Graz University of Technology
Institute for Computer Graphics and Vision

Master's Thesis

Multi-frame Rate Augmented Reality

Philipp Grasmug

Graz, Austria, December 2013

Advisor

Prof. Dr. Dieter Schmalstieg

Institute for Computer Graphics and Vision, Graz University of Technology

Deutsche Fassung:
Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008
Genehmigung des Senates am 1.12.2008

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am …… ……..
(Unterschrift)

Englische Fassung:

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

…… ……..
date (signature)

Abstract

In this work we present a method for improving the visual quality of an augmented reality system. By combining the characteristics of two different sensors, we increase the spatial resolution of a video stream using sub-pixel accurate image registration. By decoupling the rendering process of the augmented information from the displaying frequency of the system, we can augment the scene using computationally expensive rendering techniques. We utilize image-based rendering to overcome the resulting temporal artifacts. Moreover, we evaluated our methods by comparing the achieved quality with conventional augmented reality methods. Finally, we explain limitations and assumptions of our algorithm and discuss further work.

Kurzfassung

Mit dieser Arbeit präsentieren wir einen Ansatz zur Qualitätssteigerung von Augemented Reality Systemen. Die Qualität des Video Signals wird unter Zuhilfenahme von Standbildern, die mit einem zweiten Sensor aufgenommen werden, verbessert. Dies geschieht indem der Versatz zwischen den Pixeln beider Bilder berechnet wird und eine nicht-lineare Verschiebung anhand des Ergebnisses dieser Berechnung durchgeführt wird. Um die visuelle Qualität der eingeblendeten Information zu steigern, entkoppeln wir den Berechnungsprozess von der Berechnungsrate des restlichen System. Das erlaubt uns ein größeres Zeitfenster zu nutzen wodurch auch aufwändige Rendering Algorithmen zum Einsatz kommen können. Die entstehende Lücke zwischen den Berechnungsraten wird mithilfe eines Bild-basierendem Rendering Verfahrens gelöst. Wir evaluieren die erzielten Ergebnisse, speziell im Hinblick auf Qualität, im Vergleich zu konventionellen Methoden. Zudem erläutern wir die Einschränkungen unseres Systems und diskutieren mögliche weitere Forschungspunkte.

Contents

List of Figures

Chapter 1

Introduction

Contents

Augmented reality is a topic that draws a lot of attention lately from different fields of science. Enriching the environment with computer generated information makes sense in many scenarios like, for example games, but also medical and technical applications. Allowing the user to see or even interact with virtual elements creates a new way of human-computer interaction that is more natural than exploring an environment with mouse and keyboard. By showing the user, for example, what is below the surface or what could be in some place in the real environment generates an enhanced perception of virtual information.

Modern mobile devices with a programmable graphics processing unit (GPU) allow to run such applications on smartphones or tablets. This makes the technique even more versatile, since it enables augmented reality application to take place in every day situations. This leads to more attention on this topic, since more and more people see the potential of augmented reality applications.

Increasing the spatial resolution of images is normally achieved by developing new sensor chips for cameras with a higher pixel density or larger sensors. Both of these improvements have drawbacks in terms of image quality (smaller

Figure 1.1: Exemplary result comparing the output of our system to an image bilinear interpolated to the same resolution.

pixels mean less light, leading to more noise) and efficiency. Also, capturing images at a high resolution means a lot of data has to be transferred, which requires a bus with sufficient transfer rate. But there are also software approaches for performing that task. These approaches often combine information from multiple subsequent images of the current scene or arbitrary images from a database to compute a realistic or at least plausible high resolution version of the input image. Combining images from different sensors is alternative method, which has been researched lately. A high spatial resolution is desirable in many cases, since it offers more details. For fields like, for instance, medical and forensic imaging, a high level of detail often means better examination results, since those task often rely on small subtle differences.

The augmented information in the applications described earlier ranges from simple textual information to visualization of volumetric data sets or photo realistic rendering. The computation of such augmentations, especially in high resolution, is time consuming and often not achievable in real time. Figure 1.1 compares the result of our approach to an image bilineary interpolated to the same output resolution.

1.1 Contribution

Augmented reality deals with blending information into the environment using a device for capturing the scene (in most cases, a video camera) and a displaying device like a smartphone, tablet or screen. Augmented reality is used to enrich scenes with information or to provide a user interface to view virtual objects in a real scene. Typically capturing and tracking the scene and enriching it with information are the basic steps involved. Increasing the resolution of an image or video is usually done by combining information from different subsequent images or from an image database. Combining those images is slow and beyond real time since a search operation has to be performed in order to detect matching image fragments.

We first present an approach that accomplishes this task by combining advantages of different sensors. Photo cameras provide images in high spatial resolution at a low frame rate, whereas video cameras provide video streams in lower quality, but at a higher frame rate. Using a sub-pixel accurate registration of the images, we can combine the advantages of both sensors and make a step towards interactivity of super-resolution.

The second part of our work focuses on improving the quality of the augmentation. Depending on the method, rendering information in high quality (not only high spatial resolution) is slow and has only limited interactivity. If we want to enrich the scene using high quality renderings we have to handle a disparity between the frame rate of the video stream and the rate at which the augmentation is computed. Image-based rendering is a method that is capable of rendering novel views from single or multiple two dimensional images. Utilizing the concept of image-based rendering, we

can solve the frame rate divergence issue. Without this step, the augmentation would jitter due to the different frame rate and result in a bad user experience.

Tracking algorithms are used in our system for the augmentation of the scene, but are outside the scope of this work. Our method is agnostic in terms of hardware and also rendering techniques.

1.2 Organisation of this Work

The following chapter gives an overview of the methods and algorithms that are related to our work and on which our system is based. We describe selected methods for optical flow computation and their details. Furthermore, we introduce image-based rendering methods, which are relevant for our work. Finally, we describe approaches for photorealistic rendering with their advantages and drawbacks and algorithms for computing high resolution images from low resolution input data.

The third chapter describes our system as well as the possible hardware setups. We cover the different stages involved in our multi-frame rate setup and also discuss critical points and issues and how we addressed them.

Chapter four shows an evaluation of our work on different scenarios. We compare the quality increases we were able to achieve with the ground truth and give interpretations on experienced effects.

The last chapter concludes the thesis. It discusses the results we could achieve and gives an outlook on future work.

Chapter 2

Related Work

Contents

2.1 Optical Flow

The optical flow describes the motion of the pixels in an image from one frame to a subsequent one[Gib50]. Optical flow can result from movement of the viewer or from movement of the objects in the scene. Computing the flow field for a sequence of images is a challenging task due to ambiguities in the nature of the problem as discussed in the following paragraph. In order to be able to solve the optical flow problem, some assumptions have to be made. One of the most basic assumptions[o2005] is the brightness constancy. Brightness constancy assumes that even if the position of an object changes over a small period of time, its illumination and therefore brightness pattern will remain the same. Equation 2.1 models this assumption.

$$f(x + \Delta x, y + \Delta y, t + \Delta t) \approxeq f(x, y, t) \tag{2.1}$$

$f(x, y, t)$ describes the brightness of a pixel at position x and y at time t where Δx and Δy are the changes in position and Δt is the change in time. This equation gives a good starting point for estimating the optical flow, but still leaves some problems. One of the most commonly mentioned problems is the aperture problem [PK85]. If

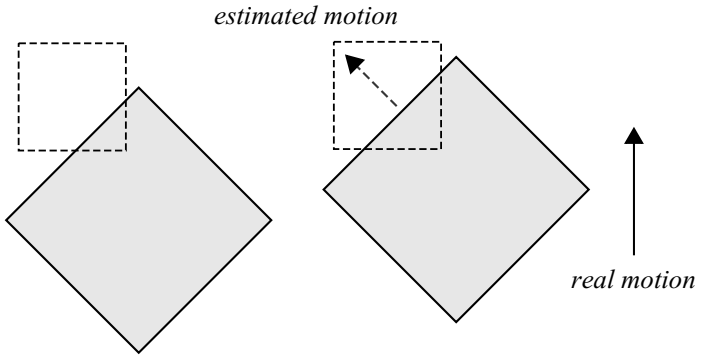


Figure 2.1: Illustration of the aperture problem. The dashed square represents the view region of the subsequent images. When viewing only a small region of the image, the estimated motion differs from the real motion.

only a small aperture of two subsequent images is seen the estimated motion can be ambiguous, since the spatial gradient at those points only contains one component or even vanishes. Figure 2.1 illustrates this issue. Since the flow vector consists of two components, further steps have to be made in order to be able to compute a valid solution. From this example one might also see that computing the flow from a feature-rich image is less challenging than from a feature-poor one. The task of determining the optical flow always is an optimization problem, since in most cases ambiguous solutions exist. By setting up constraints like piecewise smoothness of the flow field, the solution is regularized to be a good approximation of the motion in the scene. Without the regularization terms, the solution would simply tend to fullfill the basic assumptions.

$$f(x + \Delta x, y + \Delta y, t + \Delta t) = \frac{\partial f}{\partial x}\Delta x + \frac{\partial f}{\partial y}\Delta y + \frac{\partial f}{\partial t}\Delta t + \dots \tag{2.2}$$

$$\nabla I \cdot v + I_t = 0 \tag{2.3}$$

By applying a Taylor expansion to the left part of equation 2.1 and substituting equation 2.2 into 2.1, we get a common formulation of the brightness constancy

assumption where $\nabla I = (I_x, I_y)$ and $v = (u, v)$, with I_x and I_y being the derivatives of the image with respect to x and y. Δt vanishes since it is 1. This formulation is a basic model on which many algorithms are built.

The applications for optical flow estimation are numerous and can be found in different fields. Common examples are segmentation, object tracking and sub-pixel accurate image registration. Computing the optical flow gives pixelwise correspondences, which further can be used to reconstruct depth information, if the camera intrinsics and extrinsics are known. By intersecting two rays, which are sent from the center of projection of each camera through a pixel and the corresponding pixel in the second image into the scene, depth information can be retrieved. A distinction has to be made between algorithms which yield sparse flow fields and those who produce dense fields. Since we use the optical flow to register two images of the same scene in a sub-pixel accurate manner, an algorithm which computes a dense flow field is needed. Computing a dense flow field from a sparse one using interpolation is also an option and discussed later in section 3. In the following subsections, we describe a few of the many different motion estimation algorithms, which are important to us and influenced our work.

2.1.1 Horn and Schunck

Horn and Schunck [HS81] proposed a global method for computing the optical flow. In this case, global means that every pixel in the image can have influence on every other pixel in the image. This can be helpful especially with homogeneous regions, since flow information is propagated from edges and corners over those regions. This method is based on equation 2.1, but adds a second term called the smoothness constraint which is defined as $(\frac{\partial u}{\partial x})^2 + (\frac{\partial u}{\partial y})^2$ and $(\frac{\partial v}{\partial x})^2 + (\frac{\partial v}{\partial y})^2$. This constraint ensures that the flow in the neighbourhood is smooth and discontinuities are rare.

$$E^2 = \int\int (\nabla I \cdot v + I_t) + \alpha^2 \left(\left(\frac{\partial u}{\partial x} \right)^2 + \left(\frac{\partial u}{\partial y} \right)^2 + \left(\frac{\partial v}{\partial x} \right)^2 + \left(\frac{\partial v}{\partial y} \right)^2 \right) \mathrm{d}x \, \mathrm{d}y \quad (2.4)$$

2.4 shows the model of the algorithm, where α is a weighting factor for the smoothness constraint. Introducing an approximation $\overline{u}, \overline{v}$ that is an average of the flow

components u, v in a small neighbourhood for the Laplace operators, the following formulation can be written.

$$(\alpha^2 + I_x^2 + I_y^2)(u - \overline{u}) = -I_x[I_x\overline{u} + I_y\overline{v} + I_t] \tag{2.5}$$

$$(\alpha^2 + I_x^2 + I_y^2)(v - \overline{v}) = -I_y[I_x\overline{u} + I_y\overline{v} + I_t] \tag{2.6}$$

This gives a pair of equations for every pixel in the image which can be solved by the following iterative scheme.

$$u^{n+1} = \overline{u}^n - \frac{I_x[I_x\overline{u}^n + I_y\overline{v}^n + I_t]}{(\alpha^2 + I_x^2 + I_y^2)} \tag{2.7}$$

$$v^{n+1} = \overline{v}^n - \frac{I_x[I_x\overline{u}^n + I_y\overline{v}^n + I_t]}{(\alpha^2 + I_x^2 + I_y^2)} \tag{2.8}$$

It is noticeable that the calculation of u^{n+1} and v^{n+1} don't directly rely on previous values of u and v but rather their averaged values $\overline{u}$ and $\overline{v}$. This leads to the effect that homogeneous regions can be filled with values from corner and edges if enough iteration steps are done.

A drawback of this model is the quadratic smoothness term in equation 2.4. Due to this term, discontinuities in the flow field are unlikely since outliers penalize the energy function strongly. We will see solutions to this problem in approaches described in later sections.

2.1.2 Lucas and Kanade

The algorithm of the Lucas and Kanade [LK+81] is based on the idea of matching two curves in one dimensional space like illustrated in figure 2.2. The solution uses the well known linear approximation of the first derivative of a function $f'(x) \approx \frac{f(x+h)-f(x)}{h}$, which can be reformulated to

$$h \approx \frac{g(x) - f(x)}{f'(x)} \tag{2.9}$$

by substituting $f(x+h)$ with $g(x)$. This approximation of $f'(x)$ works well for small values of h. By computing this for multiple values of x, a better approximation can be found. Naturally a solution for h is good if f has a linear behaviour in

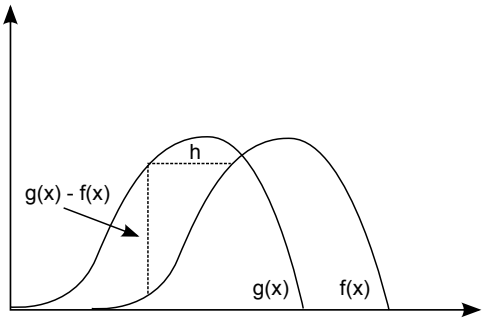


Figure 2.2: Matching two curves by finding the disparity vector h

regions between $f(x)$ and $f(x+h)$. To assure that good approximations of h are favoured, a weighting function $w(x) = \frac{1}{f''(x)}$ is introduced. Using this function, regions with low curvature are favoured over regions with high curvature. Combining those formulations in an iterative manner further gives us

$$h_{k+1} = h_k + \sum_{x \in \Omega} \frac{w(x)\,(g(x) - f(x+h_k))}{f'(x+h_k)} / \sum_{x \in \Omega} w(x) \tag{2.10}$$

where Ω denotes the sampled neighbourhood. This idea then is generalized for two or more dimensions by defining x and h as n-dimensional row vectors. This gives us the following formulation for $f(x+h)$.

$$f(x+h) \approx f(x) + h\frac{\partial}{\partial x} f(x) \tag{2.11}$$

The proposed algorithm is in general faster since the calculation of the disparity vector h, which represents the flow vector for a given pixel, is based on a small local neighbourhood. A problem that arises with this method is that the flow field is only dense in high-frequency regions of the image.

Anandan [Ana89] described a multi scale approach to address this issue. The flow vector is computed in a coarse-to-fine approach, where the result of the previous level is used as initial estimation for the next level. Another problem related with all differential methods is the appearance of noise and discontinuities in images, which leads to errors in the derivatives. Especially for global methods like the Horn/Schunck algorithm (2.1.1), this is an issue, while local methods tend to be robust against noise. To resolve this problem, it is common to add a

smoothing term. Bruhn et al. [BWS05] research different smoothing techniques in this context and propose a hybrid algorithm that combines local and global approaches in a multi-scale fashion.

