (ETRI) of the MPEG-Korea Forum [99]. This sequence has a resolution of 1024×768 pixels as well. The depth-maps of this sequence were generated using the MPEG Depth Estimation Reference Software (DERS) [17]. I've chosen to include this sequence in the evaluation because, in contrast to the Ballet sequence, this sequence was filmed outdoors which leads to a much more irregular and more natural background, providing a challenging task for the hole-filling algorithms. Just as the MSR sequence, this sequence has also gained some popularity among researchers, so comparisons against published performance numbers are easily possible. In this sequence, I use the images of Camera No. 6 and transform them to the viewpoint of Camera No. 7 using the VSRS software.

5.5. Performance Overview

Table 5.1 shows the results of the objective metrics on the synthesized view of the Ballet sequence averaged over all 100 frames of the sequence. I'm comparing the performance of both my algorithms VS-BP and VS-CSH against the the MPEG VSRS reference implementation [73] which uses Bertalmio's inpainting algorithm [62], the algorithm of Daribo et al. [38], Gautier's algorithm [40] and two very recently published algorithms by Li et al. [33] and Luo et al. [92]. All algorithms which I could evaluate on my own were fed with the same input material (disoccluded view, depth map and hole mask) which I've created with the VSRS software with the same parameter settings for all algorithms. The VSRS software is designed to output complete warped and filled images, so I've modified the source code to output the intermediate images which I then passed on to the other hole-filling algorithms. The VSRS software does not have any further parametrization for the hole-filling step. The

Table 5.1.: Objective evaluation of the inpainting result

	Tanimoto2008 [73]	Daribo2011 [38]	Gautier2011 [40]	Li2019 [33]	Luo2020 [92]	VS-BP	VS-CSH
$PSNR_Y$ [dB]	28.7	30.3	31.4	31.5	32.1	**33.2**	32.5
$PSNR_Y$ holes only [dB]	19.2	24.2	24.0	-	-	**26.2**	24.9
SSIM	0.92	0.87	0.88	**0.93**	0.87	**0.93**	**0.93**
SSIM holes only	0.60	0.68	0.69	-	-	**0.72**	0.71
3DSwIM	0.52	0.33	0.35	-	-	**0.53**	0.49
DSQM	0.28	0.49	0.49	-	-	**0.25**	**0.25**

algorithm by Daribo was run with the default window size of 9×9 pixels proposed by Criminisi and the depth prioritization parameter β was manually tuned to give the best possible result. Gautier's algorithm was run with the parameters provided with the software. Unfortunately, the implementation which was provided to me by the author doesn't support the full resolution of 1024×768 pixels so the images were resampled to half their original size which actually improves the PSNR value of the result by a little margin. I was not able to get a reference implementation of the algorithms of [33] and [92], so I used the values which they published in their articles. Unfortunately, this means that I can only compare against selected metrics which were published.

The objective metrics for this comparison were PSNR on the luma channel of the image, SSIM, 3DSwIM and DSQM. Note that for DSQM, lower values are better. I've provided additional values for PSNR and SSIM which were evaluated on the holes only, i.e., the metrics were only applied on the inpainted content. One can see from Table 5.1 that VS-BP provides the best hole-filling result of all the algorithms in this evaluation in terms of the applied objective metrics. VS-CSH is a close contender on the second place for almost all metrics except 3DSwIM where, suprisingly, the VSRS software scores the second place. Only for SSIM, the algorithm of Li et al. [33] performs equally well.

5.5.1. Performance over Time

Figure 5.1 shows how the values of selected objective metrics change over the course of the complete *Ballet* sequence. The red line is the result of VSRS [123], yellow is the result of Gautier's algorithm [40], blue is the result of VS-CSH and violet is the result when the hole is filled with the ground truth content of Camera No. 5. Unsurprisingly, all hole-filling algorithms perform worse than the ground truth, however, one can see that VS-CSH is performing the best out of these algorithms.

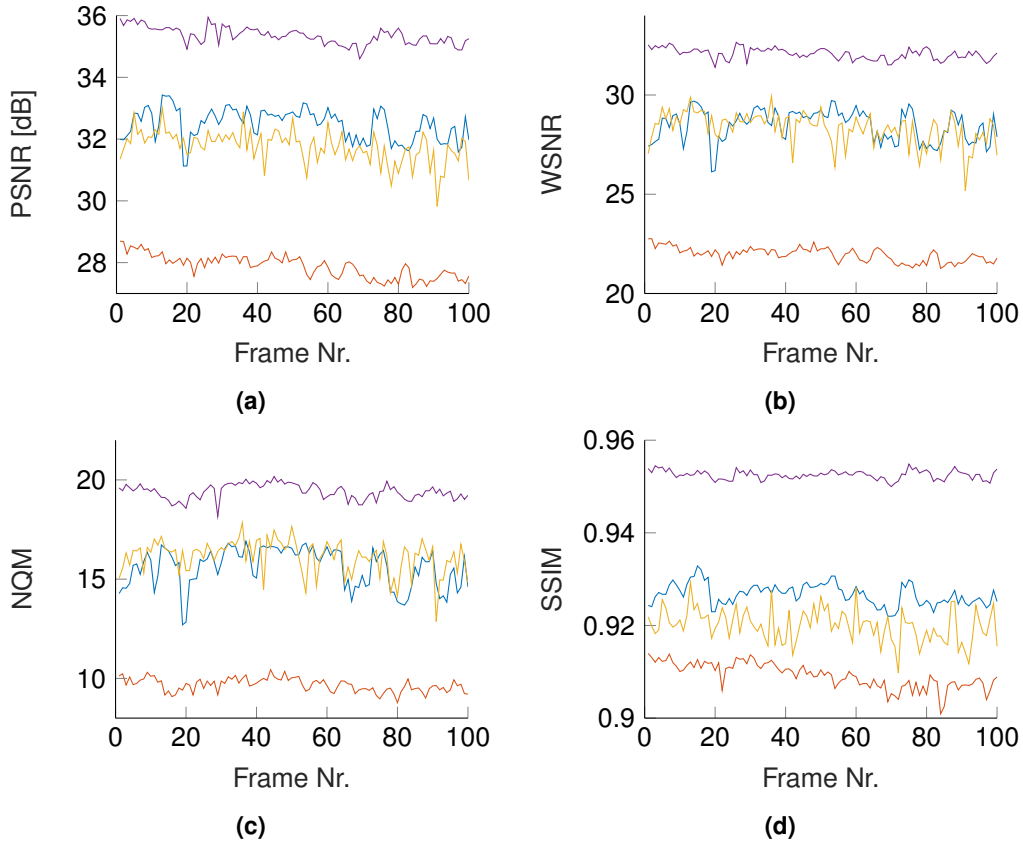


Figure 5.1.: Course of the objective metrics over the complete *Ballet* sequence. The red line is the result of VSRS [123], yellow is the result of Gautier's algorithm [40], blue is the result of VS-CSH and violet is the result when the hole is filled with the ground truth content of Camera No. 5.

5.5.2. Temporal Consistency

Temporal consistency is an important aspect of disocclusion filling algorithms. As most of the algorithms presented in this thesis operate on individual frames of a video, it is not guaranteed that the synthesized regions of the output images remain consistent over time. When these individual images are then played sequentially, the observer will experience more or less noticeable flickering of in the image in the synthesized regions where the content may change rapidly from one frame to the next, especially around the borders of the foreground objects. This makes the disocclusion filling process more apparent as the eye is more sensitive to rapid changes in the video sequence [124], thus counteracting the goal of visually plausible content generation.

To evaluate the temporal consistency of the inpainting results, I'm using a measure called *Frame Differential Flicker* (FDF) as described in [125]. This measure analyzes the amount of change in the disoccluded areas of an image sequence and compares it to the naturally occurring change in the ground truth sequence.

In the first step, this measure calculates the difference between two consecutive images I_{t-1} and I_t for each pixel p over a so called *timeline* $T_p = \{I_t(p) : t = 1 \dots N\}$

$$\kappa_p = \frac{1}{|T_p|} \sum_t |I_t(p) - I_{t-1}(p)|,$$

which provides a value for the absolute flicker of a single pixel over the duration of the sequence, similar to how Intra-frame flicker is being measured in video coding [126]. This difference is then evaluated over the disoccluded areas in the video to provide

$$\kappa_D(I) = \frac{1}{|D|} \sum_{T_p \in D} \kappa_p,$$

where D denotes all the timelines in the disoccluded area of the image sequence.

Finally, this flicker of the disoccluded areas is compared to the flicker of the ground truth sequence G to obtain the *Frame Differential Flicker*

$$FDF(I) = |\kappa_D(I) - \kappa_D(G)|.$$

Table 5.2.: Frame Differential Flicker for two test sequences and different algorithms. The values marked with an asterisk (*) were taken from [92] and could not be verified independently.

Test Sequence	[73]	[40]	[92]	VS-BP	VS-CSH
$\text{Ballet}_{5\rightarrow4}$	1.77	3.68	3.7*	2.91	**0.15**
$\text{Lovebird}_{6\rightarrow7}$	**-1.07**	6.94	-	3.80	4.74

Schmeing and Jiang, the authors of [125], note that it may become difficult to find the timeline of a certain pixel in the disoccluded area when the foreground object is moving because the disoccluded area is changing its location, shape and size inside the frame. They therefore restrict their measure to static scene geometries with very simple objects of synthetic rendered videos. In my opinion, the limitation to static foreground objects is unnecessary. While it is true that the pixel value in the disoccluded area may change due to the foreground object moving in the scene, this change is equally reflected in the ground truth sequence and thus does not contribute to the FDF. I have therefore evaluated the FDF over the set union of the two disoccluded regions of two consecutive frames.