2.1.3 Brox

Brox et al. [BBPW04] published a variation based algorithm that incorporated different improvements from other flow estimation algorithms. First of all, the algorithm uses a brightness gradient constancy assumption defined as $\nabla I(x, y, t) = \nabla I(x+u, y+v, t+1)$ in addition to the well known brightness constancy assumption. This assumption makes the model invariant to changes in illumination. Furthermore, a function $\psi(s^2) = \sqrt{s^2 + \varepsilon^2}$ is applied to the data and the smoothness term, which leads to less penalization of outliers and a robust energy function. Since ε is a constant to preserve the convexity of the function, no new parameters are introduced. The data term is therefore defined as

$$E_{Data}(u, v) = \int_\Omega \psi \left(|I_t(x + w) - I_{t+1}(x)|^2 + \gamma |\nabla I_t(x + w) - \nabla I_{t+1}(x)|^2 \right) \mathrm{d}x \quad (2.12)$$

where γ is a weighting parameter for the brightness gradient constancy term. The smoothness term is defined similar to the smoothness constraint of Horn and Schunck (2.4), with the addition of the ψ function, which yields

$$E_{smooth}(u, v) = \int_\Omega \psi \left(|\nabla_3 u|^2 + \nabla_3 v|^2 \right) \mathrm{d}x \quad (2.13)$$

where $\nabla_3 := (\partial_x, \partial_y, \partial_t)$. Incorporating the partial derivative t ensures not only a spatial, but also temporal smooth flow field. If only a sequence of two images is available, this gradient is replaced by a spatial-only version. The intention of applying the ψ function to the smoothness term is to receive a piecewise smooth flow field due to the eased penalization of outliers.

2.1.4 TV-L1 Flow

Werlberger and Pock [WPB10] proposed an algorithm that is based on non-local total variation. The regularization term is defined as

$$R(u) = \int_\Omega \int_\Omega w(x,y)(|u_1(x) - u_1(y)|_\epsilon + |u_2(x) - u_2(y)|_{\epsilon)}\,\mathrm{d}x\,\mathrm{d}y \tag{2.14}$$

where $u = (u_1, u_2)$ is the flow vector, x, y are positions in the image and $|q|_\epsilon$ denotes the Huber norm. The weighting function, $w(x, y)$ calculates the weight factor based on color similarity and spatial distance. Using this function, a low level segmentation is encoded into the regularization. The classic brightness constancy constraint which has been discussed earlier, fails in many real world scenarios and therefore leads to bad results. Werlberger and Pock address this issue by using the normalized cross correlation for their data term, since it is invariant to changes in illumination due to the normalization.

2.1.5 Simple Flow

Tao et al. [TBKP12] published an algorithm for computing the optical flow especially on high resolution images. The algorithm uses the following simple formulation for the error function

$$e(x_0, y_0, u, v) = |I_t(x_0, y_0) - I_{t+1}(x_0 + u, y_0 + v)|^2 \tag{2.15}$$

where x_0, y_0 denotes the position of the current processed pixel, u, v the flow components and I_t is the image at time t. To ensure smoothness of the flow field, a solution for u, v is sought that is also plausible for the neighbourhood around x_0, y_0. This constraint is modelled as

$$(u_0, v_0) = argmin_{(u,v)\in\Omega} \sum_{(x,y)\in N} e(x, y, u, v) \tag{2.16}$$

where Ω is the set of possible values for u, v and N is the neighbourhood around the current processed pixel. To account for discontinuities, a weighting function similar

to joint bilateral filtering [TM98] is applied, which yields the following formulation

$$E(x_0, y_0, u, v) = \sum_{(x,y)\in N} w_d w_c e(x, y, u, v) \tag{2.17}$$

$$w_d = exp(-|(x_0, y_0) - (x, y)|^2/2\sigma_d) \tag{2.18}$$

$$w_c = exp(-|I(x_0, y_0) - I(x, y)|^2/2\sigma_c) \tag{2.19}$$

where x_0, y_0 is the current processed pixel. The solution is now obtained by finding the u, v pair that minimizes E. This is done by calculating E for all possible $u, v \in \Omega$. Since only integer values are considered, the number of possible solutions is given by N^2. To capture large movement, the neighbourhood N has to be very large. Therefore, a multi scale approach is used to address this issue. A initial estimate is computed by upsampling the result from the previous scale level. This step is optimized by introducing a irregularity estimate defined as $max_{(x,y)\in N}|u(x) - u(x_0) + v(x) - v(x_0)|$. If this value is above a certain threshold, the full computation is performed. Otherwise the flow is interpolated for this pixel in order to save computation time. Since the calculations for each pixel are independent of every other pixel, this algorithm is highly parallelizable and well suited for the GPU.

2.1.6 SIFT Flow

The usual approach for computing the optical flow is to use brightness or brightness gradient differences to identify matching pixels. [LYT11] described a different approach that uses a 128-dimensional SIFT[Low99] feature descriptor per pixel for matching. The idea behind this approach is to be able to compute the optical flow between two images that only partly show the same scene or show even different scenes but similar objects. The formulation of the model used for minimization is standard and solved using dual layer belief propagation in a multi-scale matching scheme. Since the images in our system, which have to be registered, show a large disparity in some scenarios, especially if two different sensors are used, this approach would be ideal. The experiments show that the algorithm produces good results, but is slow and has a high memory consumption, which makes it useless for our scenario.

2.2 Image-based Rendering

The general idea of image based rendering (IBR) is to derive a novel view from real or synthetic images. The usual way of rendering an image of a synthetic scene is by using one of the standard algorithms like, for example, rasterization, ray tracing or path tracing. Especially when realistic images are desired, the computational complexity is high and computing such images takes a lot of time. Another issue related with these approaches is the varying runtime complexity. Depending on what is currently viewed by the virtual camera, rendering times can deviate strongly. In contrast, the computational cost of image based rendering methods is constant, since it only relates to the number of processed pixel instead of the scene complexity. It cannot be said in general that IBR is faster than conventional rendering, but it is more predictable due to its constant computational cost.

A goal of our system was to increase the quality and coherence of the augmented information, for which we decided to use physical based rendering techniques. Since realistic rendering methods are hardly computable in real time - especially at high resolutions - we were looking for a way to overcome this divergence in the frame rate at which the augmentations can be computed and the rate at which the display is refreshed. The user of an augmented reality system has to view the scene using a device like a camera, smartphone or tablet. Therefore, the motion of the camera is predictable and jumps from one view to a totally different one are unlikely. This frame-to-frame coherence in combination with the constant computational cost makes IBR techniques a perfect candidate for overcoming this gap.

A common way for introducing IBR methods is by starting with the *plenoptic function* introduced by Adelson et al. [AB91] which is defined as

$$P_7 = P(V_x, V_y, V_z, \Theta, \Phi, \lambda, t) \tag{2.20}$$

where V_x, V_y, V_z denote the position of the viewer looking in a direction given by the angles Θ and Φ with a specific wavelength λ at time t. This function is not only a basic formulation for IBR techniques, but rather a general model for all rendering

algorithms. Different simplifications of this model have been made where P_2 is the simplest defined as

$$P_2 = P(\Theta, \Phi) \tag{2.21}$$

P_2 describes a panorama view with fixed position, wavelength and time. In the following sections we will describe different IBR methods which will refer to this generic definition of rendering in different forms. A usual way to categorize IBR techniques is by using the image based rendering continuum [Len98, Kan97] shown in figure 2.3. Another categorization, related to the IBR continuum, is the classification into methods with no geometry, methods with implicit geometry and methods with explicit geometry [SK00]. Implicit geometry are, for example, point correspondences where explicit geometry is depth information or underlying mesh data.

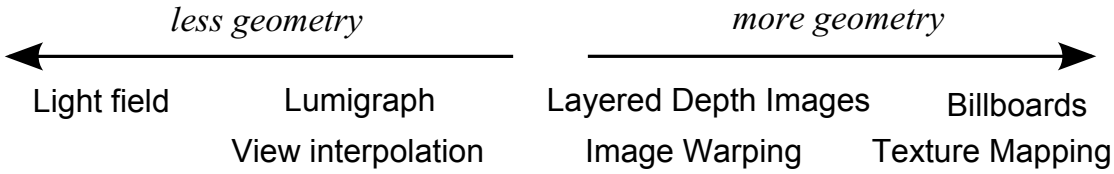


Figure 2.3: The image based rendering continuum categorizing different methods by their use of geometric information of the scene.

2.2.1 View Interpolation

View interpolation [CW93] uses dense correspondences - in other words, the optical flow - between pairs of images to interpolate intermediate views. Where computing the optical flow for synthetic images is easy (camera pose and per pixel depth information is given for both views), algorithms described in section 2.1 have to be used for real images. The quality of the computed novel views strongly depends on the quality of the optical flow. This approach works well if the images are similar. Since linear interpolation is used, only parallax and linear movement is approximated well. Problems that arise with this approach are overlaps of pixels (two ore more input pixels are mapped onto one output pixel) and holes due to missing information (mostly because of disocclusion). When depth information is given (in the case of synthetic images), the overlap problem can be solved using the z-buffer algorithm.

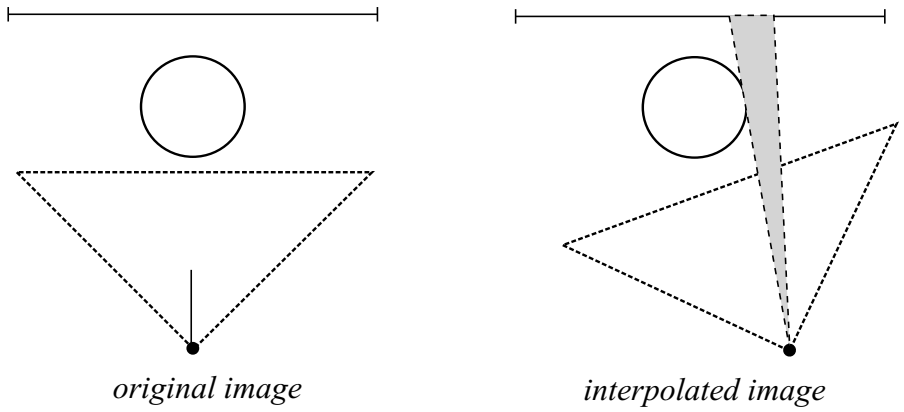


Figure 2.4: Holes result from changes in perspective which reveal areas that are not visible in the original image. The gray area is revealed in the interpolated image but no information is available.

Holes are a problem that come with many IBR techniques. Due to change in perspective, areas are revealed which were not visible in the source image. This problem is illustrated in figure 2.4. Holes and cracks also appear if the camera of the interpolated view is closer to the scene than the original one. This can be seen as a resampling problem, where the sampling rate of the reconstruction is higher than the original one. The work of Chen et al. addresses this issue by warping from more than one image and by interpolating from neighbouring pixels.

2.2.2 Layered Depth Images

Shade et al. [SGHS98] proposed two novel IBR primitives with per-pixel depth information. *Sprites with depth* are an extension to traditional sprites. Using the associated depth information parallax can be simulated by warping the depth values to the new view and correcting the pixel positions accordingly. This technique adds realism to a scene but cannot resolve the previously discussed hole issue. *Layered Depth images (LDI)*, the second introduced primitive, store a list of color-depth duples per pixel in front-to-back order. Every object that is hit by the viewing ray of the pixel is included in the LDI. The complexity of the image therefore scales linearly with the depth complexity of the scene. By warping the pixels using the depth information and rendering them in back-to-front order the

hole issue can be addressed while providing correct alpha blending. Parts of the image which are revealed due to the change in perspective are taken from one of the other layers. Further the previously discussed resampling problem can be addressed by splatting the pixels in the needed size.

The work of Shade et al. also stated that IBR primitives should be selected according to the depth complexity of the object and especially the distance to the camera. Objects far away can be represented using simple environment maps or sprites since changes in perspective will only produce small parallax effects which are hardly noticeable and therefore can be ignored. In opposite objects which are close to the camera (and which have a large internal depth complexity) show strong parallax effects and also disocclusion which require more sophisticated techniques.

2.2.3 Image Warping

The idea of *Image Warping* [MB95, MJ97] relies on associated per pixel depth information. This information together with the position and orientation of the camera is used to re-project each pixel into three dimensional space and then into the view of the new desired virtual camera. For synthetic scenes this is pretty straight forward, since storing per-pixel depth values and camera position can be easily done during rendering. The problems of holes and resampling are also present in this approach and can again be addressed with warping from multiple images and splatting of pixels. Another issue with this algorithm is visibility. When more than one pixel from the original image are warped to the same output pixel, some kind of depth test has to be performed, otherwise artefacts arise.

Image Warping is a very suitable method for our purpose, since the required information is easily available, and the warping process can be done within milliseconds on modern programmable graphics hardware. This makes this technique an ideal candidate for overcoming the frame rate gap in the augmentation part of our system. Visibility issues and gaps can also be addressed efficiently, as described later in section 3.

2.2.4 Billboards

Billboards are just planar geometry with an assigned texture, which are normally used to represent objects far away from the camera. Due to their simplistic nature, the quality of their representation is rather limited, since no parallax effects appear. Billboards can be aligned perpendicularly with the camera or arbitrarily in space. Large movement of the camera, especially rotation around the billboard, reveals its flat nature, which limits the use to far away placement. In section 2.2.2 we already mentioned an extension called Sprites with Depth, which adds depth information to simulate parallax effects.

2.2.5 View-depended Texture Maps

Applying textures to a geometric model is a widely used technique for adding details and realism to objects. Different lighting effects than can be observed in nature depend on the viewers position to the object. These effects cannot be simulated using classic texture mapping. View-dependend texture mapping [DTM96] uses multiple images taken from different views to simulate these effects. The textures are blended according to the affinity of the current view and the view from which the texture was originally taken.

2.3 Photorealistic Rendering

Since the dawn of computer graphics, researchers tried to find ways and methods to produce results which looked better and more realistic. Soon it was obvious that creating photorealistic looking images is a computationally expensive task. With the advent of dedicated graphics hardware, it was possible to produce decent looking images in real time. Realtime graphics used, for example, in games or simulations try to approximate visual effects, so that they can be computed within a few milliseconds. Physically based rendering tries to model real lighting phenomena and solves them using various approaches. Direct (local) illumination is the light that comes directly from a light source and can be computed easily. What adds realism to a scene is the indirect (global) illumination. That is the integral over all light rays which are reflected, refracted and also blocked (in case of shadows) for

a particular point. Calculation of the global illumination is a complex task that is computationally expensive. Due to programmable graphics hardware, those calculations can be done considerably faster than years ago, but still not in real time.

A good starting point to describe physically based rendering is the *rendering equation*, which was simultaneously introduced by [Kaj86] and [ICG86]. The rendering equation is a general model that describes the radiance which leaves a point x in direction ω_0 and can be defined as

$$L_o(x, \vec{\omega_o}) = L_e(x, \vec{\omega_o}) + \int_\Omega f_r(x, \vec{\omega_i}, \vec{\omega_o}) \cdot L_i(x, \vec{\omega_o}) \cdot (\vec{\omega_i} \cdot n)\, \mathrm{d}w_i \tag{2.22}$$

where

x ..... location
ω_o ..... outgoing direction
ω_i ..... incoming direction
Ω ..... hemisphere over all possible values of ω_i
L_e ..... emitted radiance
f_r ..... BRDF* giving the proportion of the light which is reflected
L_i ..... incoming light from direction ω_i
n ..... surface normal

In the following subsections, we describe various approaches to solving this equation. Some of them make drastic simplifications to this equation in order to make it easier to compute.