Table 5.2 shows the result of the evaluation. Both algorithms presented in this thesis compare favorably to even the most recent state-of-the-art algorithms. Flicker is significantly lower than of the algorithm of Gautier et al. [40] and better than the reference algorithm of the VSRS 3.0 [73] in the *Ballet* sequence. In this sequence, both algorithms are also better than the recent algorithm of Luo et al. [92]. Lacking an implementation of their algorithm, I could only compare to their published number for the *Ballet* sequence. VSRS performs relatively well because the Bertalmio inpainting algorithm it employs by design leads to very smooth filling results due to the underlying flow equations. This optimization on spatial consistency rather than visual acuity was already shown in the previous analysis. The FDF metric for this algorithm becomes negative because the flicker is even lower than the flicker of the ground truth sequence.

5.5.3. Runtime Comparison

The hole-filling algorithms I've presented so far exhibit significant differences in computational complexity which leads to significant differences in the runtime of the algorithms. While it is very difficult to make an objective comparison of different algorithms written in different programming languages with different levels of optimization, I briefly want to

present measurements for the runtime of all the algorithms and discuss the results with respect to their implementation details.

As the runtime of most of the algorithms depends on the area of the holes, I've executed them on all 100 frames of the *Ballet* sequence to get a good distribution over different sizes of holes. The holes of the *Lovebird* sequence are smaller on average, so the *Ballet* sequence with its large baseline should give a good estimation for the higher end of the execution time for a typical application of these algorithms.

Table 5.3 shows the runtime of the presented algorithms for a single frame of the *Ballet* with the median value and the lower and upper quartile indicated over all frames.

All of the algorithms in this comparison were executed on a PC with an Intel Core i7-5500U Dual-Core CPU with a maximum clock frequency of 3.0 GHz and 8 GB of RAM.

The fastest algorithm of this comparison is the inpainting algorithm of Bertalmio et al. [62] which is being used in the VSRS reference implementation. It is not surprising that this algorithm wins this comparison as it uses the inpainting method which comes with the OpenCV library[1], implemented in C++ and being heavily optimized. Also, this algorithm can use very efficient solvers for the underlying Navier-Stokes equations. However, it is also the algorithm with the worst image quality in this comparison.

For the algorithm of Gautier et al. [40] I was provided with the reference implementation by the authors in a binary format. The software was written in C++ as well and compiled using the Microsoft Visual Studio compiler. The algorithm is still relatively slow which can be explained by the exhaustive patch search it is employing which makes it relatively inefficient.

The VS-BP algorithm I've developed is written in Matlab and unfortunately uses a straightforward and therefore rather inefficient implementation of the belief propagation algorithm. This makes it one of the slowest algorithms in this comparison, comparable to the Criminisi inpainting algorithm. However, it makes up for its computational complexity by its superior image quality. I would expect that an implementation in C++ using an efficient belief propagation library such as OpenGM [127] to solve the MRF would make it significantly faster. An evaluation on a comparable inpainting problem conducted by the authors of this library

[1] `https://opencv.org/`

Table 5.3.: Runtime of different hole-filling algorithms on 100 images of the *Ballet* sequence.

Test Sequence	VSRS [73]	Gautier [40]	Criminisi [63]	VS-BP	VS-CSH
$Q_{0,25}$	9 s	57 s	5 h 12 min	3 h 43 min	27 s
Median	9 s	2 min 2 s	6 h 41 min	5 h 24 min	35 s
$Q_{0,75}$	10 s	2 min 20 s	7 h 23 min	6 h 12 min	41 s

showed an increase in the execution speed by at least one order of magnitude compared to different implementations of the same algorithm [127].

My VS-CSH algorithm was implemented in Matlab as well. However, I could make use of the excellent CSH implementation by Korman and Avidan [93] which is largely implemented in C and embedded in Matlab through the use of MEX functions. I've implemented all my extensions for using CSH for hole filling in C as well. This makes this algorithm one of the fastest in this comparison, while still retaining excellent image quality.

6. Subjective Evaluation

The framework for crowd-sourced subjective testing described in the next chapter has already been published in [128, 129, 130, 131, 132].

6.1. Introduction

Objective quality metrics are an invaluable tool in the assessment of the performance of image and video processing algorithms because they allow for a quick and inexpensive way to compare the performance of different algorithms. However, as I have described in the previous chapter, their expressiveness in terms of the image quality a human observer would observe is always subject to debate. While it seems to be good form to present the PSNR values when publishing a newly developed algorithm, numerous studies have shown that the correlation of PSNR and the mean objective score measured in a subjective study is very poor, and even simple examples can be found to demonstrate that different kinds of errors with significant differences in the perceived quality of an image can lead to the same PSNR value [133].

Especially in the context of this thesis, my formulated goal was to develop algorithms which are able to complete images in a way which is natural to a layperson. An ideal hole-filling algorithm would fill in the disocclusions in a way which is unnoticeable to a human observer. This suggest that there is no objectively quantifiable *correct* way to fill in the hole, let alone one that one could compare against some ground truth reference. To me, it therefore seems necessary that I conduct a subjective study to substantiate the results of the objective metrics that I have provided in the previous chapter.

Subjective studies, however, often come at a significant cost. One needs to have access to a specially equipped laboratory (which I was fortunate enough to have at the Institute for Data Processing of the TUM), the equipment should be calibrated in a time consuming procedure before the test and the test needs to be conducted with a rather large group of persons which have to be instructed individually. Due to the limited capacity of the laboratory, these tests also consume a significant amount of time, most often spreading over the course of several days or even weeks. Lastly, the test subjects are typically reimbursed for their time and effort, which amounts to a significant financial expense for conducting a study.

My colleagues and I therefore set out to find a simpler way to conduct subjective studies which is more adapted to the internet age. The result was the development of Quality-Crowd, a framework for quality evaluation in a web browser. In several studies, we have shown that the results of a subjective study conducted in such an online experiment are comparable to those which were conducted in a standard conforming laboratory environment. We were even able to show that the deviations between these two testing methodologies are smaller than one would expect when comparing the results of tests which have been conducted at different laboratories [132].

In this chapter, I will therefore present a subjective study which has been performed on the virtual views generated by my hole-filling algorithms. The purpose of this study is two-fold. With one single study, I want to assess the subjective image quality of the virtual views generated with the two algorithms presented in this thesis, but at the same time verify the validity of using QualityCrowd to conduct this experiment. My approach therefore is as follows. I recreate the study which was conducted by Bosc et al. [103] in the QualityCrowd framework instead of the laboratory and compare the results of both methodologies. The goal is to find out whether conducting the study in an online test yields comparable results to those of the laboratory experiment. At the same time, I'm extending the test set by virtual views views which have been generated with the algorithms presented in this thesis. Should the hypothesis hold true that the results of the original test set are comparable to those of the study of Bosc, I'm getting an assessment of the subjective quality of my algorithms at the same time, for which I otherwise would have needed to conduct a new subjective study in the lab.

6.2. Test Setup

6.2.1. The QualityCrowd Framework

QualityCrowd in the latest version 2 provides a server architecture which enables a very quick setup of subjective studies on the internet. It has only minimal requirements on the server software (an HTTP server like Apache with a recent version of the PHP interpreter) and can quickly be installed with an interactive installation assistant. Once installed, new subjective tests (so called *batches*) can be set up with a simple test description language called *QualityCrowd-Script*. It supports different media formats for still images and videos and different ratings scales such as Absolute Category Rating (ACR), either discrete or continuous, but also impairment scales such as Double Stimulus Impairment Scale (DSIS) or Pairwise Comparison (PC). Each test subject gets an individual link to access the study and QualityCrowd collects the results which can then be downloaded in CSV or Excel format. QualityCrowd2 also automatically generates statistics while the study is being conducted such as the progress of each test subject through the questions and meta information such as the time a subject spends at each question. QualityCrowd is Open Source Software and can be downloaded at the Github account of the Institute for Data Processing of the Technische Universität München[1].

6.2.2. Test Set

Bosc et al. [103] conducted their experiment with three different test sequences: the *Book Arrival* sequence of the Heinrich-Hertz Institute which has a resolution of 1024×768 pixels and was recorded with 16 cameras with a spacing of 6.5 cm; the *Lovebird* sequence provided by ETRI and the MPEG Korea Forum [99] which has a resolution of 1024×768 pixels and was recorded with 12 cameras with a spacing of 3.5 cm; and the *Newspaper* sequence by Gwangju Institute of Science and Technology (GIST), also with a resolution of 1024×768 pixels and a spacing between cameras of 5 cm. Out of every sequence, four novel views were generated. Adapting the notation of the previous chapter, these synthesized views are $\text{Lovebird}_{6\to7}$, $\text{Lovebird}_{6\to8}$, $\text{Lovebird}_{8\to6}$, $\text{Lovebird}_{8\to7}$,

[1] `https://github.com/ldvpublic/QualityCrowd2`

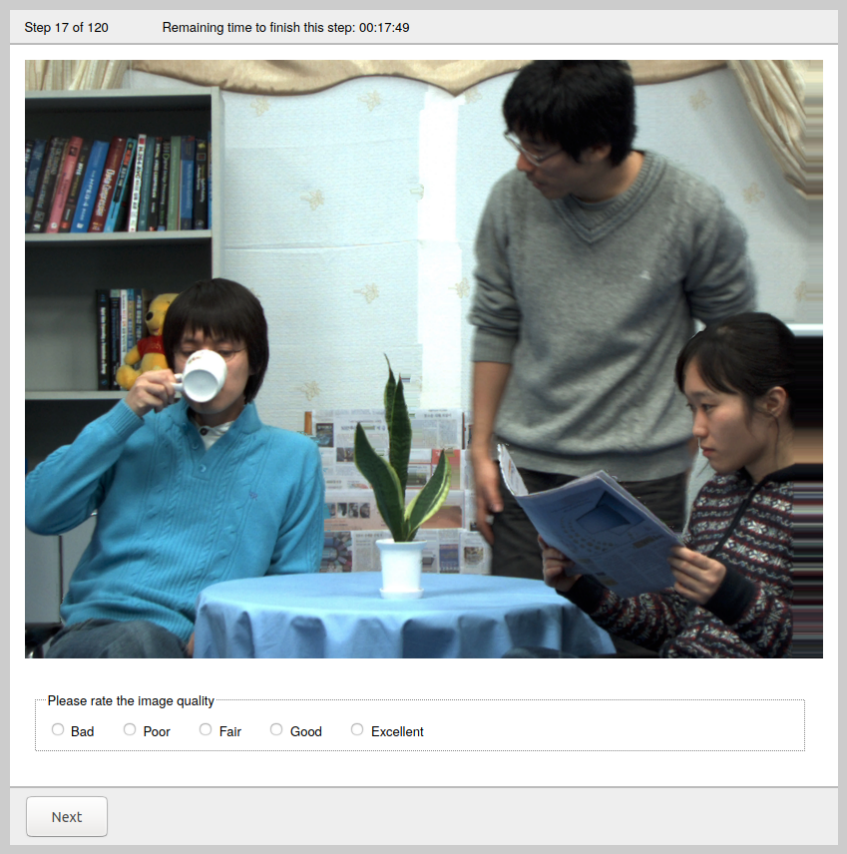


Figure 6.1.: This figure shows a screenshot of the QualityCrowd browser interface as seen by a subject which participates at the study. The synthesized view is shown and the subject is tasked to rate the image quality on a discrete five point ACR scale.

$\text{Newspaper}_{4\to5}$, $\text{Newspaper}_{4\to6}$, $\text{Newspaper}_{6\to4}$, $\text{Newspaper}_{6\to5}$, $\text{Book}_{8\to10}$, $\text{Book}_{10\to8}$, $\text{Book}_{8\to9}$, and $\text{Book}_{10\to9}$.

They were using 7 different algorithms on these three sequences:

A1: Holes

These are the synthesized views where the disoclussions are left present in the form of black holes.

A2: Fehn et al. [10]

This method applies a Gaussian low-pass filter to the depth image which smoothes out depth-discontinuities which manifest as edges with high contrast in the depth

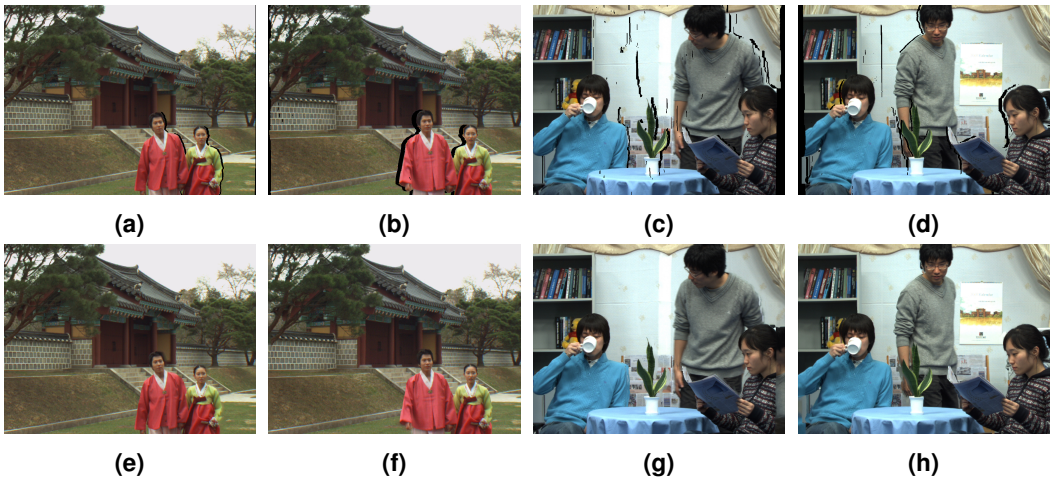


Figure 6.2.: The test set consists of 120 images, comprising 3 different test sequences, each with 4 different viewpoints and 10 different view synthesis methods. In this figure, I've selected Frame No. 104 of the *Lovebird* sequence transformed from Camera 6 to Camera 7; Frame No. 112 transformed from Camera 8 to Camera 6; Frame No. 136 of the *Newspaper* sequence transformed from Camera 4 to Camera 5 and Frame No. 104 transformed from Camera 6 to Camera 5. The top row (images (a) to (d) shows the transformed viewpoint with the disocclusions, the bottom row the filling result of my VS-CSH algorithm (images (e) to (h)).

image. Thereby, the algorithm eliminates the depth discontinuities, similar to the method by Zhang et al. [11]. After warping the image, these smooth depth gradients lead to visible distortions of the image. Also, this technique can't be applied to disocclusions at the border of the image, which is why these disocclusions are simply cropped away.

A3: Fehn et al. [10] with Telea [32] inpainting

This algorithm uses the same method as A2 but the disocclusions at the border are inpainted using the method proposed by Telea [32].

A4: MPEG VSRS [123]

This is the view synthesis method used in VSRS proposed by Tanimoto, Mori and others [123, 25] which also uses Telea [32] as the inpainting step.

A5: Müller et al. [36]

This view synthesis method uses a hole-filling algorithm which fills the holes line-wise. Each horizontal line of the hole is filled with the color of the nearest background pixel which is horizontally extrapolated into the hole. They claim that "this simple constant-color extrapolation of the background pixel leads to better results,

than an unconstrained linear interpolation between both values" [36]. However, this method obviously fails to reproduce texture or any kind of linear structure which is not horizontal.

A6: Ndjiki-Nya et al. [41]

This is one of the variants using Criminisi's [63] algorithm in the inpainting step, not dissimilar to the algorithms developed by Daribo [38] or Gautier [40] which have been used in the previous chapter for the objective testing.

A7: Köppel et al. [134]

Köppel et al. [134] extend the algorithm of Ndjiki-Nya et al. by saving a sprite of the background. Therefore, background information which was present in earlier frames can be used for the inpainting step of the holes. However, the information stored in the background sprite might become outdated quickly, which would degrade the synthesized view.

A8: Original

These are the original camera views which have not been synthesized.

With the combination of these three sequences with four different camera views and eight algorithms, they've ended up with a total of 96 images which were rated in this study.

6.2.3. Laboratory Study

Bosc et al. conducted their study in an "ITU conforming test environment" [103], where they presented the stimuli to 43 naïve observers on a TVLogic LVM401W LCD reference monitor in a test setup in conformance to the recommendations of ITU-R BT.500 [135]. They've conducted two separate studies, one with an absolute category rating (ACR) and another with a pairwise comparison (PC) study. I will only use the results of the ACR study in my comparison because the PC study takes much longer to complete (40 min vs. 20 min) and therefore isn't very well suited for an online experiment. The age and gender distribution of the test subjects unfortunately is not known to me. They conducted pre-screening of the subjects with Snellen and Ishihara charts to check for visual acuity and color blindness and post-screening according to the rejection criteria formulated by the Visual Quality Experts Group (VQEG) [136], however did not have to exclude any

participant. Bosc et al. unfortunately didn't describe any further details about the study, especially not how the participants were instructed and whether they were using anchor sequences in a stabilization phase.

6.2.4. QualityCrowd Study

My goal with the QualityCrowd study was to replicate the experiment conducted in [103] as closely as possible in a web-based setting with the addition of the synthesized images of the two algorithms presented in this thesis. Fortunately, Bosc et al. published their dataset online[2] so that I could download all the images they used in their study. I've then applied my view synthesis algorithms to the same frames and the same combination of camera pairs of the *Lovebird* and *Newspaper* sequence which were used in the data set, so that I had 112 images in total.

It lies in the nature of the tests which are conducted using QualityCrowd that they violate many of the recommendations which are formulated in ITU-R BT.500 [135] concerning the study environment, the presentation device, the distance between observer and screen and so forth because one has largely no control over how the images are presented at the site of the test subject. I did, however, ask all participants to conduct the study on a web browser on a *computer* and explicitly asked them to not use their mobile devices because of the limited screen size of mobile phones which might be too small for the artifacts of the algorithms to be sufficiently well resolved.

The participants were invited via e-mail where each participant got an individual link to the QualityCrowd test site. The study group consisted mostly of friends and family of mine and several colleagues, however, I've paid attention to only invite non-experts which were not aware of the nature of the algorithms. The only instruction about the purpose of the study I've given was to compare "the visual quality of several image processing algorithms". In total, 24 observers (12 female, 12 male) participated in the study, which is the recommended number of test subjects in the VQEG test plan [136], with a range of 34 to 60 years of age.

In the written instructions which were presented to the test subjects on the QualityCrowd

[2] http://ivc.univ-nantes.fr/en/databases/DIBR_Images/

platform, I've explained how the test will be conducted and asked the participants to complete the test on their own. QualityCrowd ensures that each participant can conduct the test only once with their individual link and that the subjects can't jump back or repeat a test step.

The first step of the test was a training session. In this training session, I've presented two images of the study consecutively, one of the group A1 with the holes and one of the original views of the A8 group and explained the rating scale. I've chosen these two images to serve as anchor images for the subjects to adjust to the scale. The ratings of the training session were discarded.

In the subsequent test, the 112 images were shuffled in random order, however, I did include a stabilization phase at the very beginning of the test which consisted of 6 images with the complete range of MOS values known from the lab study. The purpose of this stabilization phase again was to give the test subjects the opportunity to adjust their inner rating scale. The results of the stabilization phase were discarded as well and the 6 images repeated later on in the actual test. More information about the purpose and the effect of the stabilization phase in a subjective test can be found in [137].

The test was conducted in January of 2020. All 24 participants completed the test. As the setting of an online test doesn't allow a meaningful pre-experiment screening of the observers, I've only applied post-experiment screening for possible rejections of participants. I followed the recommendations of VQEG [136] and ITU-R BT.500 [135]. The VQEG recommendations suggest to eliminate the ratings of a participant if its Pearson correlation coefficient per sequence is lower than 0.75 vs. the rest of the viewers. The ITU recommendations define a more involved rejection criterion which counts how often a test subject's ratings exceed an interval defined by the mean and the standard deviation of all ratings. In the end, I did not have to exclude any participants from the study.

6.3. Comparison of QualityCrowd with the Laboratory Study

After conducting the study in the QualityCrowd platform, first of all I want to compare the results of both studies and find out whether there are any significant differences in the studies which might stem from the different methodologies used to get the MOS values.