2.3.1 Raytracing

The basic concept of ray tracing is to shoot a ray from the eye position through every pixel of the image plane into the scene. For each ray, the closest intersection is stored and the pixel is coloured according to the material properties of the object. This concept was introduced by Appel [App68] and had a huge advantage over scanline rasterization. Every object for which a ray intersection equation can be defined is easily displayable using this algorithm. Whitted [Whi79] developed this idea further

*Bidirectional reflectance distribution function

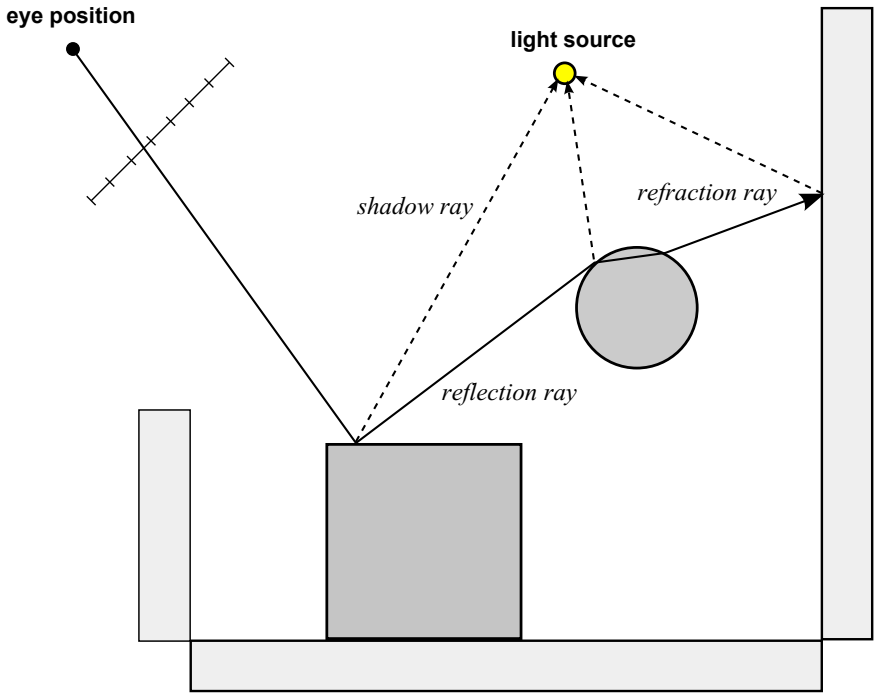


Figure 2.5: The concept of recursive raytracing.

by introducing recursive ray tracing. Figure 2.5 illustrates the idea. As soon as a ray hits an object, it is further traced as *reflection ray*, *refraction ray* and/or *shadow ray*. The reflection ray is calculated from the incident angle of the original ray and the surface normal, the refraction ray is calculated from the surface properties (material) of the object and can be an entering or exiting ray and, finally, shadow rays are simple rays from the intersection point to every light source. Recursively tracing these three types of rays, reflections, transparent objects and hard shadows can be simulated. This extension is a step towards realism, but still far away from photo realistic rendering, since global illumination is missing. A problem with this approach is that reflections and refractions are perfect, and shadows are hard, which is not what we observe in the real world. Cook et al. [CPC84] presented an approach to this issue based on oversampling, called distributed ray tracing. The basic concept of ray tracing makes three assumptions to simplify the rendering equation.

1. The incoming light is only considered from the positions of the light sources.

2. Reflected light is only considered from the perfect mirroring direction.

3. Refracted light is only considered from the perfect refraction direction.

Therefore the integral is replaced by a sum over all recursion steps weighted by their importance. Due to this simplification a lot of the realism in the resulting images is lost. By distributing rays according to mirror or refraction functions, diffuse reflection and refractions can be achieved. Soft shadows can be simulated by distributing rays over the area of the light source. Utilizing the same concept in a temporal manner, motion blur can be realized. Adding this extensions drastically improves the realism of raytraced images, but with the drawback of high computational cost.

Not only improvements in quality, but also in computational complexity of the algorithm have been made. In the original formulation each ray was intersected with each object in the scene, which lead to a high number of unnecessary intersection tests. By spatially subdividing the scene using an octree [Gla84], the number of intersection tests can be cut down dramatically.

2.3.2 Radiosity

The radiosity algorithm [GTGB84] is another approach for solving the rendering equation, with the limitation that every surface is considered to be diffuse. For the computation of the radiosity the scene is divided into patches. The calculation for each patch is based on equation 2.23

$$B_e = E_e + \rho_e \sum_{s=1}^{n} B_s F_{es} \tag{2.23}$$

B_e denotes the emitted radiosity of a patch, which is the self-emitted radiosity E_e plus the sum of the radiosity of all other patches times a form factor F_{es} between the emitting and the receiving patch. The form factor denotes how much light is transported from the sender to the receiver. The sum of the radiositys of all patches is weighted by the reflexivity ρ_e. This gives a set of linear equations, which can be written as

$$\begin{aligned} B_1 &= E_1 + \rho_1(B_1 \cdot F_{11} + \cdots + B_n \cdot F_{1n}) \\ &\vdots \\ B_n &= E_n + \rho_n(B_n \cdot F_{n1} + \cdots + B_n \cdot F_{nn}) \end{aligned} \tag{2.24}$$

which can be reformulated to

$$\begin{aligned} B_1 - \rho_1(B_1 \cdot F_{11} + \cdots + B_n \cdot F_{1n}) &= E_1 \\ &\vdots \\ B_n - \rho_n(B_n \cdot F_{n1} + \cdots + B_n \cdot F_{nn}) &= E_n \end{aligned} \tag{2.25}$$

and finally written in matrix notation as

$$\begin{pmatrix} (1-\rho_1 \cdot F_{11}) & \rho_1 \cdot F_{12} & \cdots & \rho_1 \cdot F_{1n} \\ \rho_2 \cdot F_{21} & (1-\rho_2 \cdot F_{22}) & \cdots & \rho_1 \cdot F_{2n} \\ \vdots & \vdots & & \vdots \\ \rho_n \cdot F_{n1} & \rho_n \cdot F_{n2} & \cdots & (1-\rho_n \cdot F_{nn}) \end{pmatrix} \begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{pmatrix} = \begin{pmatrix} E_1 \\ E_2 \\ \vdots \\ E_n \end{pmatrix} \tag{2.26}$$

Solving this set of equations gives the emitted radiance for each patch, which is then interpolated for every pixel to get a smooth result. This approach works well as long as no patch occludes another patch and therefore is not or only partly visible. In that case, the calculated form factor between those two patches is wrong. This problem can be solved by sending rays from on patch to uniformly distributed sampling points on the surface of the other patch. The form factor is multiplied by the visibility factor, which is given as the number of unblocked rays divided by the total number of traced rays.

A huge drawback of the original radiosity algorithm is the high computational and also space complexity. Cohen et al. [CCWG88] proposed a algorithm called *progressive refinement* for faster computation of the radiosity of a scene. For each patch the total radiosity and the unsent radiosity is stored. In each iteration step a patch with a high unset radiosity is selected. This patch then sends its radiosity to all other patches. With this approach only n form factors have to be computed each iteration (n^2 in the original algorithm).

Hanrahan et al. [HSA91] proposed a further extension towards making the algorithm converge faster. Using different subdivision levels for the patches and a associated tree structure, the number of iterations can be cut down. The idea is to calculate the transfer on high levels of the patch tree and only refine, if it is necessary. Refinement can happen at the sender or the receiver side. The decision

whether it is necessary to refine or not can be based on different heuristics, like, for example, if the form factor is above a certain threshold.

2.3.3 Path Tracing

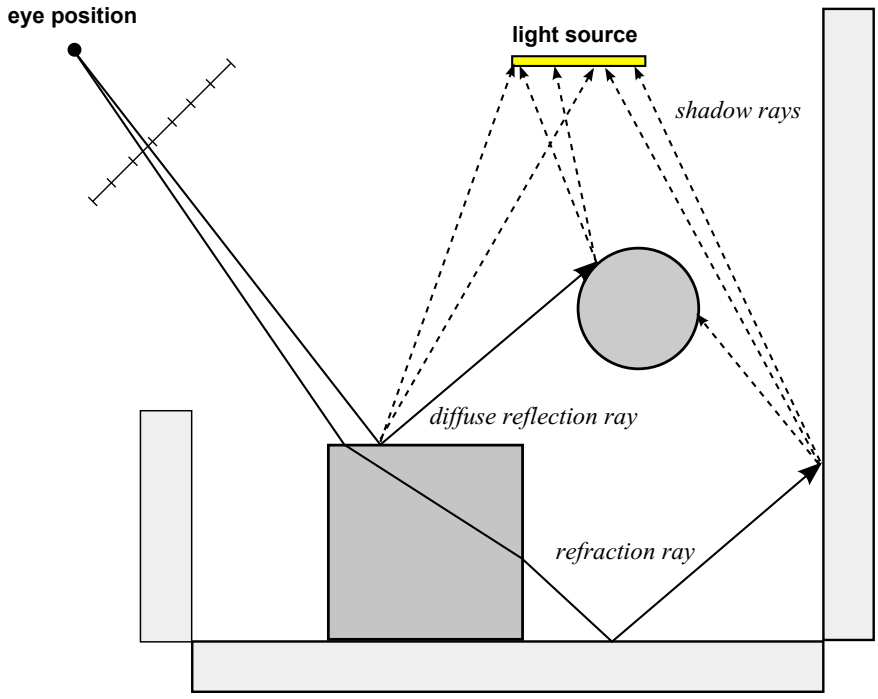


Figure 2.6: The concept of path tracing. For the sake of simplicity, shadow rays are not drawn for all intersection points.

Path tracing [Kaj86] is a an approach on solving the rendering equation using Monte Carlo Integration. Monte Carlo Integration approximates definite integrals by randomly sampling them.

$$\int_a^b f(x)\,\mathrm{d}x \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)} \tag{2.27}$$

In equation 2.27 $p(x)$ is an arbitrary density function of the distribution of the samples. Applying this technique onto the integral of the rendering equation gives good unbiased results. The basic idea is to generate new rays according to a density distribution function every time a ray hits a surface. Those paths are

traced and the radiances are collected along this path (this is essentially what distributed ray tracing does). There are two major problems with this algorithm. First, the number of rays grows exponentially with every indirection step. Second, it takes a lot of samples to get a high probability of hitting a light source.

To account for those issues the following adaptations have to be made: Only one path per ray is traced. Therefore rays are not split when an object is hit. For every pixel not only one, but N rays are sampled and the average value is taken. Furthermore, the calculation of the lighting is split up into direct and indirect illumination. With this extension, the direct illumination is computed at every intersection. This can be done by sampling over the area of the light source or by sampling over the hemisphere of the intersection point. Figure 2.6 illustrates that idea.

A further question that comes with this approach is when to stop the indirection of a ray. Using a fixed value would introduce a biased error in the resulting image. Therefore a *Russian roulette* approach is used. At every bounce a random value is generated. If the value is above a certain threshold the ray is terminated. Since only rays from the eye into the scene are traced, some light paths are very unlikely. This makes effects like caustics very unlikely and aliased. To account for this issue bidirectional path tracing [Laf96] was introduced. This refinement suggests not only to trace the rays from the eye but also from the light source into the scene. The combination of the intersection points of both types of paths are used for the illumination estimation. Another drawback of classic path tracing is that it takes a large number of samples for the algorithm to converge. Especially with complex lighting situation, this leads to noisy results. Metropolis light transport [VG97] addresses this issue by first finding paths to the light source and storing them. The found paths are then sampled in slightly altered manner. This alteration makes the algorithm converge much faster in complex lighting situations.

2.3.4 Photon Mapping

Henrik Jensen proposed an approach somewhat different from previously developed techniques. Photon mapping [Jen96] is a two step algorithm for approximating the rendering equation that can handle transparent objects and especially caustics very well. Methods like ray tracing shoot rays from the eye into the scene. In contrast, photon mapping starts by tracing particles sent from the light sources into the scene.

The first step of the algorithm is the computation of a photon map. From every light source photons are sent into the scene. When a photon hits a diffuse surface, it is stored, and a new one, which is altered due to the surface properties, is emitted according to a density distribution function. If a specular surface is hit, the photon is reflected (or refracted) without being stored. Therefore only hits of diffuse materials are stored. A second map called *caustics map* is used to store only photons which have been reflected or refracted once. When generating this map, photons are sent especially in the direction of specular objects to speed up the computation and to produce better results. In the second step the information from the photon and caustics map is used by a ray or path tracer to compute the final image.

The information stored in the two maps cannot be used directly since the photons are distributed too sparsely, which would result in a very noisy image. Instead a disk around the intersection point of the ray is sampled and the indirect illumination is calculated based on those photons. The calculation is the sum of the energy divided by the area of the sampled disk.

2.4 Super-resolution

Enlarging images is a task which is present in many fields and applications. Simply upscaling and interpolating the information results in blurry images, which is not what we want. The standard approach to increasing the spatial resolutions would be to build sensors with a higher pixel density or larger sensors for the capturing devices. Super-resolution (SR) is an algorithmic approach that combines information from several images or from large image databases into a true or at

least a plausible high resolution version of the source image(s). In this context plausible means that feature correspondences from other images (not the same scene) are used to enhance structures, giving a visual appealing result, which does not necessarily resemble the real scene in every detail.

An assumption of early SR algorithms was that multiple images of the same scene are available, which have a sub-pixel resolution shift between them. If the images were shifted by integer values, no new information would be present. To model the SR problem the following formulation was proposed by [EF97] to relate a high resolution (HR) image to multiple low resolution (LR) images.

$$y_k = DB_kM_kx + n_k \tag{2.28}$$

In this model, y_k is the kth LR image, which is the HR image x multiplied by a warp matrix M_k, a blur matrix B_k, and a subsampling matrix D_k. n_k defines a noise vector. Therefore, the LR image y_k is a subsampled, blurred, noisy and shifted version of the HR image. The restoration process of the HR image is the inverse procedure of the described model. Different approaches have been proposed for solving this problem. On the following pages we want to give a brief overview over the main techniques, based on surveys [PPK03, SG12]. Afterwards we describe methods, which are closer related to our system, in detail.

Based on the general SR formulation in equation 2.28, different approaches in the spatial and the frequency domain have been proposed. Three fundamental steps of those algorithms are estimation of motion between the images, image registration and deblurring. Not all algorithms perform every step.

Frequency Domain Algorithms in the frequncy domain exploit the aliasing of LR images to reconstruct a HR image. Huang and Tsai [HT84] presented an algorithm that built upon the relative motion between the LR images. The method utilizes the aliasing relationship between the continuous Fourier transformation of the HR image and the discrete Fourier transformation of the LR images. Kim et a. [KBV90] published an extension to this method that could handle blurry and noisy images.

Rhee et al. [RK99] used the discrete cosine transformation instead of the discreet Fourier transformation, since it is less expensive in terms of computational power.

Spatial Domain In the spatial domain typically used estimators are least squared error and maximum a posteriori. Since the SR problem is an optimization problem, regularization terms have to be added to get satisfying results. A constrained least squared error estimator [KSKH91] is defined as

$$\sum_{k=1}^{p} ||y_k - W_k x||^2 + \alpha ||Cx||^2 \tag{2.29}$$

where k as the number of LR images, and W_k is a matrix defining the shifting, subsampling and blurring. C is a high pass filter and α the regularization factor. The regularization is based on the assumption that most parts of the image contain low frequency data, and only small parts contain high frequency data. Therefore, high frequency components are penalized in order to get smooth images. Komatsu et al. [KIAS93] formulated an maximum a posteriori estimator as

$$\hat{x} = argmin \left[\sum_{k} |y_k - W_k \hat{x}|^2 + \alpha \sum_{c \in S} \varphi(x) \right] \tag{2.30}$$

where φ is a potential function used for regularization defined on the neighbourhood c.

Example-based techniques (also called image hallucination) [FPC00, FJP02, BK00] try to model the relationship between LR and HR images using corresponding patches of LR and HR images that do not necessarily show the same scene. The correspondence between those pairs is learned from a large database and used to synthesize a plausible HR image. The main difference between the classical SR methods and example-based ones is that members of the former category try to combine information from different views of the same scene, while members of the latter one assume that the high frequency information is available in the learned database.

Glasner et al. [GBI09] presented an algorithm that combines both categories. The idea is that structures appear redundantly in natural images. If those structures are shifted by sub-pixel offsets, classic SR approaches can be utilized. If the structures appear in different scales, example-based ones can be used. Exploiting this observation, a HR image can be computed from a single LR source image.

Bhat et al. [BZS+07] proposed a system similar to ours that uses static images to enhance a video of the same scene. The method computes view-dependent depth data for the images and the video and uses this information to apply a variety of effects including super resolution.

Another interesting work has been presented by Schubert and Miko [SM08]: Super resolution of a video is computed using a combination of large image databases and high resolution still images of the same scene taken in a large interval. Internet image databases are incorporated to retrieve images, for example, of a famous building which is shown in the video. To match the images, features are extracted, and a homography is computed using the RANSAC algorithm. A binary mask for the overlap is calculated, which is used in the next step to decide if a generic edge preserving prior should be used or if the solution should be close to the high resolution prior.

In contrast to classic super-resolution techniques, we aim to design a algorithm, which can be executed online at least in the near feature. Therefore, we cannot perform computationally complex search operations. Also the boundary conditions for our system are different, since we, for example, cannot look in the future to access additional information, which is possible and common in offline super-resolution algorithms.

2.5 Render caching

The idea of render caching is to reuse information from previous frames to speed up the computation of the current one. When the frame rate is high, the changes from a previous frame to the current one are small and therefore, the temporal coherence is high. This information can then directly (e.g., color of a pixel) or indirect (e.g., intermediate results) be used. The term *render cache* was introduced by Walter et al. [WDP99], who used it as a data structure to speed up rendering of otherwise none interactive methods. The information from the cache is utilized by re projecting it into the new camera space.

The reprojection can be realized in two directions. First, the data can be reused by forward projecting the cache into the current view. Since not every pixel has a one-to-one equivalent in the novel view, holes and resampling artefacts appear. Yu et al. [YWY10] proposed a GPU implementation of the forward reprojection algorithm, which uses a per pixel disparity vector to compute the new position. Implementing this approach is difficult and can be computational expensive. Didyk et al. [DER+10] proposed an efficient method, which fits a coarse, regular mesh grid to the cached image. The mesh is aligned to the depth discontinuities of the cache. Each vertex is warped to its new position using the cached image as texture. Holes are avoided by stretching of the mesh and visibility is resolved by fold overs.

Reverse reprojection goes the other way around and computes the screen space position using the geometry of the scene. With this information and the depth per pixel, the color can be looked up in the reverse projection cache [NSL+07, SJW07]. If no cache entry is found, the shading has to be computed from scratch. The advantage of forward reprojection is that the geometry does not need to be processed in order to perform the lookup. Since there is no one-to-one mapping for every pixel, holes an resampling artefacts (cracks) appear. With reverse reprojection, holes only appear if the information is not given in the cache (disocclusion), but with the additional cost of processing the geometry for the lookup.

In both cases holes, mostly from disocclusion, are an issue. This artefacts can be removed by inpainting using the information of neighbouring pixels like proposed by Andreev [And10]. This step can be repeated until all holes are filled. A more advanced way of hole filling is push-pull interpolation [MKC07]. In the first step, an image pyramid is built and holes are filled from the neighbouring pixels of the next finer level. The second step traverses the pyramid in the inverse order by filling holes from the coarser levels.

Another importation aspect of render caching is when to refresh pixels in the cache. If a pixel is reused over many frames, the reprojection error accumulates resulting in noticeable artefacts. Scherzer et al. [SJW07] proposed three strategies for updating the cache by uniform tiling of the screen, random refresh regions and interleaved refresh regions. Besides directly reusing the data from the cache, a running average can be used to combine samples from previous computation, which is called *amortized sampling*.

The applications for render caching are numerour. Sitthi-Amorn et al. [SaLY+08] proposed an algorithm for the acceleration of pixel shader computations by directly reusing information from the cache whenever available. Antialiasing can be achieved using a higher-resolution running average cache, like proposed by Yang et al. [YNS+09]. Scherzer et al. [SSMW09] proposed an algorithm, based on the idea of amortized sampling, to render soft shadows. An interesting work on spatio-temporal upsampling of renderings was published by Herzog et al. [HEMS10]. By combining multiple low resolution renderings, using a modified joint-bilateral filter, a high-resolution image can efficiently be computed. The survey of Scherzer et al. [SYM+11] gives a good overview of the state of the art methods on using temporal coherence in real-time rendering.

Chapter 3

Approach

Contents

In this work we present a system that is capable of increasing the overall quality of an augmented reality setup. Quality in this context is defined as the spatial resolution of the video stream and the realism of the augmented information. While spatial resolution can be measured, the quality of the augmented data is less well defined and depends on the scenario. Therefore, instead of focusing on a particular rendering method, we describe a setup that can be used with different techniques.

Rendering images in high resolution and quality is a computationally expensive task. We investigated how we can speed up the rendering process to make realistic rendering algorithms usable in interactive setups for simple scenes. We found decoupling of the rendering process from the displaying frequency in combination with image-based rendering a suitable technique with a good trade off between quality and speed.

Computing a high resolution version of a low resolution video stream using high resolution still images is closely related to the field of super resolution. We try to approach this problem from another direction by combining the advantages of different sensors or sensor settings.

In the following subsections, we first want to give an overview of the system with its constraints and assumptions. We introduce the two major parts, which have been described briefly in the previous paragraphs, and describe the encountered issues and difficulties. Finally, we present the main aspects of our system.

3.1 Overview

Our system consists of two major parts which can be viewed separately. The first part is the improvement of the spatial resolution of the video stream using an additional sensor that captures high resolution still images at a slow frame rate. This part is called *super resolution* and is described in section 3.2. The second part deals with the augmentation of the scene. High quality renderings computed at a slow frame rate are used as augmented information. This part is called *information augmentaion* and is treated in section 3.3.

In both subsystems, we have the issue of different frame rates: the rate at which we render the augmentations and capture high resolution images and the rate at which the display is refreshed. The system in general is driven by the display frame rate. The challenge is to overcome the described divergence. Figure 3.1 illustrates the main elements of the system and the steps involved in our pipeline.

Augmented reality usually involves tracking. For our system, we use a tracking library based on OpenCV*. Since tracking is not in the scope of this work, we will not go much into detail. The same applies to optical flow estimation used in the image registration process. We utilize libraries from different sources, but do

*http://opencv.org/, retrieved on November 06, 201

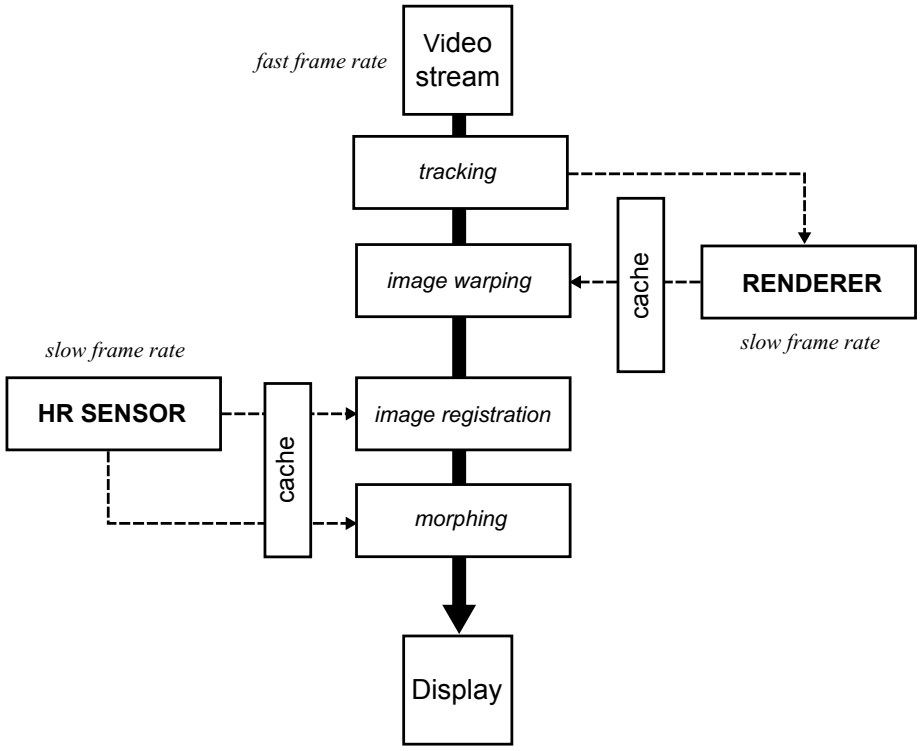


Figure 3.1: A schematic overview of our system.

not present any novel optical flow algorithms.

3.2 Online Super-resolution

A high spatial resolution is a property which is desirable for many applications. High resolution means more information and therefore more details. Charge-coupled devices (CCD) and complementary metal oxide semiconductor (CMOS) sensors are usually used in digital image acquisition. The traditional way of increasing the spatial resolution of those sensors is by either increasing the pixel density per unit or the sensor size. Increasing the pixel density means that the area of each pixel gets smaller, which means less available light which further results in noise. This approach is therefore only a limited solution. Enhancing the spatial resolution

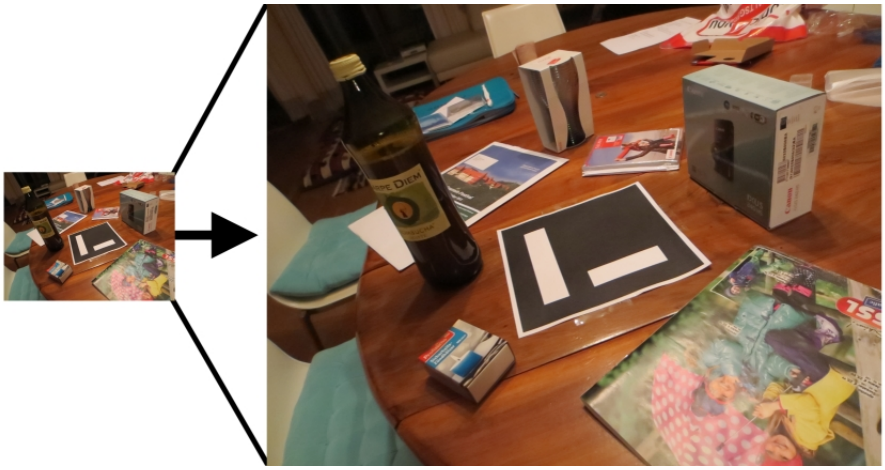


Figure 3.2: Using our method we can upscale a video taken at 640x480 pixel and 30 Hz to 2304x1728 pixel using still frames captured at 5 Hz within 200ms on average.

by building larger sensors also has its boundaries. Larger sensors mean larger capacitance, which makes it harder to speed up the charge transfer rate [KAIS93].

Super-resolution approaches this task from the algorithmic side. The classic methods use multiple images or are example-based. As already mentioned in section 2.4, the boundary conditions and requirements are different to classic super-resolution algorithms, since we want to execute our algorithm online in the near future. We present an approach that combines the characteristics of two sensors an can be applied to different hardware setups. The first sensor provides a video stream in a low resolution but at a high frame rate (e.g., 30 Hz). A second sensor, which is placed right next to the first one, captures still images in high resolution at a slow frame rate. The sensors are placed next to each other to get a similar view of the scene. A different hardware setup would be to use *dual mode cameras*. Dual mode cameras are capable of capturing still frames at a high resolution while recording a video. Both setups can be used with our method. Section 3.2.4 states two possible hardware arrangements for our system. At the moment at which the still frame is captured, both sensors show a nearly identical images of the scene (with slight differences due to the spatial placement). Over time the images begin to diverge due to the movement of the camera or the scene. To be able to use the information from the high resolution image, we have to correct this divergence. We do this by computing the optical flow between a

subsampled version of the latest high resolution image and the current frame of the video stream. The resulting sub-pixel accurate displacement map is used to transform the HR image to match the current video frame. Our algorithm can be outlined by the following steps, where I_{LR} denotes the current video frame and I_{HR} the latest HR image.

1. Compute the optical flow from I_{LR} to I_{HR} and vice versa. The computation is done at the resolution of I_{LR}.
2. Compute the confidence for every pixel according to equation 3.2.
3. Upscale the flow field using bilinear interpolation.
4. For each pixel, lookup the color of the selected HR field using the computed flow field from I_{LR} to I_{HR}.
5. Blend I_{HR} with I_{LR} image according to the confidence map.

3.2.1 Image Registration

The first step of our algorithm is to register the images sub pixel accurate using optical flow algorithms. We found out that the quality of the image registration depends mainly on these aspects:

1. Disparity between both images
2. Depth complexity of the scene
3. Quality of the optical flow estimation
4. Overlap of the field of view
5. Illumination conditions

Disparity between the images results from spatial placement of the sensors and time difference between acquisition of the HR image and the current video frame. One can easily see that not the time difference, but the motion of the camera in this time period is relevant. Assuming an average constant motion of the video sensor, we can say that the quality drops when the time difference gets larger. This is closely

related to the overlap of the field of view. Still disparity can also result from motion of objects in the scene, while the overlap of the field of view is only related to the camera movement and the capture frequency. High depth complexity means a large number of objects which are placed in front or behind each other in the scene. When the camera moves, new areas of the scene get revealed, which are not present in the latest HR image. Obviously, this effect happens more often if many objects are in the scene and especially when the are placed at different distances to the camera, since they are more likely to occlude each other. The quality of the image registration as the last main factor is always a trade-off between speed and quality. Since optical flow estimation is an optimization problem, we have to expect that the resulting flow field contains errors. The computational complexity of the optical flow scales with the number of pixels. We decided to calculate the flow at the resolution of the video frame and interpolate the resulting flow field in order to save time. This is similar to computing a sparse optical flow and interpolation in between. Experiments show that it is sufficient to use this approach. Finally, changes in illumination can too be a difficulty for the image registration process. Depending on the formulation (e.g., brightness constancy assumption vs brightness gradient constancy assumption) of the optical flow algorithm, this issue can be handled more or less well.

3.2.2 Confidence Metric

In order to be able to compute an artefact-free upscaled version of the video stream, we have to account for the mentioned problems. In detail, this means that we have to detect occlusions and regions, where the computation of the flow field yielded erroneous results. Common metrics like *endpoint difference* [ON94] and *angular difference* [BFB94] are not suitable for our case, since they are not reference free. To establish this property, we modelled our metric based on a simple observation. If we compute the optical flow from image I_1 to I_2 and vice versa, the flow vectors should approximately be the same with inverted sign.

$$uv_{forward}(p) \approx -uv_{backward}(p + uv_{forward}(p)) \tag{3.1}$$

Figure 3.3: Visualization of the confidence map. Red shading denote regions with low confidence like, for example, the area around the bottle which has been disoccluded due to the movement of the camera.

Equation 3.1 formulates this observation, where uv is the 2-component flow vector with respect to the image position $p = (x, y)$. Based on this equation, we derived a metric which tells us whether the flow vector at a certain position is correct or not.

$$confidence = 1 - \lambda(\frac{|uv_{forward}(p) + uv_{backward}(p + uv_{forward}(p))|}{\alpha}) \tag{3.2}$$

$$\lambda(x) = \begin{cases} x & \text{if } x \leq 1 \\ 0 & \text{else} \end{cases} \tag{3.3}$$

We call this metric *confidence* and compute a map for the whole flow field as defined in Equation 3.2, where α is a weighting factor that defines how strongly the difference is penalized. The formula is based on the distance between the two flow vectors interpreted as points and ideally is zero. The confidence map is recomputed every frame. This also means that we not only have to compute the flow once, but twice each frame, to be able to compute this quality metric.

Figure 3.4: The left image shows the effects of (dis)occlusion on morphing compared to the solution using our confidence metric.

3.2.3 Morphing and Blending

To finally compute the super-resolution version of the video stream, the HR image has to be morphed. Morphing means shifting the pixels according to the computed optical flow field. This operation can be done in two directions. Forward morphing means that the flow is calculated in a way that the HR image is registered with the current video frame and the pixels ar then splatted accordingly. Splatting in general has the problem that the resulting image looks perforated due to non-uniform distribution of the pixels. The more promising way is to use backward morphing. In this case, the video frame is registered with the HR image and a lookup is performed. As a result, every pixel has associated data, and interpolation can be performed easily, which is important, since the registration is sub-pixel accurate. The final output image is a alpha blended combination of the HR image and the LR video stream according to the confidence map. By applying this approach, we can avoid artefacts that otherwise would result from disocclusion or incorrect computed flow vectors. Figure 3.4 compares two results with and without this step. Since the flow field is usually good in high frequency regions of the image, we can preserve those important details. Low frequency regions tend to yield worse results in terms of optical flow computation. Using information from the LR frame in those areas means only a negligible loss of information. In the case of disocclusion, no information is available for this region, which means blending with the LR image is the only solution.