Figure 6.3 shows a scatter plot of the results of the studies. One can see a high correlation

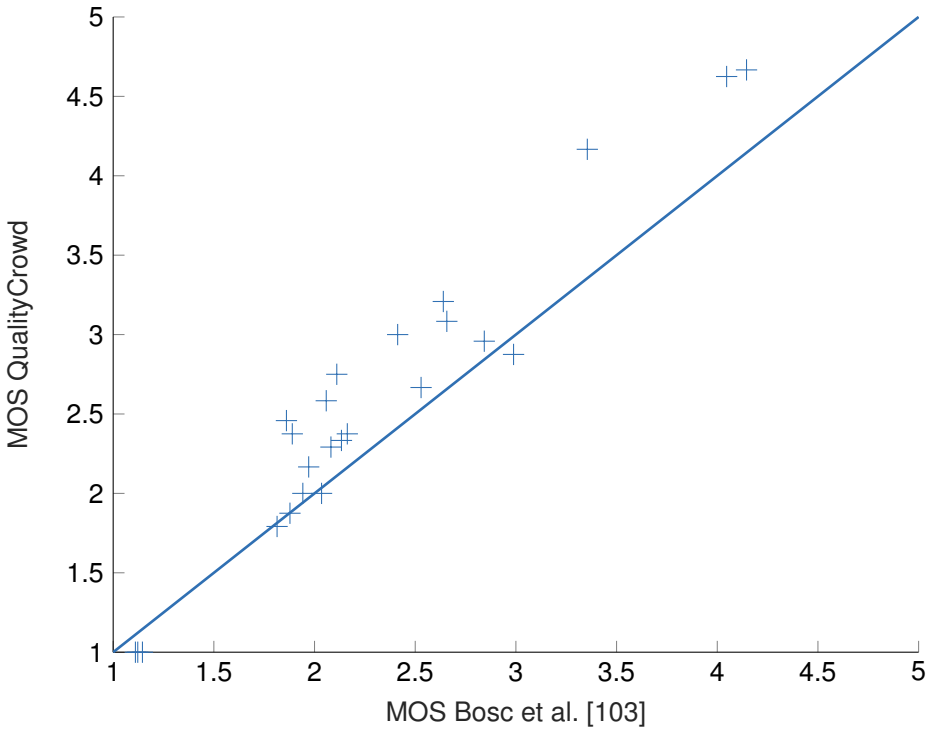


Figure 6.3.: Scatter plot comparing the MOS values of the lab experiment of [103] and my study conducted with QualityCrowd. The MOS of the QualityCrowd study has a tendency to be slightly better, which is probably a result of the test subjects' better utilization of the full rating scale in the online test.

between the results of both studies with a Pearson correlation coefficient of $\rho_P = 0.967$ and a Spearman rank correlation of $\rho_S = 0.913$. For reference, this is considerably better than the lowest acceptable correlation between the results of studies conducted in different labs (the so called lab-to-lab correlation) using the same methodology, which was defined by the VQEG as $\rho_P \geq 0.94$.

However, correlation alone is not a good indicator for the agreement between two different measurement methodologies [138]. I've therefore created a Bland-Altman plot of the results, which allows to have a more detailed look at the differences between both methodologies. A Bland-Altman plot [138] is a tool often used in the field of medicine or analytic chemistry for the comparison of two methods of measurement. It plots the differences between the two sets $S_1 - S_2$ over the average of both results $\frac{S_1+S_2}{2}$, including three horizontal lines which show the mean value and the 95% confidence intervals. One can see

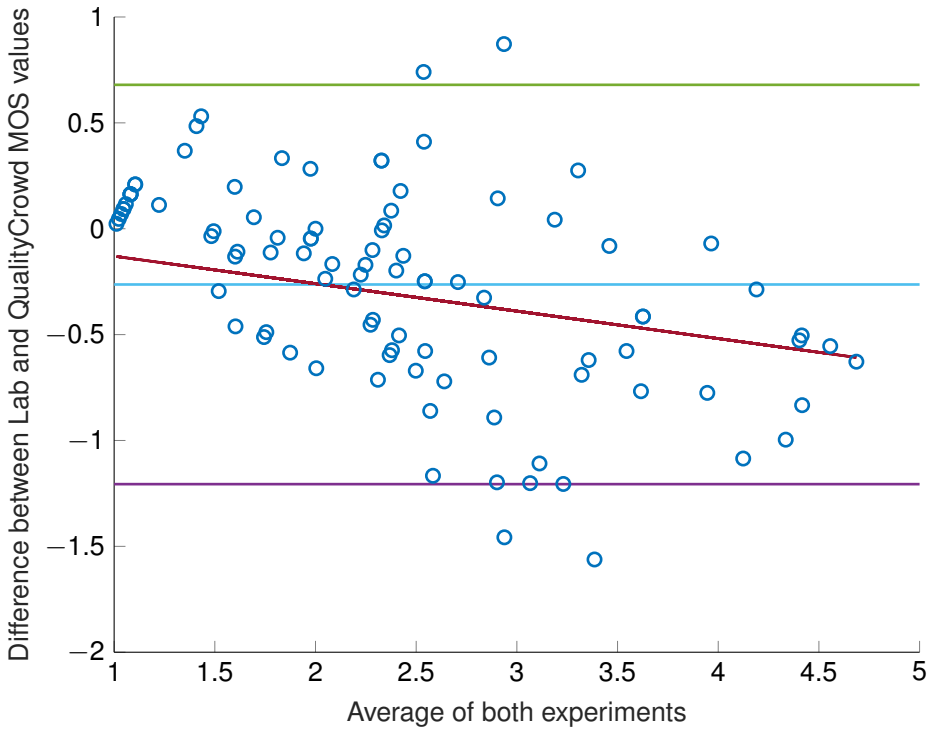


Figure 6.4.: The Bland-Altman plot of the MOS values of the lab experiment of [103] and my study conducted with QualityCrowd shows the difference between both measurement methods. Most measurements lie within the 95% confidence interval shown with the two horizontal lines. There is a mean offset of -0.26, which is not quite statistically significant (p=0.065). There seems to be a slight systematic bias towards better MOS values in the QualityCrowd study, where the test subjects seem to rate high quality pictures better in the QualityCrowd study than in the Lab experiment.

that there is a slight difference in the mean MOS of both studies of -0.26 which is not statistically significant (p=0.065). Also, one can see that there might be a a slight systematic bias towards better MOS values in the QualityCrowd study where the test subjects seem to rate high quality pictures better in the QualityCrowd study than in the Lab experiment, which I've shown with the regression line that I've added. It looks like the test subjects in the QualityCrowd study seem to rate images with a good MOS better than in the lab study, while for low MOS values both studies seem to be in relative agreement. I suspect this to be a result of the direct and indirect anchoring [135] in the training session and the hidden stabilization phase of the QualityCrowd study, which probably led the test subjects to make better utilization of the full ACR scale.

Apart from these slight differences, almost all of the ratings lie well within the 95% confidence intervals, which means that there is no statistically significant difference between both studies.

This result shows that the subjective image quality test which I've conducted on QualityCrowd has delivered valid results for DIBR images, comparable to those that one would get from a laboratory study. Having validated my methodology, we can now have a closer look at the performance of the algorithms in the study and of course especially have a look at the performance of both view synthesis algorithms presented in this thesis.

Figure 6.5 shows the average MOS values for the *Lovebird* and the *Newspaper* sequence for all 10 different algorithms which were evaluated in the study. For the eight algorithms which were evaluated in both studies I've plotted both results, whereas the VS-BP and the VS-CSH algorithms have only been evaluated in the QualityCrowd experiment. One can see that, unsurprisingly, the images with the disocclusions present as black holes performed worst in the study, while, also not suprising, the original camera views performed best. All the view synthesis algorithms are ranked somewhere in the middle between these two extremes. The ranking of the algorithms is well preserved between both studies; the QualityCrowd study has ranked the algorithms in the same order as the lab study – with one notable exception which is the Fehn algorithm with Telea inpainting (A3). There, the lab study has a lower MOS than the QC study. I had a closer look at this discrepancy and, unfortunately, there seems to be an error in the raw data provided by Bosc et al. of the study. While in their paper they computed an average MOS of 2.41 for this algorithm (which is probably the correct value), if one calculates the MOS from the raw data of the study one gets a MOS of only 2.157. The MOS values for all the other algorithms, however, seem to be correct and plausible. Still, I find it to be a nice example of the expressiveness of the QualityCrowd study that it was able to uncover this error in the data of the original study.

6.4. Subjective Visual Quality of VS-BP and VS-CSH

The results of the QualityCrowd study show that the VS-CSH algorithm is rated to be on par or have a slightly better visual quality than the next best algorithm of the study

"Fehn cropped" with a MOS of 2.79 vs 2.73 (this difference is not statistically significant with $p = 0.587$). This is particularly notable because in the "Fehn cropped" algorithm the disocclusions at the border of the image are actually cropped away which should give this algorithm a considerable advantage over one which is filling in the disocclusions (see Chapter 6.2.2 for details). VS-CSH is significantly better than the third best algorithm in this study, which is the one of Müller et al. [36] which is using an inpainting technique to fill in the disocclusions (MOS 2.79 vs 2.48, $p < 0.007$). VS-BP did not perform quite as well as the objective metrics suggested, and it is also my personal subjective impression that VS-CSH performs better on the selected test set than VS-BP. With a mean opinion score of 2.44, it is on par with the visual quality of the algorithms by Müller et al. [36] (MOS 2.47) and Ndjiki-Nya et al. [41] (MOS 2.52). The differences in visual quality are not statistically significant. VS-BP is, however, significantly better than the hole-filling algorithm in the VSRS by Mori et al. [25] (MOS 2.06, $p < 0.05$).