Since we assume a constant camera motion, we initialize the optical flow computation with the result from the last frame to give the algorithm a good initial estimate of the flow field. Furthermore, our approach can be applied to static and dynamic scenes, since the optical flow can be computed in both cases. The quality of the result depends on the speed of the movement of camera and objects in the scene, since the estimation of the optical flow tends to get erroneous with fast motion. In the worst case, our algorithm falls back to an bilinear interpolated version of the low resolution video frame due to the confidence metric.

In many cases, the resolution of the output device is lower than what our approach can produce so that the result has to be scaled down again to fit the screen. To make use of the high resolution, we implemented a zooming operation which lets the user have a detailed look at subregions of the upscaled video stream. In this case the user can again benefit from the high resolution even if the output device is not natively capable of displaying it.

3.2.4 Hardware Setup

In this section we want to describe two possible hardware setups for our system. Different combinations of cameras are possible and dual mode cameras can be used too. An important factor is the arrangement of the cameras. They should be positioned as close as possible next to each other to get a very similar view on the scene. In addition, the lenses should be the same, as well as the settings of the sensors. If the imaging characteristics of the sensors or lenses differ a lot, additional difficulties are introduced for the image registration process. In that case, measures to normalize the images like, for example, removing distortion and histogram matching need to be applied. The result is then closely related to the efficiency of these measures. Table 3.1 shows a possible hardware setup using Point Gray industry grade cameras. Due to the small sizes (29x29x30mm), a close placement of the sensors is possible. Point Gray lately also produces cameras which can directly record videos in a high resolution (4.1 megapixel with up to 90 fps). Enhancing the resolution by improving the sensor chip is expensive and still limited by the factors described in the first paragraph of section 3.2. Therefore, especially

	HR Camera	**LR Camera**
Model	FL2G-50S5C-C	FL2G-13S2C-C
Spatial resolution	2448x2048	1032x776
Temporal resolution	7.5 fps	30 fps

Table 3.1: Possible hardware setup using Point Grey components

for non-industry grade cameras, we propose to use an algorithmic approach like presented in this work rather than pure hardware improvement.

Another interesting hardware setup would be a device which features the Frankencamera API [AJD+10]. This API lets the user control important parameters of the image sensor like shutter speed, aperture, ISO value etc. Such a camera could be configured to run in the described dual mode with the advantage of no initial difference between the HR and the LR image, since the sensor is in both cases the same. The hardware available at the moment is sadly not capable of executing our algorithm. Hopefully, this specification is implemented on future mobile devices.

3.3 Augmenting Information

The second major part of our work deals with the augmentation of the scene. We aimed at building a system that enriches the scene with high quality renderings. Using physically based rendering techniques, realistic images can be computed, but at a high computational expense. Since we need to render these images at a high spatial resolution to match the super-resolution video stream the complexity is even higher. If the rendering step of our system is slow, the overall frame rate of our system is slowed down drastically. Therefore, we want to decouple the rendering from the rest of the pipeline to preserve a smooth frame rate, which is necessary for user interaction and a good user experience. Decoupling the process alone does not solve the problem, since due to different frame rates temporal artefacts are introduced. We overcome this issue by using image-based rendering techniques. Since we have easy access to per-pixel depth information, image warping can be used. Furthermore, we can compute a depth map of the scene, since we know the extrinsic parameters from the tracking and the intrinsic parameters from calibration.

By computing the optical flow between two selected key frames, we can solve the correspondence problem. With the depth information, we can simulate occlusion of the virtual objects by the real world.

3.3.1 Camera Calibration

In order to be able to augment the scene with renderings, we need to calibrate the camera(s) intrinsically and extrinsically. The intrinsic calibration is done using a checkerboard marker. We account for radial distortion and use an intrinsic matrix in the form of equation 3.4

$$A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

where f_x, f_y are the focal lengths, and (c_x, c_y) is the optical center. The extrinsic parameters of the camera are obtained using the fiducial marker tracking library Aruco[†]. The calibration is further used for rendering and image warping of the augmented information and for estimating the scene depth by triangulation.

3.3.2 Rendering Setup

Decoupling the rendering component from the rest of the system can be done in general using two approaches: Oversampling based algorithms like path tracing compute a high number of samples for every pixel until the algorithm converges. By computing these samples over multiple frames the workload can be distributed. This workload distribution can by done by following two different strategies: First the computation can be *time sliced*. Therefore, only a subset of all samples per pixel is computed each frame. The final image is ready after a number of frames depending on the size of the subset. A similar technique is to *spatially slice* the image by computing a sub-region of the rendering each frame with the full number of samples. Since the objects in the scene are most likely not distributed uniformly, this method results in an unsteady frame rate. With both strategies, the final image is available after a distinct number of frames depending on the splitting criterion and on the desired total number of samples. Figure 3.5 illustrates both

[†]http://www.uco.es/investiga/grupos/ava/node/26, retrieved on November 06, 201

strategies. The second approach is to execute the rendering in a second thread on

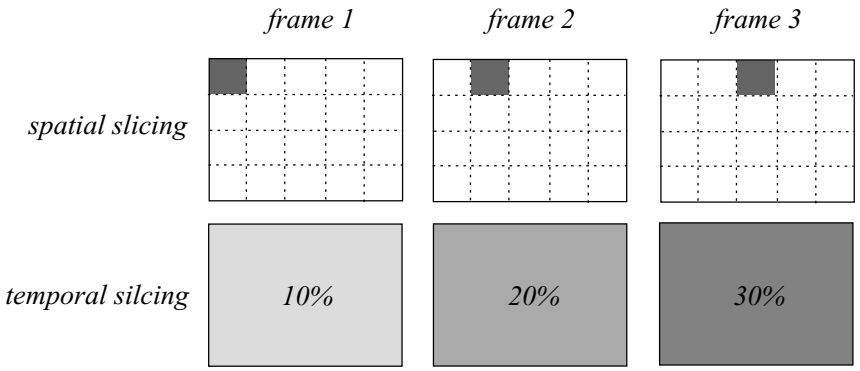


Figure 3.5: Spatial versus temporal slicing. In the upper row the shaded rectangle denotes the region which is computed at frame n. In the lower row the shading denotes the percentages of samples per pixel which have been computed to this point.

a different GPU. This decouples the process completely, but requires additional hardware. Also a barrier is needed to copy the computed results back to the main GPU. In this case, the workload is completely shifted to another dedicated rendering node. Both approaches have their advantages and drawbacks. Using a second GPU is a strong hardware constraint, which will currently not be met by any mobile device and therefore would limit the applicability of this approach. We decided to use the time slicing strategy which does not decouple the rendering process completely, but provides a stable frame rate and does not introduce any further constraints. In all cases we have a multi-frame rate setup since the frequency at which the rendering is computed differs from the displaying frequency.

For our experiments we implemented a path tracer to augment the scene. This algorithm is just an example of the rendering methods that can be used. Since image-based rendering, in our case image warping, only relies on per-pixel colour and depth information, essentially any rendering technique from classical rasterization to volume rendering can be used. The workload distribution strategy has to be chosen accordingly.

3.3.3 Image-based Rendering

To overcome the difference in frame rate, which results from decoupling the rendering process, we utilize image-based rendering techniques. Per pixel depth information can easily be stored during the rendering process. In combination with the matrices used for rendering and the current camera matrix, we can re-project every pixel to derive a novel view of the scene. Some issues related to this approach have to be addressed in order to produce satisfying results. The first issue is one we have already encountered with super-resolution of the video stream. Since image warping uses a static image from a different viewpoint than the current, disocclusion is again a topic. A common way to address this issue is to cache or even render different views in a preprocessing step, as described in [HKS10]. In addition to the latest rendered image, an appropriate stored view can be used to fill disoccluded areas. Since, in our case, storing the rendered images would require a vast amount of memory, we decided to abandon this approach.

Not only disocclusion, but also occlusion is a problem, which needs to be handled. If parts of the rendered object occlude other parts due to the change in camera position, different pixels are warped to the same location. This results in a *depth fighting* like behaviour. Another problem that comes with this approach is sampling related. If the camera in the derived view is closer to the scene than in the original one, the image is sampled at a higher rate, leading to cracks in the novel view.

The straightforward way for implementing image warping would be to multiply each pixel and its depth position by the inverse projection and model matrix used for rendering and then again by the projection and model matrix of the current view like given in equation 3.5 where P is the projection matrix, C_{ref} is the camera matrix used for rendering, $C_{current}$ is the camera matrix of the desired view and p is the position of the pixel given in homogeneous coordinates.

$$p' = P \cdot C_{current} \cdot C_{ref}^{-1} \cdot P^{-1} \cdot p \tag{3.5}$$

The problem that comes with this approach is that it is likely that different pixel are projected onto the same output position. Since we implemented the warping on

Figure 3.6: Comparison of the two hole filling methods for different warping angles. The first row shows the warped image without hole filling, the second with color interpolation and the third with re-rendering of disoccluded areas.

the GPU this results in undefined behavior. We implemented the image warping in the following way to resolve the described problems: In a first step, only the depth value of every pixel is warped into the desired view. We resolve the described race condition by using atomic operations to store only the depth values closest to the camera. This is similar to the depth buffer algorithm. In the next step, the depth and position of each pixel is used to look up the color in the original image. This is done by performing the operation given in equation 3.5, in the reverse direction. Using this multi-stage warping we can efficiently eliminate especially resampling artefacts and also small holes which result from disocclusion.

In the final step, we aim to fill small holes and cracks by interpolation. This can be done by either interpolating depth or color values. Interpolating between depth values is in general not a good idea. If the used samples belong to different depth levels (e.g., front and background objects) the resulting value has no defined meaning. Therefore, we interpolated the color values after the final step. The result is again not correct if interpolated across depth discontinuities, but in general the error is hardly perceivable.

Another approach is to re-render the disoccluded areas to fill the holes an cracks. To detect disoccluded regions, we first render only the silhouette of the model from the current viewpoint. Second, we warp the image and subtract the result from the silhouette which gives us the disoccluded region. Now, the path tracer is instructed to re-render only the identified area with a small number of samples in order to keep the computational expense low. Thereby, we can efficiently fill up all holes with the disadvantage of discontinuities between the original rendering and the re-rendered area due to the small sample count. Figure 3.6 shows a comparison of the two hole filling methods. The color interpolation is applied iteratively to close larger gaps. The number of iteration depends on the size of the revealed area. The re-rendering produces good results in any case, while the computational expense is again related to the size of the disoccluded area and is way higher compared to interpolation. For the last column of the shown scene in Figure 3.6, the re-rendering took 15 ms on average while the interpolation took 0.5 ms on average.

3.3.4 Depth Estimation

With the camera extrinsics given from the tracking and the intrinsics from the calibration we further need correspondences to be able to reconstruct depth by triangulation [Fau93]. Using the optical flow we can compute pixelwise point correspondences. Having all this information at hand, we now can calculate a depth estimation of the scene. Figure 3.7 illustrates the geometric base of this algorithm. Using the current video frame and an older video frame with a certain delay as key frames for the estimation, we can compute the depth with the algorithm outlined by the following steps:

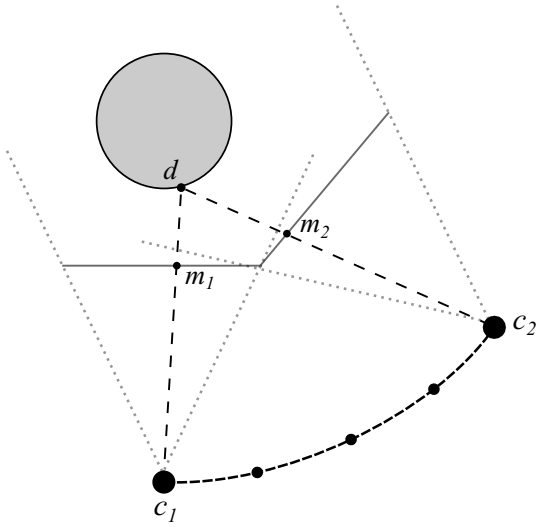


Figure 3.7: Estimating scene depth by triangulation. c_1 and c_2 denote the center of projection of the selected views while m_1 and m_2 denote corresponding points on the respective image planes.

1. Compute a ray from the center of projection into the scene for both views. This ray is given by $w = c + \lambda Q^{-1}m$, where $m = [u, v, 1]^T$ is a point on the image plane and Q is defined like shown in equation 3.6. The center of projection is given by $c = -Q^{-1}\tilde{q}$.

$$P = A[R|t] = \left[\begin{array}{c|c} \mathbf{q_1} & q_{14} \\ \mathbf{q_2} & q_{24} \\ \mathbf{q_3} & q_{34} \end{array}\right] = [Q|\tilde{q}] \tag{3.6}$$

   P_1, P_2 as well as c_1, c_2 are obtained from associated extrinsics $[R|t]$ and intrinsics A of the chosen key frames.

2. Intersect the rays. Since rays usually do not intersect at a point in $\mathcal{R}^3$ we need to computed the shortest line between the rays. If the length of that line is below a certain threshold it is treated as intersection.

3. From the intersection point (or the starting point of the shortest line between both rays) we can retrieve the reconstructed depth of the scene in that particular point.

We perform this step for every pixel in the image, which gives us a dense depth approximation. Using this information, we can now let real objects in the scene occlude augmented virtual objects to generate a more immersive AR experience. This is done using an CUDA kernel which compares the depth of the virtual scene with the estimated depth of the real scene and blends the images accordingly. Figure 3.8 shows an exemplary result. The selection of the pair of input images is essential

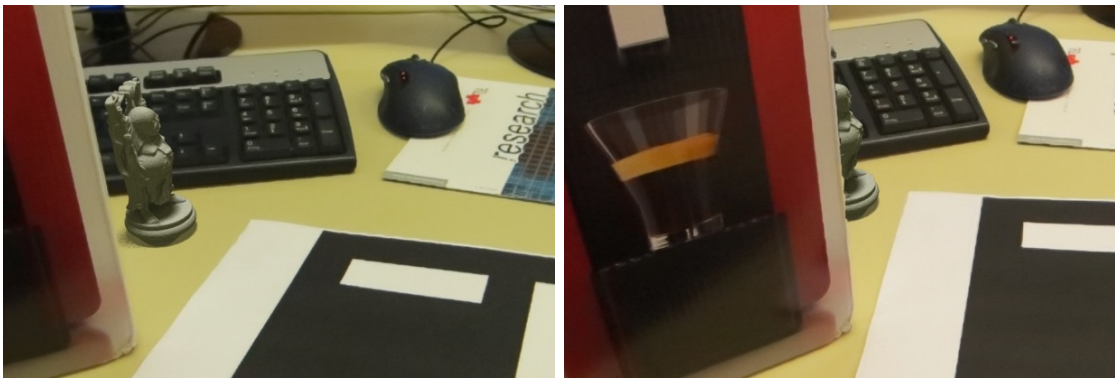


Figure 3.8: Occlusion of a virtual object (Buddha statue) by the real world using the estimated depth information.

for the depth estimation. To obtain good results, the images must have a appropriate stereo baseline. The result further depends on the correctness of the found correspondences (image registration). We need to compute the optical flow between the latest HR image and the current LR image every frame for our super-resolution algorithm. Therefore, we already have correspondences, which could be used for the depth information. Whenever the HR image is refreshed, it is nearly the same as the current image. In this case, the stereo baseline vanishes and we cannot use it for depth estimation. Therefore, we cache the latest N frames of the LR video stream and use one out of it for the stereo matching. The image from the cache is selected either with a fixed frame distance or using an angular threshold. For both cases this method fails, if the camera movement stops. To resolve this issue, a keyframe based approach should be used in the future. Unfortunately, we also have to compute the optical flow for the selected pair of images and can not use the flow field from the super-resolution step. The estimation of the optical flow is the computationally expensive part (70 ms) of this step compared to the triangulation (1 ms).

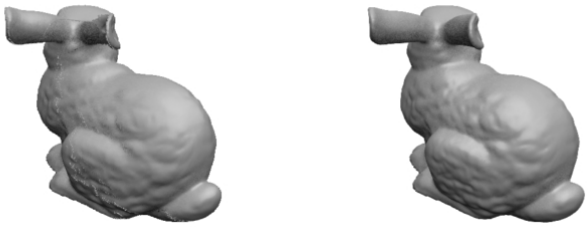
3.4 Limitations

In this section, we want to describe the boundaries and limitations of our system which arise through our assumptions and used techniques. The super-resolution part of our work is applicable on static and dynamic scenes. It has to be kept in mind that fast motion of both camera and objects in the scene leads to bad image registration results (see Figure 3.9). Due to our flow quality measure, the algorithm does not fail but rather falls back to the LR video stream. In case of fast motion the fall back might only be hardly noticeable due to motion blur. In the case of slow motion results show that high frequency details can be preserved nicely by our algorithm.

Figure 3.9: Fast motion in the scene. The red areas denote regions where the image registration failed due to fast motion. The fallback to the LR image avoids most artefacts, while some are still visible (e.g, around the yellow and pink glue tube). Results are display in the upper row, confidence map per frame in the lower.

While techniques are known to combine image-based rendering and animations [HE12, KAO+01], we limit our system to static objects. By associating transformation matrices to every pixel, simple affine transformations can be implemented. Still the results are expected to be chastening since effects like disocclusion and resampling issues would be intensified.

While image-based rendering in general is a very versatile technique, a limit is encountered, when it comes to view dependent effects. Effects like specular lighting, refraction and parallax depend on the position of the viewer relative to the scene. With our technique those effects cannot be simulated, since all information is baked into the rendering. Therefore, the quality of the result drops dramatically, if view dependent effects are simulated. A way to address this issue would be to use g-buffers [ST90] to store per pixel normals. Using this information, certain effects can be approximated in the image warping step. If, for example, refraction is simulated in the rendering, the normals can be used to calculate the reflected ray and to perform a lookup in an environment map. Figure 3.10 illustrates this issue.



(a) image warped (b) reference

Figure 3.10: View depended effects cannot directly be simulated with image warping. The effect is baked into the rendering which was computed from a different viewpoint than the current one. Figure (a) shows the image warped result of specular highlights in comparison to a reference rendering (b).

The same problem arises for the upscaling of the video stream. View dependent effects are baked into the HR image and cannot be simulated since no

geometry information of the scene is available. In most of these cases, the image registration fails and our algorithm falls back to the LR video stream for those regions of the image, but in some cases artefacts are introduced (see Figure 3.11). An example for this case are subtle reflections or refraction on objects. This results in small, but noticeable artefacts on objects featuring such material properties. We uploaded a video[‡] to Youtube, showing sources of artefacts in the super-resolution process.

(a) result (b) confidence map

Figure 3.11: Specular reflections introduce small but noticeable artefacts in the super-resolution process. Our metric can detect those regions (marked with red color) not perfectly. Therefore, small artefacts still occure. Figure (a) shows the result and figure (b) the computed confidence map.

Finally, the depth estimation is also limited to static scenes. Since we compute the depth from keyframes of different points in time, movement of objects yields false results. The stereo setup display in figure 3.7 is only valid if objects remain at the same position.

[‡]https://www.youtube.com/watch?v=r4EXwsuYKKY

3.5 Summary

In this chapter we presented a augmented reality system that enhances the overall quality of the output. The system consists of two major parts. The first part deals with super-resolution of the input video stream, while the second one addresses the quality of the augmented information.

The spatial resolution of the video stream is improved by registering high resolution images captured at a slow frame rate with the current image of the video stream. By morphing the HR image accordingly, we can produce a super-resolution version of the video stream in a decent quality. By introducing a quality metric for the estimated optical flow field, we can detect and address issues like disocclusion and artefacts resulting from large disparity between the images.

Furthermore, we augment the scene with high quality renderings by using physically based rendering methods. To be able to use this computational complex methods, we described strategies to decouple the rendering component from the rest of the system and to distribute workload over multiple frames. As a result, when the computation of an image is finished, it is already outdated and most likely does not match the current view of the scene. We employ image-based rendering techniques to overcome this gap by deriving a novel, matching view using associated depth information from the latest rendering. Thereby, we can close the gap between the rate at which the display is refreshed and the rate at which the augmentation is computed. Challenges like disocclusion and resampling problems arise from this approach, which can be addressed using a hole filling technique and a multi stage warping process.

Finally, we estimate the depth of the scene by triangulation from two key frames using the optical flow to solve the correspondence problem. Using this information we can occlude virtual objects with real objects to create a more convincing augmented reality experience.

Chapter 4

Evaluation

Contents

In this section, we present the results that we can achieve with our system. Our super-resolution approach is compared to a ground truth and to simple bilinear interpolation. We also evaluate the effect of different upscaling ratios and HR capture rates. Finally, we benchmark our image-based rendering technique used for overcoming the frame rate gap.

4.1 Testing Environment

To evaluate the performance of our system, we had to create data to use since the required hardware (like for example described in section 3.2.4) was not available to use. Therefore, we used a standard consumer photo camera, which is capable of taking pictures at a resolution of 4 mega-pixels with around 8 frames per second, to synthesize test data. We captured a scene by moving the camera very slow and then replay the recording three times faster. Using this method, we could acquire ground truth videos with a resolution of 4 mega-pixels at 24 frames per second.

We used three different scenarios to evaluate the aspects of our system. Two of the data sets show office scenes while one shows an outdoor scene. All scenes have different complexity in terms of depth, depth range and texture. Figure 4.1 shows images of the three used scenes. The garden scene has a high depth complexity, but

(a) Garden (b) Living room (c) Office

Figure 4.1: The three scenes used for benchmarking our system.

is rich of repeating patterns (leaves) and features slight movement of the leaves in the background. The living room scene has the highest depth complexity and depth range with multiple occluding objects and shows some homogeneous regions (table). Finally, the office scene has very little depth complexity, since it is mostly planar and shows a large structure with a high frequency texture. Each scene has different requirements, which we want to evaluate. For the evaluation of the augmentation part, we use the three models shown in figure 4.2. The images are rendered using a path tracer which is capable of simulating direct light, ambient occlusion, color bleeding and soft shadows. We limit our evaluation to view independent rendering effects according to the explanation given in section 3.4.

As metric for similarity, we use the structural similarity index (SSIM) [WBSS04] and HDR-VDP-2 [MKRH11], since it is consistent with human eye perception. We use this metric to show that the results of our algorithm are similar to the reference image and free of artefacts. It can be seen in the following sections that the SSIM and the HDR-VDP-2 metric is also high for the bilinear interpolated results since blurriness has no strong impact as long as the structural similarity is good. Therefore, we use the sharpness metric LPC-SI [HWS10] to show that our results are close to the sharpness of the ground truth. The values for the SSIM metric range from -1 (no similarity) to 1 (same image), from 0 (no similarity) to 100 (same image) for

(a) Buddha (b) Bunny (c) Dragon

Figure 4.2: The three models used for the evaluation of the augmentation quality.

HDR-VDP-2, while for the reference free LPC-SI metric a higher value means more sharpness. Our hardware setup is a Intel Core i5 running at 3.4 GHz with 8 GBs of memory and a Geforce GTX 680. For capturing the test data we used a Canon IXUS 240 HS.

4.2 Results

In the following sections we describe the different evaluation scenarios, show the results and give interpretations on them.

4.2.1 Super-resolution

In this section we present results for the super resolution part of our work. First, we compare our results to the ground truth and to an upscaled (using bilinear interpolation) result. Furthermore, we evaluate the effect of different HR frame rates and upscaling ratios and, finally, we give interpretations and discuss the results. We executed our method using the TV1 algorithm from flowLib* for the image registration. The input video stream had a resolution of 640x480 pixel (0,3 megapixels) at 30Hz. The HR still images had a resolution of 2304x1728 (4 megapixels) pixels, which is also the resolution of

*http://gpu4vision.icg.tugraz.at/index.php?content=subsites/flowlib/flowlib.php, retrieved on November 06, 2013

the output. Still frames were captured at a frequency of 8 Hz. The movement of the camera was smooth and steady and a combination of rotation and translation.

Figures 4.5, 4.6 and 4.5 display the results for the three test scenes. In each case, the SSIM and HDR-VDP-2 is high, which indicates a high similarity and no artefacts. But also the SSIM and HDR-VDP-2 for the bilinear interpolated image is high, since blurriness is taken only slightly into account by any metric that is based on the perception of the human eye as long as the structural similarity is high. In contrast the results for the LPC-SI metric display the strength of our algorithm. The sharpness of our result is close to the sharpness of the ground truth and way above the results of the bilinear interpolated version. Figure 4.3 shows

Figure 4.3: Impact of different upscaling ratios and HR frequencies on the quality.

the impact of different upscaling ratios and HR image capture frequencies on the similarity to the ground truth. An uspcaling ratio of 4 means that the output is 4 times the size of the input. A HR step size of 10 means that for every 10th

LR image a corresponding HR image was available (equivalent to a HR capture frequency of 3 Hz if the LR stream is captured at 30 Hz). Also, the optical flow is computed at the size of the LR input stream. The results show that the upscaling ratio (and therefore the upscaling of the optical flow) has much more impact on the similarity than the frequency at which the HR images are captured. The reason for this is that in the case of high upscaling ratios the flow field has to be interpolated highly, which results an imprecise registration of fine image details. Also, regions taken from the LR video stream (in areas where the image registration failed) are highly interpolated and therefore show a complete lack of details. The image registration, especially for high frequent regions of the image, works well even if the difference between the images is large. Figure 4.4 shows exemplary results of our super resolution algorithm. The advantage of our algorithm is clearly noticeable from the sharpness at high frequency regions in contrast to the bilinear interpolated version. With the configuration used for evaluation, our algorithm was executed within 200ms for upsampling from 640x480px to 2304x1728px. In this case, 180 ms were needed for computing the forward and the backward optical flow (computed at a resolution of 640x480), while 20 ms were needed for data transfer, computation of the confidence map and blending. Depending on the complexity of the scene (speed of camera motion and motion of objects in the scene), satisfying results can be achieved in less time by reducing the resolution of the optical flow computation. With some adjustments, we are confident that our algorithm can be executed in real time on future hardware.

4.2.2 Augmentation

To measure the quality of our image-based augmentation, we used the three different models show in figure 4.2. For the ground truth, we rendered the registered augmentation stream in full quality (for each frame, we rendered the model with the full number of samples). To have a comparison, we rendered the augmentation stream again with the full number of samples but on a smaller resolution. The resolution was chosen in a way so that the rendering takes as much time is the rendering step in our IBR configuration. The rendering step in our system does not only consist of the image warping, but also contains the computation of the subset of samples for the next rendering. The sampling step

Figure 4.4: Exemplary results of our super resolution algorithm. The first column shows reference images, the middle our approach and the right bilinear interpolated images.

does not contribute to the current view, but when benchmarking the time of the rendering, it still has to be taken into account to have a fair comparison.

Figures 4.8, 4.9 and 4.10 show plots of the similarity measured with the SSIM metric. The results clearly state that the image-based rendering approach is superior to bilinear interpolation of a low resolution rendering. The jagged behaviour of the image warping similarity function results from the slow frequency in which the renderings are computed. Every time a new image is rendered, the similarity increases drastically while decreasing with the age of the image. The dragon and bunny model have large concave regions which, leads to self occlusion of parts of the model. This explains the slightly worse result compared to the Buddha model. The rendering of the augmented information took on average

around 150 ms, while the image warping (using interpolation for hole filling) took 0.5 ms.

4.3 Summary

The evaluation of the super-resolution part of our work shows our approach is capable of improving the spatial resolution of a video stream using high resolution still frames. We showed that we can improve the resolution by up to 360 percent without introducing noticeable artefacts, while preserving high frequency details and a high degree of sharpness. We also evaluated the effect of different upscaling ratios and HR frame rates on the quality and show that our algorithm yields good results even for difficult configurations.

The results for the augmentation part of our work confirm that image-based rendering can efficiently be utilized to enrich a scene with high quality and high resolution renderings. Using our proposed workload distribution path tracing can be used to augment scenes at frame rates close to real time for scenarios with a small number of objects.

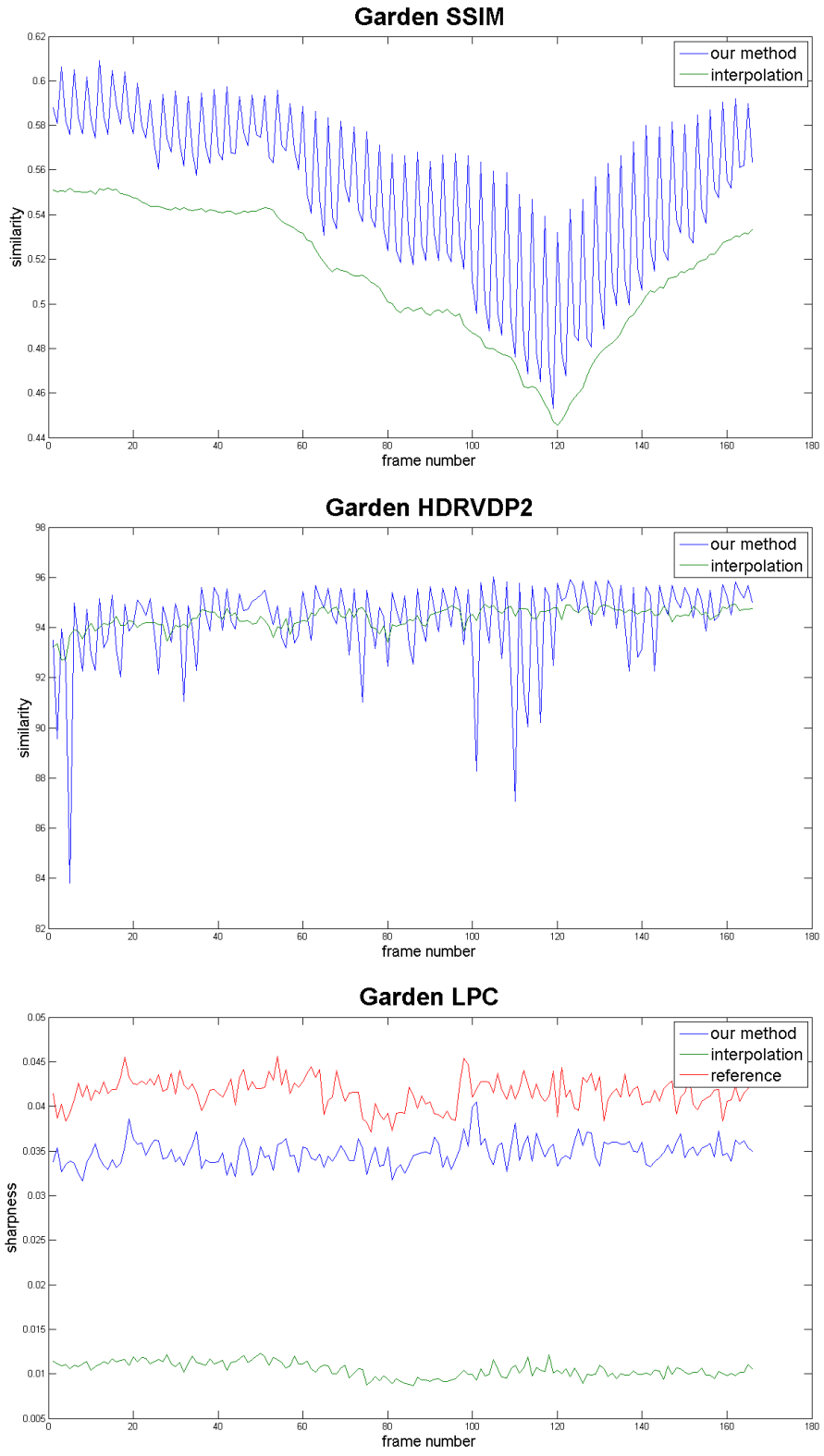


Figure 4.5: Similarity to the ground truth using the SSIM and HDR-VDP-2 metric and sharpness using the LPC-SI matric for the garden scene.