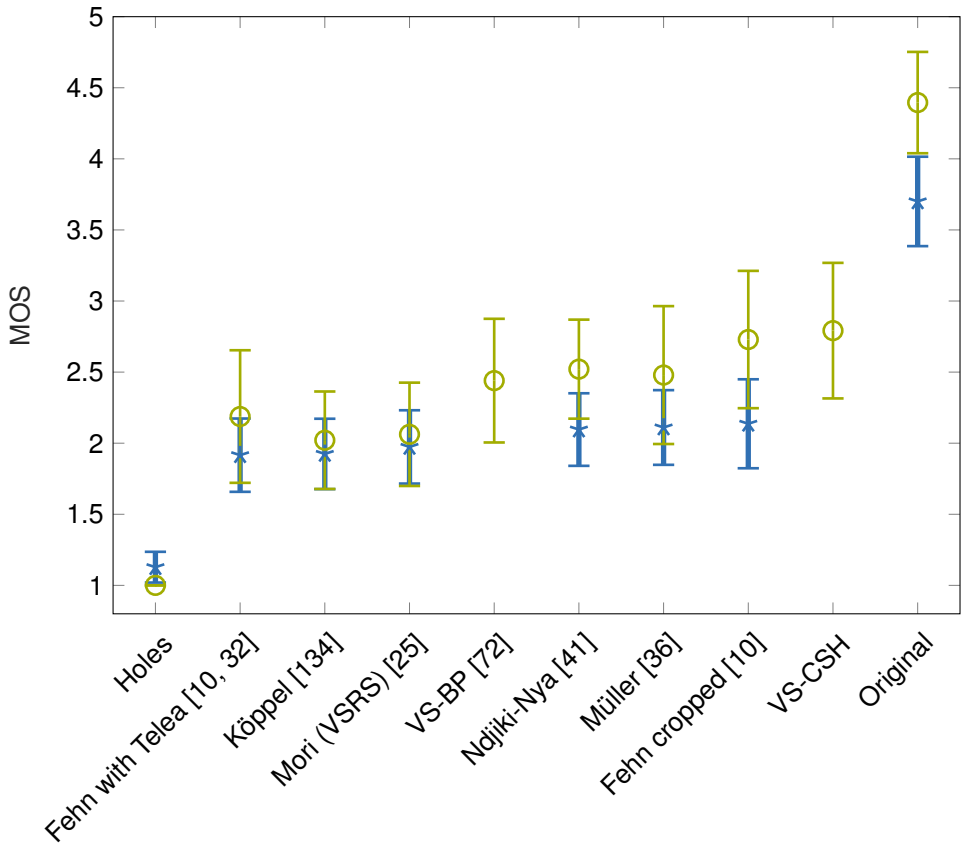


Figure 6.5.: This plot shows the average MOS and the 95% confidence intervals for the *Lovebird* and the *Newspaper* sequence for all 10 different algorithms. The blue plot shows the results of the laboratory study of [103] while the green plot shows the results of the QualityCrowd study. The algorithms have been sorted by their MOS of the laboratory study in ascending order, with the images with holes ranking worst in the experiment and the ground truth camera images ranking best. VS-BP and VS-CSH have only one data point each because they were not part of the laboratory study. Out of the view synthesis algorithms in this subjective study, VS-CSH scored the highest MOS with 2.8.

7. Conclusion

This thesis presented two new hole-filling algorithms for view synthesis. VS-BP applies a global optimization approach on Markov random fields incorporating the information of the depth map. This not only leads to consistent inpainting results but also to a higher algorithmic efficiency due to rigorous label pruning based on depth range. It has a significant improvement in objective image quality even when compared to very recently published state-of-the-art algorithms, and compares very favorably in the subjective study.

VS-CSH was developed with the goal to further improve the computational efficiency of hole-filling algorithms. By employing a state-of-the-art hashing scheme, I've achieved a tremendous increase in processing speed by several orders of magnitude while still retaining excellent image quality. Considering the objective metrics, it performs almost as well as VS-BP, however, the results of the subjective study rank it as the algorithm with the highest visual quality of the study.

In the quest for evaluating the visual quality of both algorithms, I've conducted a crowdsourced subjective study, which to my knowledge, has never been done before for DIBR images. I've therefore validated the results of this study against an independent laboratory study and shown that it is a valid method to conduct subjective experiments with significantly less effort than laboratory studies.

In this thesis, I've developed algorithms which operate on individual images. Their performance on videos could most probably be improved even further by taking into account the temporal correlation of consecutive frames. The literature knows numerous approaches on how to do that and I've made specific suggestions for both algorithms which were presented here. For VS-BP, this has actually been done by Kim et al. [80] while I was preparing this thesis with at least subjectively good looking results; and a similar approach could also be applied to VS-CSH.

Bibliography

1. J. Carey, “Media technology adoption,” in *Mediated Communication*, P. M. Napoli, Ed. Berlin, Germany; Boston, MA: De Gruyter, 2018, pp. 71–90.

2. B. F. Coll, F. Ishtiaq, and K. O. Connell, “3DTV at home: status, challenges and solutions for delivering a high quality experience,” in *Proceedings of the Fifth International Workshop on Video Processing and Quality Metrics for Consumer Electronics*, Scottsdale, AZ, 2010, pp. 1–6.

3. R. T. Held and M. S. Banks, “Misperceptions in stereoscopic displays: A vision science perspective,” in *Proceedings of the 5th Symposium on Applied Perception in Graphics and Visualization*, Los Angeles, CA, 2008, pp. 23–32.

4. J. Konrad, “Enhancement of viewer comfort in stereoscopic viewing: parallax adjustment,” in *Proceedings of the IS&T/SPIE Sympsosium on Electronic Imaging, Stereoscopic Displays and Virtual Reality Systems*, San Jose, CA, 1999, pp. 1–12.

5. K. Müller, P. Merkle, and T. Wiegand, “3-D video representation using depth maps,” *Proceedings of the IEEE*, vol. 99, no. 4, pp. 643–656, Apr. 2011.

6. H. Urey, K. V. Chellappan, E. Erden, and P. Surman, “State of the art in stereoscopic and autostereoscopic displays,” *Proceedings of the IEEE*, vol. 99, no. 4, pp. 540–555, Apr. 2011.

7. N. Ozbek, A. M. Tekalp, and E. T. Tunali, “A new scalable multi-view video coding configuration for robust selective streaming of free-viewpoint TV,” in *Proceedings of the IEEE International Conference on Multimedia and Expo*, Beijing, China, 2007, pp. 1155–1158.

8. S. E. Chen and L. Williams, “View interpolation for image synthesis,” in *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, New York, NY, 1993, pp. 279–288.

9. A. Redert, M. O. De Beeck, C. Fehn, W. IJsselsteijn, M. Pollefeys, L. Van Gool, E. Ofek, I. Sexton, and P. Surman, “Advanced three-dimensional television system technologies,” in *Proceedings of the 1st International Symposium on 3D Data Processing Visualization and Transmission*, Padova, Italy, 2002, pp. 313–319.

10. C. Fehn, “Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV,” in *Proceedings of SPIE Stereoscopic Displays and Virtual Reality Systems XI*, vol. 5291, San Jose, CA, 2004, pp. 93–104.

11. L. Zhang and W. Tam, “Stereoscopic image generation based on depth images for 3D TV,” *IEEE Transactions on Broadcasting*, vol. 51, no. 2, pp. 191–199, Jun. 2005.

12. I. Daribo and H. Saito, “Bilateral depth-discontinuity filter for novel view synthesis,” in *Proceedings of the IEEE International Workshop on Multimedia Signal Processing*, Saint Malo, France, 2010, pp. 145–149.

13. H.-Y. Shum and S. B. Kang, “A review of image-based rendering techniques,” in *Proceedings of SPIE Visual Communications and Image Processing*, vol. 4067, Perth, Australia, 2000, pp. 2–13.

14. W. R. Mark, “Post-rendering 3D image warping: Visibility, reconstruction, and performance for depth-image warping,” Ph.D. dissertation, University of North Carolina at Chapel Hill, Chapel Hill, NC, 1999.

15. O. Faugeras, *Three-Dimensional Computer Vision: A Geometric Viewpoint*. Cambridge, MA: MIT Press, 1993, vol. 38.

16. D. Scharstein and R. Szeliski, “A taxonomy and evaluation of dense two-frame stereo correspondence algorithms,” *International Journal of Computer Vision*, vol. 47, no. 1-3, pp. 7–42, Apr. 2002.

17. M. Tanimoto, T. Fujii, and K. Suzuki, “Reference software of depth estimation and view synthesis for FTV/3DV,” ISO/IEC JTC1/SC29/WG11 MPEG, M15836, Lausanne, Switzerland, Tech. Rep., 2008.

18. C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, “High-quality video view interpolation using a layered representation,” *ACM Transactions on Graphics*, vol. 1, no. 212, pp. 600–608, Aug. 2004.

19. B. Huhle, S. Fleck, and A. Schilling, “Integrating 3D time-of-flight camera data and high resolution images for 3DTV applications,” in *Proceedings of the 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video*, Kos Island, Greece, 2007, pp. 3–6.

20. M. Schmidt and B. Jähne, “A physical model of time-of-flight 3D imaging systems, including suppression of ambient light,” in *Dynamic 3D Imaging. Lecture Notes in Computer Science*, A. Kolb and R. Koch, Eds. Berlin, Heidelberg, Germany: Springer, 2009, vol. 5742, pp. 1–15.

21. H. G. Jeon, J. Park, G. Choe, J. Park, Y. Bok, Y. W. Tai, and I. S. Kweon, “Accurate depth map estimation from a lenslet light field camera,” in *Proceedings of the 28th IEEE Conference on Computer Vision and Pattern Recognition*, Boston, MA, 2015, pp. 1547–1555.

22. Y. Anisimov, O. Wasenmüller, and D. Stricker, “A compact light field camera for real-time depth estimation,” in *Proceedings of the 18th International Conference on Computer Analysis of Images and Patterns*, vol. 11678, Salerno, Italy, 2019, pp. 52–63.

23. A. Chalmers and K. Debattista, “HDR video past, present and future: A perspective,” *Signal Processing: Image Communication*, vol. 54, pp. 49–55, May 2017.