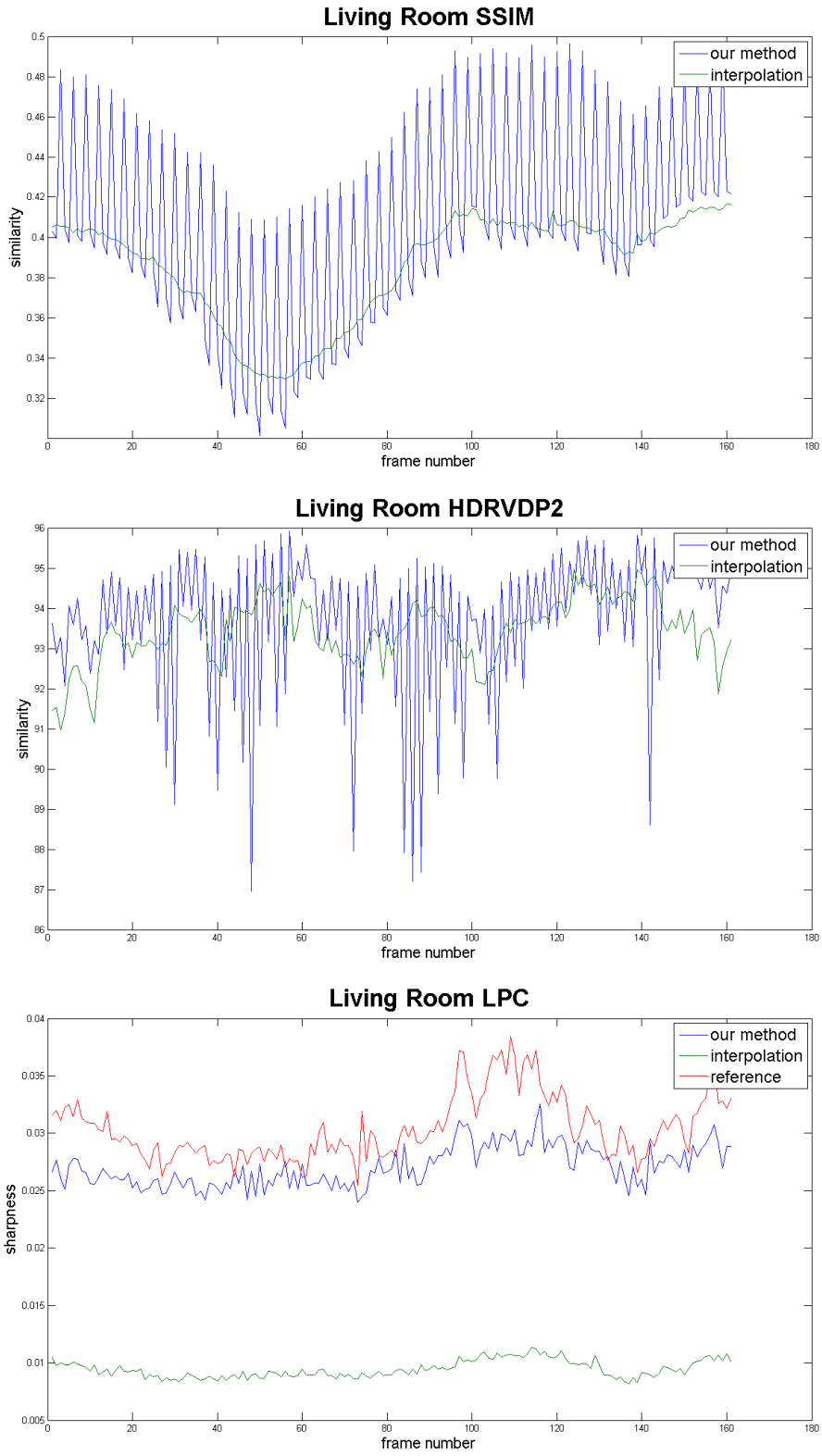


Figure 4.6: Similarity to the ground truth using the SSIM and HDR-VDP-2 metric and sharpness using the LPC-SI matric for the living room scene.

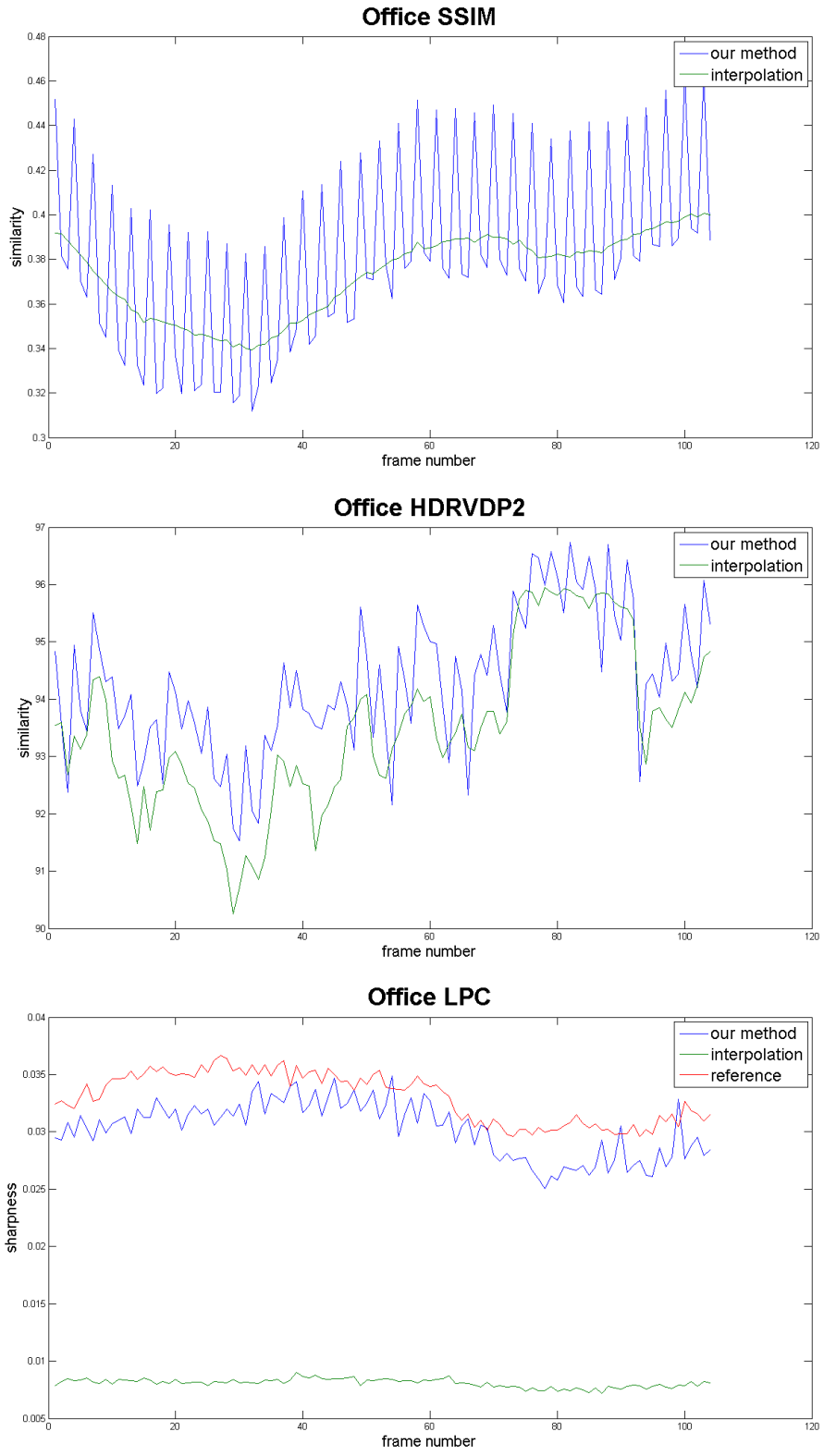


Figure 4.7: Similarity to the ground truth using the SSIM and HDR-VDP-2 metric and sharpness using the LPC-SI matric for the office scene.

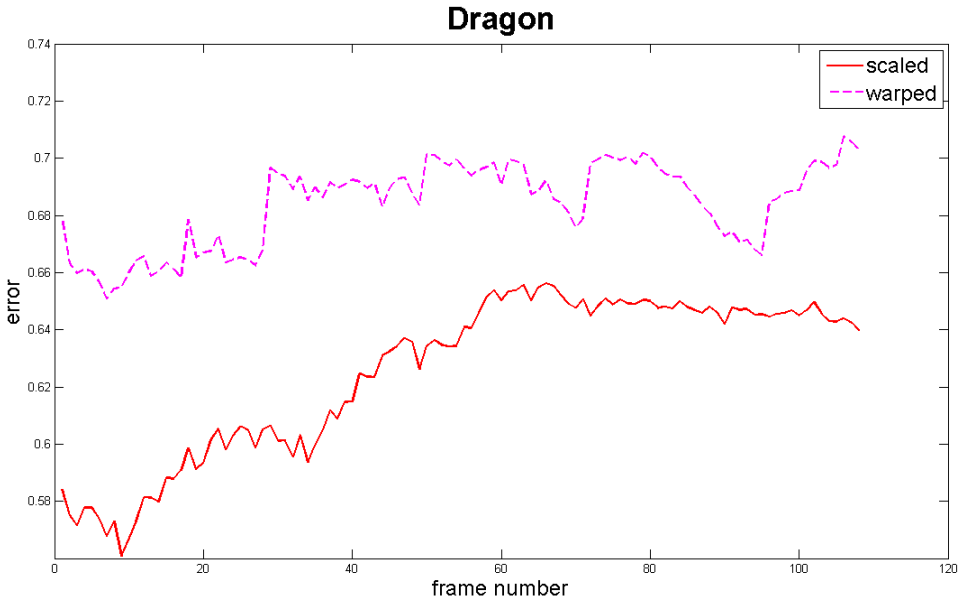


Figure 4.8: SSIM for the dragon model comparing image-based augmentation to interpolation of a lower resolution rendering.

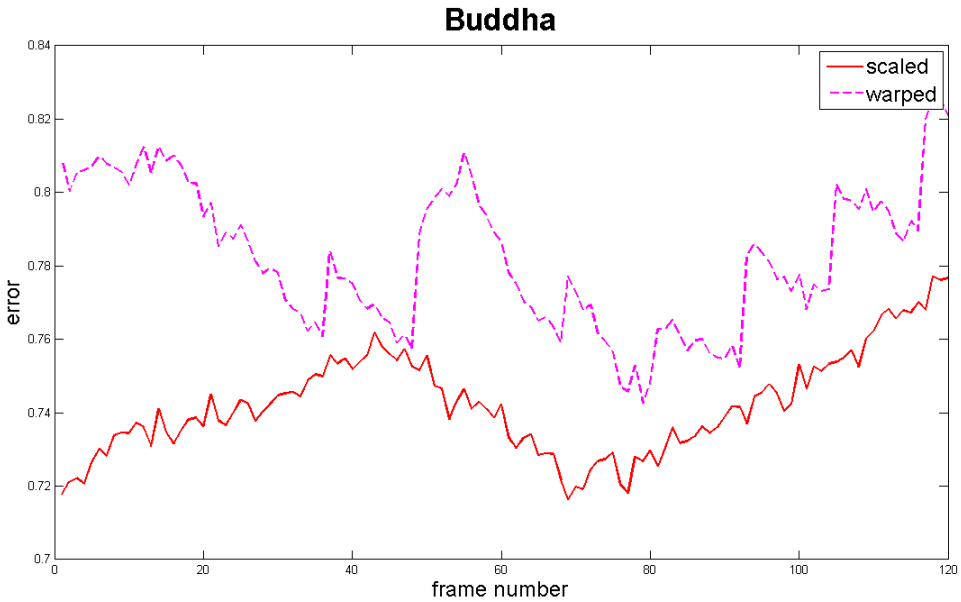


Figure 4.9: SSIM for the Buddah model comparing image-based augmentation to interpolation of a lower resolution rendering.

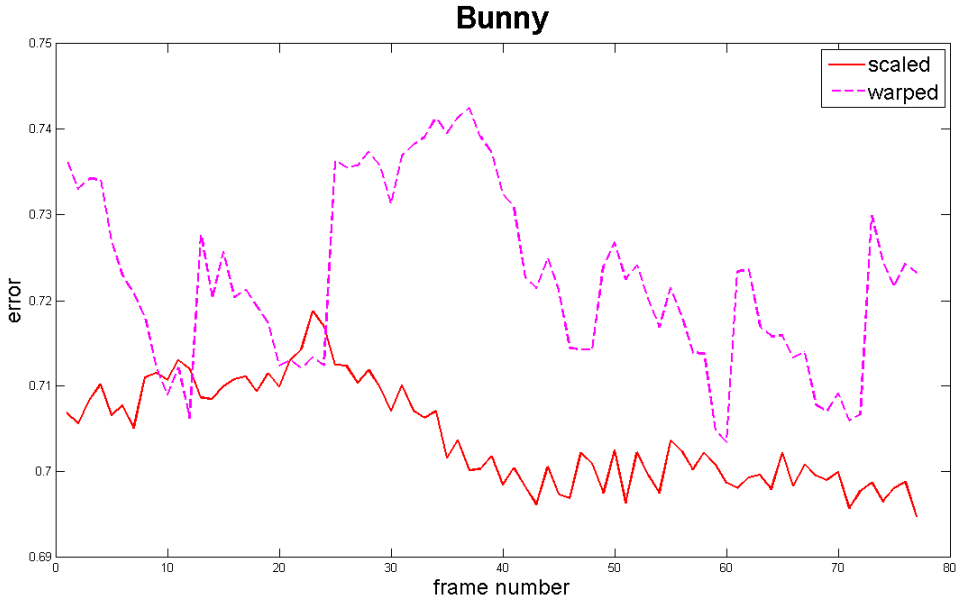


Figure 4.10: SSIM for the bunny model comparing image-based augmentation to interpolation of a lower resolution rendering.

Chapter 5

Conclusion & Future Work

We developed a system capable of improving the overall quality of an augmented reality setup. Quality in this case means on the one hand the spatial resolution of the video stream and on the other hand the visual quality of the augmented information. Still images captured at a slow frequency are used to improve the spatial resolution of the low resolution video stream. By registering the high resolution still image with sub-pixel accuracy, we can morph it to fit the current video frame. Furthermore, we designed a reference free metric for the quality of the optical flow that lets us decide whether the image registration was successful or not. Based on this metric, we blend the high resolution image with the video stream. As a result, we get a method that can upsample videos from 640x480px to 2304x1728ms within 200ms. The algorithm is failure proof, which means that in the worst case results falls back to the quality of the video stream introducing only small artefacts in extreme cases. The evaluation show that this approach can successfully be applied and yields good results in different scenarios.

For the augmentation of the scene we implemented a path tracer, which computes visually appealing and realistic results. The computational complexity of this algorithms is high, which does not allow for interactive rendering, especially if the spatial resolution is also high. We described approaches to decouple the frequency of the rendering process from the displaying frequency by distributing the workload over multiple frames. The resulting difference in frame rate is then addressed using image warping. The results show that this strategy can be applied

to move complex rendering algorithms towards interactivity without a large loss of quality.

5.1 Future Work

The sub-pixel accurate image registration step is done by computing the optical flow between the still frame and the current video frame. This has to be done twice each frame, since we need the forward and the backward flow to be able to compute our quality metric. This step is the most time consuming part of the algorithm. Since we know the camera pose from tracking, the flow computation can be simplified to a one dimensional problem by using epipolar geometry. This simplification would not only speed up the computation, but also might give better results in terms of quality, since the search space is significantly smaller. Furthermore, we aim to move the super-resolution algorithm of our work to mobile devices. Especially for this case, the image registration has to be simplified to achieve a satisfying performance.