24. F. Dufaux, P. Le Callet, R. K. Mantiuk, and M. Mrak, Eds., *High Dynamic Range Video: From Acquisition to Display and Applications*. London, UK: Academic Press, 2016.

25. Y. Mori, N. Fukushima, T. Yendo, T. Fujii, and M. Tanimoto, “View generation with 3D warping using depth information for FTV,” *Signal Processing: Image Communication*, vol. 24, no. 1-2, pp. 65–72, Jan. 2009.

26. C. Tomasi and R. Manduchi, “Bilateral filtering for gray and color images,” in *Proceedings of the IEEE International Conference on Computer Vision*, Bombay, India, 1998, pp. 839–846.

27. J. Neundorf, V. Schmidt, T. Lagemann, and F. Hofmeyer, “Automatic color matching between stereo images,” in *Proceedings of the IEEE International Conference on Consumer Electronics*, Berlin, Germany, 2012, pp. 185–189.

28. D. Gadia, D. Villa, C. Bonanomi, A. Rizzi, and D. Marini, “Local color correction of stereo pairs,” in *Proceedings of SPIE Stereoscopic Displays and Applications XXI*, vol. 7524, San Jose, CA, 2010, p. 75240W.

29. W. J. Tam, G. Alain, L. Zhang, T. Martin, and R. Renaud, “Smoothing depth maps for improved steroscopic image quality,” in *Proceedings of SPIE Three-Dimensional TV, Video, and Display III*, vol. 5599, Philadelphia, PA, 2004, pp. 1–11.

30. W. Y. Chen, Y. L. Chang, S. F. Lin, L. F. Ding, and L. G. Chen, “Efficient depth image based rendering with edge dependent depth filter and interpolation,” in *Proceedings of the IEEE International Conference on Multimedia and Expo*, Amsterdam, Netherlands, 2005, pp. 1314–1317.

31. Y.-R. Horng, Y.-C. Tseng, and T.-S. Chang, “Stereoscopic images generation with directional gaussian filter,” in *Proceedings of the IEEE International Symposium on Circuits and Systems*, Paris, France, 2010, pp. 2650–2653.

32. A. Telea, “An image inpainting technique based on the fast marching method,” *Journal of Graphics Tools*, vol. 9, no. 1, pp. 25–36, 2004.

33. P. Li, Y. Jin, B. Sheng, D. Lin, Y. Nie, and E. Wu, “Multiview-coherent disocclusion synthesis using connected regions optimization,” *Computer Animation and Virtual Worlds*, vol. 30, no. 3-4, p. e1894, May 2019.

34. P. Kauff, N. Atzpadin, C. Fehn, M. Müller, O. Schreer, A. Smolic, and R. Tanger, “Depth map creation and image-based rendering for advanced 3DTV services providing interoperability and scalability,” *Signal Processing: Image Communication*, vol. 22, no. 2, pp. 217–234, Feb. 2007.

35. S. Zinger, L. Do, and P. H. De With, “Free-viewpoint depth image based rendering,” *Journal of Visual Communication and Image Representation*, vol. 21, no. 5-6, pp. 533–541, Jul. 2010.

36. K. Müller, A. Smolic, K. Dix, P. Merkle, P. Kauff, and T. Wiegand, “View synthesis for advanced 3D video systems,” *EURASIP Journal on Image and Video Processing*, pp. 1–11, Feb. 2009.

37. A. Criminisi, P. Pérez, K. Toyama, P. Perez, and K. Toyama, “Region filling and object removal by exemplar-based image inpainting,” *IEEE Transactions on Image Processing*, vol. 13, no. 9, pp. 1200–1212, Sep. 2004.

38. I. Daribo and H. Saito, “A novel inpainting-based layered depth video for 3DTV,” *IEEE Transactions on Broadcasting*, vol. 57, no. 2, pp. 533–541, Jun. 2011.

39. K.-J. Oh, S. Yea, and Y.-S. Ho, “Hole filling method using depth based in-painting for view synthesis in free viewpoint television and 3-D video,” in *Proceedings of the Picture Coding Symposium*, Chicago, IL, 2009, pp. 1–4.

40. J. Gautier, O. Le Meur, and C. Guillemot, “Depth-based image completion for view synthesis,” in *Proceedings of the 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video*, Antalya, Turkey, 2011, pp. 1–4.

41. P. Ndjiki-Nya, M. Köppel, D. Doshkov, H. Lakshman, P. Merkle, K. Müller, and T. Wiegand, “Depth image based rendering with advanced texture synthesis,” in *Proceedings of the IEEE International Conference on Multimedia and Expo*, Suntec City, Singapore, 2010, pp. 424–429.

42. I. Ahn and C. Kim, “A novel depth-based virtual view synthesis method for free viewpoint video,” *IEEE Transactions on Broadcasting*, vol. 59, no. 4, pp. 614–626, Dec. 2013.

43. P. Buyssens, O. L. Meur, M. Daisy, D. Tschumperle, and O. Lezoray, “Depth-guided disocclusion inpainting of synthesized RGB-D images,” *IEEE Transactions on Image Processing*, vol. 26, no. 2, pp. 525–538, Oct. 2017.

44. C. Zhu and S. Li, “Depth image based view synthesis: New insights and perspectives on hole generation and filling,” *IEEE Transactions on Broadcasting*, vol. 62, no. 1, pp. 82–93, Sep. 2016.

45. N. Komodakis and G. Tziritas, “Image completion using efficient belief propagation via priority scheduling and dynamic pruning,” *IEEE Transactions on Image Processing*, vol. 16, no. 11, pp. 2649–2661, Oct. 2007.

46. M. Schmeing and X. Jiang, “Faithful disocclusion filling in depth image based rendering using superpixel-based inpainting,” *IEEE Transactions on Multimedia*, vol. 17, no. 12, pp. 2160–2173, Sep. 2015.

47. C. M. Cheng, S. J. Lin, and S. H. Lai, “Spatio-temporally consistent novel view synthesis algorithm from video-plus-depth sequences for autostereoscopic displays,” *IEEE Transactions on Broadcasting*, vol. 57, no. 2, pp. 523–532, Jun. 2011.

48. R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, “Slic superpixels compared to state-of-the-art superpixel methods,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274–2281, Nov. 2012.

49. H. A. Hsu, C. K. Chiang, and S. H. Lai, “Spatio-temporally consistent view synthesis from video-plus-depth data with global optimization,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 1, pp. 74–84, Jan. 2014.

50. H. G. Kim and Y. M. Ro, “A new hole filling method based on 3d geometric transformation for synthesized image,” in *Proceedings of the IS&T International Symposium on Electronic Imaging Science and Technology*, San Francisco, CA, 2016, pp. 1–5.

51. T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely, “Stereo magnification: Learning view synthesis using multiplane images,” *ACM Transactions on Graphics*, vol. 37, no. 4, Aug. 2018.

52. J. Flynn, I. Neulander, J. Philbin, and N. Snavely, “Deep stereo: Learning to predict new views from the world’s imagery,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, 2016, pp. 5515–5524.

53. J. Flynn, M. Broxton, P. Debevec, M. DuVall, G. Fyffe, R. Overbeck, N. Snavely, and R. Tucker, “Deepview: View synthesis with learned gradient descent,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Long Beach, CA, 2019, pp. 2367–2376.

54. S. M. Ali Eslami, D. J. Rezende, F. Besse, F. Viola, A. S. Morcos, M. Garnelo, A. Ruderman, A. A. Rusu, I. Danihelka, K. Gregor, D. P. Reichert, L. Buesing, T. Weber,

O. Vinyals, D. Rosenbaum, N. Rabinowitz, H. King, C. Hillier, M. Botvinick, D. Wierstra, K. Kavukcuoglu, and D. Hassabis, "Neural scene representation and rendering," *Science*, vol. 360, no. 6394, pp. 1204–1210, Jun. 2018.

55. N. K. Kalantari, T. C. Wang, and R. Ramamoorthi, "Learning-based view synthesis for light field cameras," *ACM Transactions on Graphics*, vol. 35, no. 6, pp. 1–10, Nov. 2016.

56. B. S. Lipkin and A. Rosenfeld, *Picture processing and psychopictorics.* Orlando, FL: Academic Press, 1970.

57. A. K. Jain, "Partial differential equations and finite-difference methods in image processing, part 1: Image representation," *Journal of Optimization Theory and Applications*, vol. 23, no. 1, pp. 65–91, Sep. 1977.

58. R. G. Schayes, P. Homan, and U. Rothgordt, "Method and arrangement for correcting errors in facsimile transmission," U.S. Patent 4 188 643, Feb. 12, 1980.

59. M. B. Sachs, J. Nachmias, and J. G. Robson, "Spatial-frequency channels in human vision," *Journal of the Optical Society of America*, vol. 61, no. 9, pp. 1176 –1186, Sep. 1971.

60. Z. Wang, A. C. Bovik, H. R. Sheikh, and E. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

61. A. Kaup and T. Aach, "Efficient prediction of uncovered background in interframe coding using spatial extrapolation," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Adelaide, Australia, 1994, pp. 501–504.

62. M. Bertalmio, A. L. Bertozzi, and G. Sapiro, "Navier-stokes, fluid dynamics, and image and video inpainting," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Kauai, HI, 2001, pp. 355–362.

63. A. Criminisi, P. Perez, and K. Toyama, "Object removal by exemplar-based inpainting," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, Madison, WI, 2003, pp. 721–728.

64. A. A. Efros and T. K. Leung, “Texture synthesis by non-parametric sampling,” in *Proceedings of the 7th IEEE International Conference on Computer Vision*, Kerkyra, Greece, 1999, pp. 1033–1038.

65. S. C. Zhu, Y. Wu, and D. Mumford, “Filters, random fields and maximum entropy (frame): Towards a unified theory for texture modeling,” *International Journal of Computer Vision*, vol. 27, no. 2, pp. 107–126, Mar. 1998.

66. C. Shannon, “A mathematical theory of communication (part I),” *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, Jul. 1948.

67. A. A. Efros and W. T. Freeman, “Image quilting for texture synthesis and transfer,” in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, Los Angeles, CA, 2001, pp. 341–346.

68. A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin, “Image analogies,” in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, Los Angeles, CA, 2001, pp. 327–340.

69. M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, “Image inpainting,” in *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, New York, NY, 2000, pp. 417–424.

70. C. M. Cheng, S. J. Lin, S. H. Lai, and J. C. Yang, “Improved novel view synthesis from depth image with large baseline,” in *Proceedings of the 19th International Conference on Pattern Recognition*, Tampa, FL, 2008, pp. 1–4.

71. S. Di Zenzo, “A note on the gradient of a multi-image,” *Computer Vision, Graphics and Image Processing*, vol. 33, no. 1, pp. 116–125, Jan. 1986.

72. J. Habigt and K. Diepold, “Image completion for view synthesis using Markov random fields and efficient belief propagation,” in *Proceedings of the IEEE International Conference on Image Processing*, Melbourne, Australia, 2013, pp. 2131–2134.

73. M. Tanimoto, T. Fujii, and K. Suzuki, “View synthesis algorithm in view synthesis reference software 2.0 (VSRS 2.0),” ISO/IEC JTC1/SC29/WG11 M16090, Lausanne, Switzerland, Tech. Rep., 2008.

74. J. M. Hammersley and P. Clifford, “Markov fields on finite graphs and lattices,” University of California, Berkeley, Tech. Rep., 1971.

75. S. E. Shimony, “Finding MAPs for belief networks is NP-hard,” *Artificial Intelligence*, vol. 68, no. 2, pp. 399–410, Aug. 1994.

76. J. Pearl, “Reverend bayes on inference engines: a distributed hierarchical approach.” in *Proceedings of the Second National Conference on Artificial Intelligence*, Pittsburgh, PA, 1982, pp. 133–136.

77. Y. Boykov, O. Veksler, and R. Zabih, “Fast approximate energy minimization via graph cuts,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 11, pp. 1222–1239, Nov. 2001.

78. Y. Weiss and W. T. Freeman, “On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs,” *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 736–744, Feb. 2001.

79. J. Sun, N. N. Zheng, and H. Y. Shum, “Stereo matching using belief propagation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 7, pp. 787–800, Jul. 2003.

80. H. G. Kim, S. S. Yoon, and Y. M. Ro, “Temporally consistent hole filling method based on global optimization with label propagation for 3D video,” in *Proceedings of the IEEE International Conference on Image Processing*, Québec City, Canada, 2015, pp. 3136–3140.

81. D. E. Knuth, *The Art of Computer Programming*, 2nd ed. Redwood City, CA: Addison Wesley, 1998, vol. 3.

82. J. L. Bentley and J. Louis, “Multidimensional binary search trees used for associative searching,” *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975.

83. K. He and J. Sun, “Computing nearest-neighbor fields via propagation-assisted KD-trees,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Providence, RI, 2012, pp. 111–118.

84. R. Bellman, *Dynamic programming*. Princeton, NJ: Princeton University Press, 1962.

85. S. Arya and D. Mount, “Approximate nearest neighbor queries in fixed dimensions,” in *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, Austin, TX, 1993, pp. 271–280.

86. P. Indyk and R. Motwani, “Approximate nearest neighbors: Towards removing the curse of dimensionality,” in *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, Dallas, TX, 1998, pp. 604–613.

87. M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, “Locality-sensitive hashing scheme based on p-stable distributions,” in *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, Brooklyn, NY, 2004, pp. 253–262.

88. C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, “Patchmatch: A randomized correspondence algorithm for structural image editing,” *ACM Transactions on Graphics*, vol. 28, no. 3, p. 24, Aug. 2009.

89. S. Lu, J. Hanca, A. Munteanu, and P. Schelkens, “Depth-based view synthesis using pixel-level image inpainting,” in *Proceedings of the 18th International Conference on Digital Signal Processing*, Fira, Greece, 2013, pp. 1–6.

90. S. P. Lu, B. Ceulemans, A. Munteanu, and P. Schelkens, “Performance optimizations for patchmatch-based pixel-level multiview inpainting,” in *Proceedings of the International Conference on 3D Imaging*, Liège, Belgium, 2013, pp. 1–7.

91. M. Ciotta and D. Androutsos, “Depth guided image completion for structure and texture synthesis,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Shanghai, China, 2016, pp. 1199–1203.

92. G. Luo, Y. Zhu, Z. Weng, and Z. Li, “A disocclusion inpainting framework for depth-based view synthesis,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 6, pp. 1289–1302, Jun. 2020.

93. S. Korman and S. Avidan, “Coherency sensitive hashing,” in *Proceedings of the IEEE International Conference on Computer Vision*, Barcelona, Spain, 2011, pp. 1607–1614.

94. K. R. Rao and N. Ahmed, “Orthogonal transforms for digital signal processing,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Philadelphia, PA, 1976, pp. 136–140.

95. Y. Hel-Or and H. Hel-Or, “Real-time pattern matching using projection kernels,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 9, pp. 1430–1445, Jul. 2005.

96. G. Ben-Artzi, H. Hel-Or, and Y. Hel-Or, “The gray-code filter kernels,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 3, pp. 382–393, Mar. 2007.

97. A. Fitzgibbon, Y. Wexler, and A. Zisserman, “Image-based rendering using image-based priors,” *International Journal of Computer Vision*, vol. 63, no. 2, pp. 141–151, Jul. 2005.

98. Y. Wexler, E. Shechtman, and M. Irani, “Space-time completion of video.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 3, pp. 463–76, Mar. 2007.

99. G.-M. Um, G. Ban, Y.-S. Ho, and A. Et, “Video test material of outdoor scene,” ISO/IEC JTC1/SC29/WG11 MPEG, M15371, Archamps, France, Tech. Rep., 2008.

100. C. Keimel, J. Habigt, M. Klimpke, and K. Diepold, “Design of no-reference video quality metrics with multiway partial least squares regression,” in *Proceedings of the 3rd International Workshop on Quality of Multimedia Experience*, Mechelen, Belgium, 2011, pp. 49–54.

101. C. Keimel, J. Habigt, and K. Diepold, “Hybrid no-reference video quality metric based on multiway PLSR,” in *Proceedings of the European Signal Processing Conference*, Bucharest, Romania, 2012, pp. 1244–1248.

102. C. Keimel, M. Klimpke, J. Habigt, and K. Diepold, “No-reference video quality metric for HDTV based on H.264/AVC bitstream features,” in *Proceedings of the IEEE International Conference on Image Processing*, Brussels, Belgium, 2011, pp. 3325–3328.

103. E. Bosc, R. Péepion, P. Le Callet, M. Köppel, P. Ndjiki-Nya, M. Pressigout, and L. Morin, “Towards a new quality metric for 3-D synthesized view assessment,” *IEEE Journal on Selected Topics in Signal Processing*, vol. 5, no. 7, pp. 1332–1343, Sep. 2011.

104. ITU-T Recommendation P.910, “Subjective video quality assessment methods for multimedia applications,” International Telecommunication Union, Geneva, Switzerland, Tech. Rep., 2009.

105. Z. Wang, E. P. Simoncelli, and A. C. Bovik, “Multi-scale structural similarity for image quality assessment,” *Proceedings of the IEEE Asilomar Conference on Signals, Systems and Computers*, vol. 2, pp. 1398–1402, 2003.

106. D. M. Chandler and S. S. Hemami, “VSNR : A wavelet-based visual signal-to-noise ratio for natural images,” *IEEE Transactions on Image Processing*, vol. 16, no. 9, pp. 2284–2298, Sep. 2007.

107. H. R. Sheikh and A. C. Bovik, “Image information and visual quality,” *IEEE Transactions on Image Processing*, vol. 15, no. 2, pp. 430–444, Feb. 2006.

108. Z. Wang and A. C. Bovik, “A universal image quality index,” *IEEE Signal Processing Letters*, vol. 9, no. 3, pp. 81–84, Mar. 2002.

109. H. R. Sheikh, A. C. Bovik, and G. de Veciana, “An information fidelity criterion for image quality assessment using natural scene statistics,” *IEEE Transactions on Image Processing*, vol. 14, no. 12, pp. 2117–2128, Dec. 2005.

110. N. Damera-Venkata, T. D. Kite, W. S. Geisler, B. L. Evans, and A. C. Bovik, “Image quality assessment based on a degradation model,” *IEEE Transactions on Image Processing*, vol. 9, no. 4, pp. 636–650, Apr. 2000.

111. J. L. Mannos and D. J. Sakrison, “The effects of a visual fidelity criterion on the encoding of images,” *IEEE Transactions on Information Theory*, vol. 20, no. 4, pp. 525–536, Jul. 1974.

112. N. Ponomarenko, F. Silvestri, K. Egiazarian, M. Carli, J. Astola, and V. Lukin, “On between-coefficient contrast masking of DCT basis functions,” in *Proceedings of the Third International Workshop on Video Processing and Quality Metrics for Consumer Electronics*, Scottsdale, AZ, 2007, pp. 1–4.

113. K. Egiazarian, J. Astola, N. Ponomarenko, V. Lukin, F. Battisti, and M. Carli, “A new full-reference quality metrics based on HVS,” in *Proceedings of the Second International Workshop on Video Processing and Quality Metrics for Consumer Electronics*, Scottsdale, AZ, 2006, pp. 1–4.

114. F. Battisti, E. Bosc, M. Carli, P. Le Callet, and S. Perugia, “Objective image quality assessment of 3D synthesized views,” *Signal Processing: Image Communication*, vol. 30, pp. 78–88, Jan. 2015.

115. M. S. Farid, M. Lucenteforte, and M. Grangetto, “Perceptual quality assessment of 3d synthesized images,” in *Proceedings of the IEEE International Conference on Multimedia and Expo*, Hong Kong, China, 2017, pp. 505–510.

116. P. Kovesi, “Image features from phase congruency,” *Videre: Journal of Computer Vision Research*, vol. 1, no. 3, pp. 2–26, 1999.