The keyframes for the depth estimation are currently selected with a constant time delay to the current frame. As soon as the camera stops, the depth computation fails. In addition, the result strongly depends on the movement of the camera. Keyframes should be selected and stored using a criteria which takes the motion of the camera into account to receive a good quality for the depth estimation in any case.

At the moment, the augmented information only features view independent effects. To save computation time, the rendering could be done as a preprocessing step storing depth and color information. The live rendering then would be purely image-based. Another challenge ,which was already discussed in section 3.3, are view dependent visual effects like refraction, reflection and parallax. By storing additional per pixel information about the surface and the material in a g-buffer, those effects could be approximated in the warping process.

Currently, our system runs on manually created test data. Since results showed that our approach delivers satisfying results, the next step is to create a live setup. Furthermore, we plan to move our work to mobile devices. The Frankencamera specification [AJD+10] would be a good starting point for this. Sadly there is no proper hardware available at the moment that implements this specification. Investigations should be made towards identifying mobile hardware, which is capable of executing our algorithms.

Given the future availability of such a device, we are confident that multi frame-rate augmented reality can become a powerful and widespread technique to improve the visual quality of many AR applications.

Bibliography

[AB91] Edward H Adelson and James R Bergen. The plenoptic function and the elements of early vision. *Computational models of visual processing*, 91(1):3–20, 1991. Cited on page 13.

[AJD+10] Andrew Adams, David E Jacobs, Jennifer Dolson, Marius Tico, Kari Pulli, Eino-Ville Talvala, Boris Ajdin, Daniel Vaquero, Hendrik Lensch, Mark Horowitz, et al. The frankencamera: an experimental platform for computational photography. *ACM Transactions on Graphics (TOG)*, 29(4):29, 2010. Cited on pages 40 and 67.

[Ana89] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2(3):283–310, 1989. Cited on page 9.

[And10] Dmitry Andreev. Real-time frame rate up-conversion for video games: or how to get from 30 to 60 fps for free. In *ACM SIGGRAPH 2010 Talks*, page 16. ACM, 2010. Cited on page 29.

[App68] Arthur Appel. Some techniques for shading machine renderings of solids. In *Proceedings of the April 30–May 2, 1968, spring joint computer conference*, pages 37–45. ACM, 1968. Cited on page 18.

[BBPW04] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *Computer Vision-ECCV 2004*, pages 25–36. Springer, 2004. Cited on page 10.

[BFB94] John L Barron, David J Fleet, and Steven S Beauchemin. Performance of optical flow techniques. *International journal of computer vision*, 12(1):43–77, 1994. Cited on page 36.

[BK00] Simon Baker and Takeo Kanade. Hallucinating faces. In *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*, pages 83–88. IEEE, 2000. Cited on page 26.

[BWS05] Andrés Bruhn, Joachim Weickert, and Christoph Schnörr. Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61(3):211–231, 2005. Cited on page 10.

[BZS+07] Pravin Bhat, C Lawrence Zitnick, Noah Snavely, Aseem Agarwala, Maneesh Agrawala, Michael Cohen, Brian Curless, and Sing Bing Kang. Using photographs to enhance videos of a static scene. In *Proceedings of the 18th Eurographics conference on Rendering Techniques*, pages 327–338. Eurographics Association, 2007. Cited on page 27.

[CCWG88] Michael F Cohen, Shenchang Eric Chen, John R Wallace, and Donald P Greenberg. A progressive refinement approach to fast radiosity image generation. In *ACM SIGGRAPH Computer Graphics*, volume 22, pages 75–84. ACM, 1988. Cited on page 21.

[CPC84] Robert L Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. In *ACM SIGGRAPH Computer Graphics*, volume 18, pages 137–145. ACM, 1984. Cited on page 19.

[CW93] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 279–288. ACM, 1993. Cited on page 14.

[DER+10] Piotr Didyk, Elmar Eisemann, Tobias Ritschel, Karol Myszkowski, and Hans-Peter Seidel. Perceptually-motivated real-time temporal upsampling of 3d content for high-refresh-rate displays. In *Computer Graphics*

Forum, volume 29, pages 713–722. Wiley Online Library, 2010. Cited on page 28.

[DTM96] Paul E Debevec, Camillo J Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 11–20. ACM, 1996. Cited on page 17.

[EF97] Michael Elad and Arie Feuer. Restoration of a single superresolution image from several blurred, noisy, and undersampled measured images. *Image Processing, IEEE Transactions on*, 6(12):1646–1658, 1997. Cited on page 25.

[Fau93] Oliver Faugeras. *Three dimensional computer vision: A geometric viewpoint*. the MIT Press, 1993. Cited on page 45.

[FJP02] William T Freeman, Thouis R Jones, and Egon C Pasztor. Example-based super-resolution. *Computer Graphics and Applications, IEEE*, 22(2):56–65, 2002. Cited on page 26.

[FPC00] William T Freeman, Egon C Pasztor, and Owen T Carmichael. Learning low-level vision. *International journal of computer vision*, 40(1):25–47, 2000. Cited on page 26.

[GBI09] Daniel Glasner, Shai Bagon, and Michal Irani. Super-resolution from a single image. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 349–356. IEEE, 2009. Cited on page 27.

[Gib50] James J. Gibson. *The Perception Of The Visual World*. Boston: Houghton Mifflin, 1950. Cited on page 5.

[Gla84] Andrew S Glassner. Space subdivision for fast ray tracing. *Computer Graphics and Applications, IEEE*, 4(10):15–24, 1984. Cited on page 20.

[GTGB84] Cindy M Goral, Kenneth E Torrance, Donald P Greenberg, and Bennett Battaile. Modeling the interaction of light between diffuse surfaces.

In *ACM SIGGRAPH Computer Graphics*, volume 18, pages 213–222. ACM, 1984. Cited on page 20.

[HE12] Anna Hilsmann and Peter Eisert. Image-based animation of clothes. In *Eurographics (Short Papers)*, pages 69–72, 2012. Cited on page 49.

[HEMS10] Robert Herzog, Elmar Eisemann, Karol Myszkowski, and H-P Seidel. Spatio-temporal upsampling on the gpu. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, pages 91–98. ACM, 2010. Cited on page 29.

[HKS10] Stefan Hauswiesner, Denis Kalkofen, and Dieter Schmalstieg. Multi-frame rate volume rendering. In *Proceedings of the 10th Eurographics conference on Parallel Graphics and Visualization*, pages 19–26. Eurographics Association, 2010. Cited on page 43.

[HS81] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1):185–203, 1981. Cited on page 7.

[HSA91] Pat Hanrahan, David Salzman, and Larry Aupperle. A rapid hierarchical radiosity algorithm. In *ACM SIGGRAPH Computer Graphics*, volume 25, pages 197–206. ACM, 1991. Cited on page 21.

[HT84] T. S. Huang and R. Y. Tsay. Multiple frame image restoration and registration. In *Advances in Computer Vision and Image Processing*, volume 1, pages 317–339, Greenwich, 1984. JAI. Cited on page 25.

[HWS10] Rania Hassen, Zhou Wang, and Magdy Salama. No-reference image sharpness assessment based on local phase coherence measurement. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 2434–2437. IEEE, 2010. Cited on page 54.

[ICG86] David S Immel, Michael F Cohen, and Donald P Greenberg. A radiosity method for non-diffuse environments. *ACM SIGGRAPH Computer Graphics*, 20(4):133–142, 1986. Cited on page 18.

[Jen96] Henrik Wann Jensen. Global illumination using photon maps. In *Rendering Techniques' 96*, pages 21–30. Springer, 1996. Cited on page 24.

[KAIS93] T Komatsu, K Aizawa, T Igarashi, and T Saito. Signal-processing based method for acquiring very high resolution images with multiple cameras and its theoretical analysis. *IEE Proceedings I (Communications, Speech and Vision)*, 140(1):19–25, 1993. Cited on page 34.

[Kaj86] James T Kajiya. The rendering equation. In *ACM SIGGRAPH Computer Graphics*, volume 20, pages 143–150. ACM, 1986. Cited on pages 18 and 22.

[Kan97] Sing Bing Kang. *A survey of image-based rendering techniques*. Digital, Cambridge Research Laboratory, 1997. Cited on page 14.

[KAO+01] Hiroshi Kawasaki, Hiroyuki Aritaki, Takeshi Ooishi, Katsushi Ikeushi, and Masao Sakauchi. Image-based rendering for photo-realistic animation. 2001. Cited on page 49.

[KBV90] SP Kim, NK Bose, and HM Valenzuela. Recursive reconstruction of high resolution image from noisy undersampled multiframes. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 38(6):1013–1027, 1990. Cited on page 25.

[KIAS93] Takashi Komatsu, Toru Igarashi, Kiyoharu Aizawa, and Takahiro Saito. Very high resolution imaging scheme with multiple different-aperture cameras. *Signal Processing: Image Communication*, 5(5):511–526, 1993. Cited on page 26.

[KSKH91] Aggelos Konstantinos Katsaggelos, MR Schroeder, Teuvo Kohonen, and TS Huang. *Digital image restoration*. Springer-Verlag New York, Inc., 1991. Cited on page 26.

[Laf96] Eric Lafortune. *Mathematical models and Monte Carlo algorithms for physically based rendering*. PhD thesis, Citeseer, 1996. Cited on page 23.

[Len98] Jed Lengyel. The convergence of graphics and vision. *Computer*, 31(7):46–53, 1998. Cited on page 14.

[LK+81] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 81, pages 674–679, 1981. Cited on page 8.

[Low99] D.G. Lowe. Object recognition from local scale-invariant features. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1150–1157 vol.2, 1999. Cited on page 12.

[LYT11] Ce Liu, Jenny Yuen, and Antonio Torralba. Sift flow: Dense correspondence across scenes and its applications. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(5):978–994, 2011. Cited on page 12.

[MB95] Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 39–46. ACM, 1995. Cited on page 16.

[MJ97] Leonard McMillan Jr. *An image-based approach to three-dimensional computer graphics*. PhD thesis, Citeseer, 1997. Cited on page 16.

[MKC07] Ricardo Marroquim, Martin Kraus, and Paulo Roma Cavalcanti. Efficient point-based rendering using image reconstruction. In *SPBG*, pages 101–108, 2007. Cited on page 29.

[MKRH11] Rafat Mantiuk, Kil Joong Kim, Allan G Rempel, and Wolfgang Heidrich. Hdr-vdp-2: A calibrated visual metric for visibility and quality predictions in all luminance conditions. In *ACM Transactions on Graphics (TOG)*, volume 30, page 40. ACM, 2011. Cited on page 54.

[NSL+07] Diego Nehab, Pedro V Sander, Jason Lawrence, Natalya Tatarchuk, and John R Isidoro. Accelerating real-time shading with reverse reprojection caching. In *Graphics Hardware*, pages 25–35, 2007. Cited on page 28.

[o2005] Optical flow: Techniques and applications. 2005. Cited on page 5.

[ON94] M. Otte and H.-H. Nagel. Optical flow estimation: Advances and comparisons. In Jan-Olof Eklundh, editor, *Computer Vision — ECCV '94*, volume 800 of *Lecture Notes in Computer Science*, pages 49–60. Springer Berlin Heidelberg, 1994. Cited on page 36.

[PK85] Tomaso Poggio and Christof Koch. Ill-posed problems in early vision: from computational theory to analogue networks. *Proceedings of the Royal society of London. Series B. Biological sciences*, 226(1244):303–323, 1985. Cited on page 6.

[PPK03] Sung Cheol Park, Min Kyu Park, and Moon Gi Kang. Super-resolution image reconstruction: a technical overview. *Signal Processing Magazine, IEEE*, 20(3):21–36, 2003. Cited on page 25.

[RK99] Seunghyeon Rhee and Moon Gi Kang. Discrete cosine transform based regularized high-resolution image reconstruction algorithm. *Optical Engineering*, 38(8):1348–1356, 1999. Cited on page 26.

[SaLY+08] Pitchaya Sitthi-amorn, Jason Lawrence, Lei Yang, Pedro V Sander, and Diego Nehab. An improved shading cache for modern gpus. In *Proceedings of the 23rd ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware*, pages 95–101. Eurographics Association, 2008. Cited on page 29.

[SG12] Amisha J Shah and Suryakant B Gupta. Image super resolution-a survey. In *Emerging Technology Trends in Electronics, Communication and Networking (ET2ECN), 2012 1st International Conference on*, pages 1–6. IEEE, 2012. Cited on page 25.

[SGHS98] Jonathan Shade, Steven Gortler, Li-wei He, and Richard Szeliski. Layered depth images. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 231–242. ACM, 1998. Cited on page 15.

[SJW07] Daniel Scherzer, Stefan Jeschke, and Michael Wimmer. Pixel-correct shadow maps with temporal reprojection and shadow test confidence. In *Proceedings of the 18th Eurographics conference on Rendering Techniques*, pages 45–50. Eurographics Association, 2007. Cited on pages 28 and 29.

[SK00] Harry Shum and Sing Bing Kang. Review of image-based rendering techniques. In *VCIP*, pages 2–13. Citeseer, 2000. Cited on page 14.

[SM08] Falk Schubert and Krystian Mikolajczyk. Combining high-resolution images with low-quality videos. In *BMVC*, pages 1–10, 2008. Cited on page 27.

[SSMW09] Daniel Scherzer, Michael Schwärzler, Oliver Mattausch, and Michael Wimmer. Real-time soft shadows using temporal coherence. In *Advances in Visual Computing*, pages 13–24. Springer, 2009. Cited on page 29.

[ST90] Takafumi Saito and Tokiichiro Takahashi. Comprehensible rendering of 3-d shapes. In *ACM SIGGRAPH Computer Graphics*, volume 24, pages 197–206. ACM, 1990. Cited on page 49.

[SYM+11] Daniel Scherzer, Lei Yang, Oliver Mattausch, Diego Nehab, Pedro V Sander, Michael Wimmer, and Elmar Eisemann. A survey on temporal coherence methods in real-time rendering. In *Eurographics 2011-State of the Art Reports*, pages 101–126. The Eurographics Association, 2011. Cited on page 29.

[TBKP12] Michael Tao, Jiamin Bai, Pushmeet Kohli, and Sylvain Paris. Simpleflow: A non-iterative, sublinear optical flow algorithm. In *Computer Graphics Forum*, volume 31, pages 345–353. Wiley Online Library, 2012. Cited on page 11.

[TM98] Carlo Tomasi and Roberto Manduchi. Bilateral filtering for gray and color images. In *Computer Vision, 1998. Sixth International Conference on*, pages 839–846. IEEE, 1998. Cited on page 12.

[VG97] Eric Veach and Leonidas J Guibas. Metropolis light transport. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 65–76. ACM Press/Addison-Wesley Publishing Co., 1997. Cited on page 23.

[WBSS04] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: From error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13(4):600–612, 2004. Cited on page 54.

[WDP99] Bruce Walter, George Drettakis, and Steven Parker. Interactive rendering using the render cache. In *Rendering techniques' 99*, pages 19–30. Springer, 1999. Cited on page 28.

[Whi79] Turner Whitted. An improved illumination model for shaded display. *ACM SIGGRAPH Computer Graphics*, 13(2):14, 1979. Cited on page 18.

[WPB10] Manuel Werlberger, Thomas Pock, and Horst Bischof. Motion estimation with non-local total variation regularization. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2464–2471. IEEE, 2010. Cited on page 11.

[YNS+09] Lei Yang, Diego Nehab, Pedro V Sander, Pitchaya Sitthi-amorn, Jason Lawrence, and Hugues Hoppe. Amortized supersampling. *ACM Transactions on Graphics (TOG)*, 28(5):135, 2009. Cited on page 29.

[YWY10] Xuan Yu, Rui Wang, and Jingyi Yu. Real-time depth of field rendering via dynamic light field generation and filtering. In *Computer Graphics Forum*, volume 29, pages 2099–2107. Wiley Online Library, 2010. Cited on page 28.