117. C. T. Tsai and H. M. Hang, “Quality assessment of 3D synthesized views with depth map distortion,” in *Proceedings of the IEEE International Conference on Visual Communications and Image Processing*, Kuching, Malaysia, 2013, pp. 1–6.

118. P. Joveluro, H. Malekmohamadi, W. A. Fernando, and A. M. Kondoz, “Perceptual video quality metric for 3d video quality assessment,” in *Proceedings of the 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video*, Tampere, Finland, 2010, pp. 1–4.

119. M. S. Farid, M. Lucenteforte, and M. Grangetto, “Objective quality metric for 3D virtual views,” in *Proceedings of the IEEE International Conference on Image Processing*, Québec City, Canada, 2015, pp. 3720–3724.

120. D. Sandić-Stanković, D. Kukolj, and P. Le Callet, “DIBR synthesized image quality assessment based on morphological wavelets,” in *Proceedings of the Seventh International Workshop on Quality of Multimedia Experience*, Pylos-Nestoras, Greece, 2015, pp. 1–6.

121. ——, “Multi-scale synthesized view assessment based on morphological pyramids,” *Journal of Electrical Engineering*, vol. 67, no. 1, pp. 3–11, Jun. 2016.

122. A. Smolic, H. Brust, K. Müller, M. Müller, and T. Wiegand, “Corrected camera parameters for N9468 "Call for contributions on FTV test material",” ISO/IEC JTC1/SC29/WG11 MPEG, M15101, Antalya, Turkey, Tech. Rep., 2008.

123. M. Tanimoto, T. Fujii, K. Suzuki, N. Fukushima, and Y. Mori, “Reference softwares for depth estimation and view synthesis,” ISO/IEC JTC1/SC29/WG11 MPEG, M15377, Archamps, France, Tech. Rep., 2008.

124. L. K. Choi and A. C. Bovik, “Flicker sensitive motion tuned video quality assessment,” in *Proceedings of the IEEE Southwest Symposium on Image Analysis and Interpretation*, Santa Fe, NM, 2016, pp. 29–32.

125. M. Schmeing and X. Jiang, “Time-consistency of disocclusion filling algorithms in depth image based rendering,” in *Proceedings of the 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video*, no. 1. Antalya, Turkey: IEEE, 2011, pp. 1–4.

126. H. Yang, J. M. Boyce, and A. Stein, “Effective flicker removal from periodic intra frames and accurate flicker measurement,” in *Proceedings of the 15th IEEE International Conference on Image Processing*, San Diego, CA, 2008, pp. 2868–2871.

127. B. Andres, T. Beier, and J. H. Kappes, “OpenGM: A C++ library for discrete graphical models,” *CoRR*, vol. abs/1206.0, pp. 1–5, 2012.

128. J. Redi, E. Siahaan, P. Korshunov, J. Habigt, and T. Hossfeld, “When the crowd challenges the lab: Lessons learnt from subjective studies on image aesthetic appeal,” in *Proceedings of the 4th International Workshop on Crowdsourcing for Multimedia*, Brisbane, Australia, 2015, pp. 33–38.

129. T. Hossfeld, C. Keimel, M. Hirth, B. Gardlo, J. Habigt, K. Diepold, and P. Tran-Gia, “Best practices for qoe crowdtesting: Qoe assessment with crowdsourcing,” *IEEE Transactions on Multimedia*, vol. 16, no. 2, pp. 541–558, Feb. 2014.

130. C. Keimel, J. Habigt, and K. Diepold, “Challenges in crowd-based video quality assessment,” in *Proceedings of the Fourth International Workshop on Quality of Multimedia Experience*, Yarra Valley, Australia, 2012, pp. 13–18.

131. C. Keimel, J. Habigt, C. Horch, and K. Diepold, “Video quality evaluation in the cloud,” in *Proceedings of the 19th International Packet Video Workshop*, Munich, Germany, 2012, pp. 155–160.

132. ——, “Qualitycrowd a framework for crowd-based quality evaluation,” in *Proceedings of the Picture Coding Symposium*, Krakow, Poland, 2012, pp. 245–248.

133. S. Winkler and P. Mohandas, “The evolution of video quality measurement: From PSNR to hybrid metrics,” *IEEE Transactions on Broadcasting*, vol. 54, no. 3, pp. 660–668, Sep. 2008.

134. M. Köppel, P. Ndjiki-Nya, D. Doshkov, H. Lakshman, P. Merkle, K. Müller, and T. Wiegand, "Temporally consistent handling of disocclusions with texture synthesis for depth-image-based rendering," in *Proceedings of the IEEE International Conference on Image Processing*, Hong Kong, China, 2010, pp. 1809–1812.

135. "Recommendation ITU-R BT.500-14: Methodology for the subjective assessment of the quality of television pictures," International Telecommunication Union, Geneva, Switzerland, Tech. Rep., 2019.

136. G. Cermak, L. Thorpe, and M. Pinson, "Test plan for evaluation of video quality models for use with high definition tv content," Visual Quality Experts Group (VQEG), Boulder, CA; Stockholm, Sweden, Tech. Rep., 2009.

137. C. Keimel, A. Redl, and K. Diepold, "Influence of viewing experience and stabilization phase in subjective video testing," in *Proceedings of SPIE Image Quality and System Performance IX*, vol. 8293, Burlingame, CA, 2012, p. 829313.

138. D. G. Altman and J. M. Bland, "Measurement in medicine: The analysis of method comparison studies," *The Statistician*, vol. 32, no. 3, p. 307, Sep. 1983.

A. Appendix

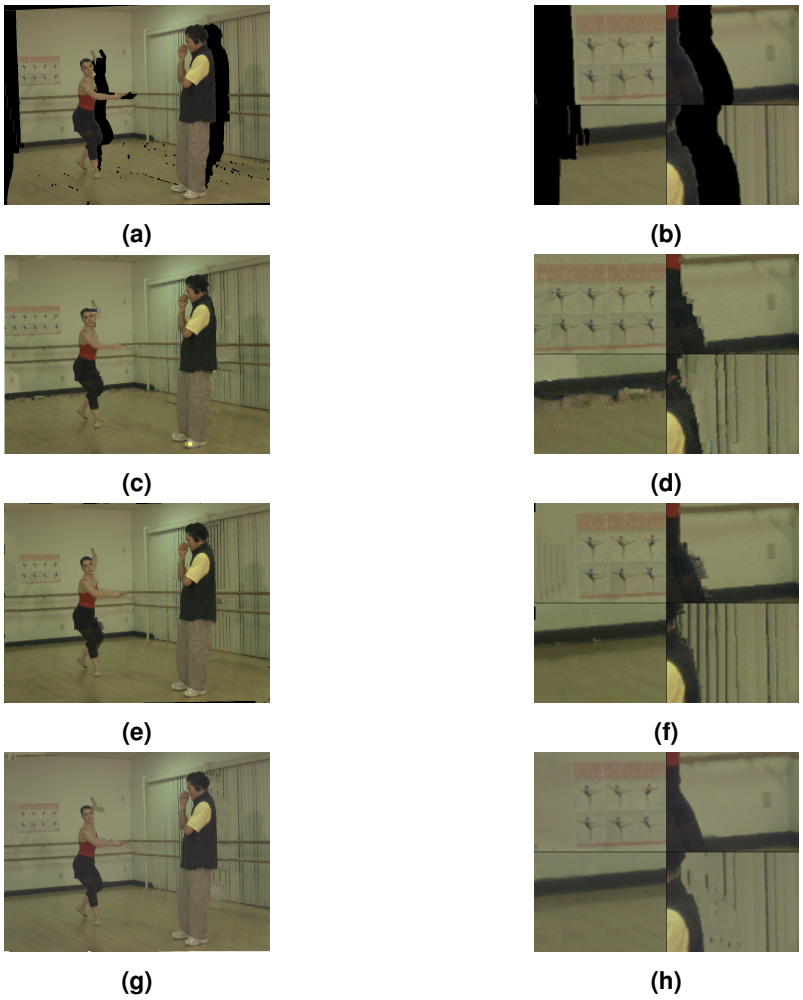


Figure A.1.: Inpainting results for the first frame of the *Ballet* sequence of two state-of-the-art algorithms and VS-BP. (a), (b) Synthesized View; (c), (d) Result of Daribo's method [38]; (e), (f) Result of Gautier's method [40]; (g), (h) VS-BP.

Algorithm 1 VS-CSH

Input: Image I, Depth Map D

Output: Virtual View I_{synth}, Virtual Depth Map D_{synth}

Image Warping

Warp image I to get the virtual view with disocclusions I_{warp}, the depth map from that viewpoint D_{warp} and the mask of the disocclusions M

Enlarge holes by dilating with the structuring element K

$n_{iterations} = 10$

for $s = -5 \ldots 0$ **do**

Resample I_{warp} by a factor 2^s to get I_{res}

Create a copy of I_{res} and fill the holes with random content to get I_t

for t = 1:$n_{iterations}$ **do**

Compute ANN field with CSH between I_{res} and I_t

Hashing

Project all patches of I_{res} and I_{t} onto M Walsh-Hadamard Kernels $\left\{WH_j\right\}_{j=1}^{M}$

Calculate the hash of each patch of I_{res} and I_{t} with the code

$$g_i(\mathbf{p}) = h_1(\mathbf{p}) \circ \cdots \circ h_M(\mathbf{p})$$

Insert hash into hash table $T\,[g_i(\mathbf{p})]$

Search

For each patch $\mathbf{p}$ in I_t find all candidates $\mathbf{c}_i$ using the hash table T

For each candidate $\mathbf{c}$:

if $SSD(\mathbf{p}, \mathbf{c}) < SSD(\mathbf{p}, ANNF(\mathbf{p}))$ **then**

if $D(\mathbf{c}) <= D(\mathbf{p})$ **then** ▷ *Check that candidate is further in the background*

$ANNF(\mathbf{p}) = \mathbf{c}$

end if

end if

Fill Holes

Fill disocclusions ($M = 1$) with all patches of the ANNF which contain $\mathbf{p}$

end for

end for