Multi-View Depth Estimation

Stijn Brouwers

promotor :
Prof. dr. Philippe BEKAERT

Eindverhandeling voorgedragen tot het bekomen van de graad
Master in de informatica multimedia

Universiteit Maastricht

universiteit hasselt

A Word of Thanks

The realization of a thesis is not always an easy assignment. So I would like to thank all the people who have helped me to be able to bring this thesis to a successful end. I would like to thank the educational team from the "UIB" university in Mallorca, where I initially started my thesis. I would also like to guide a word of thanks toward my promoter prof. dr. Philippe Bekaert and my supervisors Maarten Dumont and Yannick Francken for their guidance and support. I would also like to thank all my friends and family for their support throughout the whole years of my studies.

Abstract

It is often desired to have a digital representation of a real world scene. One approach to retrieve digital information about a scene is to capture images from it by means of digital cameras. The problem with this method is that only a two dimensional representation of the three dimensional scene is available and limits us to completely reconstruct it. The problem that is being solved by multi-view depth estimation algorithms is the problem of retrieving information about the third dimension, depth. When the depth estimation for each pixel is correct we can digitally reconstruct the scene in three dimensions.

Depth estimation algorithms can work on different input parameters. Some algorithms use only images from one point-of-view while other algorithms use captured data from different viewpoints. In this thesis three different types of algorithms will be discussed. First a depth from defocus algorithm is presented that uses images from only one point-of view. This is followed by explaining the general design of stereo view depth estimation algorithms and their input and output data. Later in the thesis an implementation of both types is described. The last type of algorithm that is discussed uses images from more than two different viewpoints to derive depth for pixels in the reference images.

Nederlandse Samenvatting

Introductie

Het is soms gewenst om een digitale versie van een reële scène te verkrijgen. Het probleem dat hiermee hand in hand gaat is het feit dat het niet voor de hand liggend is om informatie over alle drie de dimensies (x,y en z) te verkrijgen. Een mogelijke oplossing om een reële scène te digitaliseren is door afbeeldingen te maken van de scène. Het probleem met deze methode is dat we slechts over informatie uit twee dimensies beschikken, de x- en y-dimensie. De taak van diepte schatting algoritmen is om de diepte, de derde dimensie, te extraheren uit deze afbeeldingen.
Sommige algoritmen gebruiken als input afbeeldingen die genomen zijn vanuit hetzelfde standpunt. Een voorbeeld van zo een algoritme is het *diepte van projectie defocus* algoritme waarbij een illuminatie patroon op de scène geprojecteerd wordt die ons van extra informatie voorziet, nodig om de diepte te kunnen bepalen. Andere algoritmen maken gebruik van twee afbeeldingen gemaakt vanuit twee verschillende standpunten. Deze algoritmen worden *stereo diepte schatting* algoritmen genoemd. Het laatste soort algoritmen gaat onder de naam *multi view diepte schatting* algoritmen en maakt gebruik van een sequentie, meer dan twee, afbeeldingen genomen vanuit verschillende standpunten.

Diepte van projectie defocus

Het eerste soort algoritme steunt op het principe van *diepte van defocus*. Dit principe zegt dat wanneer een scène punt niet in focus is voor een camera, zijn radiance wordt waargenomen door meerdere camera pixels. Het *omgekeerde projectie principe* formuleert hetzelfde principe op een andere manier en zegt dat de radiance waargenomen door een camera pixel, beïnvloed wordt door meerdere scène punten die niet in focus zijn. Wanneer we de camera vervangen door een projector in dit principe kunnen we

dit als volgt formuleren. Wanneer we de scène belichten met een illuminatie patroon, belicht één enkele pixel van het illuminatie patroon meerdere scène punten die niet in focus zijn. Het illuminatie patroon wordt dus met een bepaalde wazigheid waargenomen op de objecten in de scène. De hoeveelheid wazigheid waargenomen op een punt is direct gerelateerd aan de diepte van dit punt.
Zang en Nayar [1] stellen voor om een illuminatie patroon te projecteren op een vlak gelegen achter de scène. Een afbeelding wordt dan gemaakt van de scène terwijl deze belicht wordt met dit patroon. Het patroon wordt vervolgens één pixel verschoven en er wordt een nieuwe afbeeldingen gemaakt. Dit wordt enkele malen herhaald en levert ons een temporele radiantie sequentie op voor elke pixel. De volgende stap in het algoritme gebruikt deze data uit het spatiale domein en converteert deze naar het frequentie domein. Wazigheid in het frequentie domein komt voort uit het toepassen van een low-pass filter die zichtbaar is in figuur 2.5. De hoeveelheid wazigheid wordt bepaald door de mate waarin deze low-pass filter afneemt in frequenties. Door het toepassen van formule (2.4) verkrijgen we de k-de frequentie. Wanneer we de ratio nemen tussen de tweede en de derde frequentie, θ genaamd, hebben we een maat voor de wazigheid en dus een mogelijkheid om de diepte te bepalen voor het scène punt. Het overgaan van θ naar diepte gebeurt door middel van een opzoek tabel die als volgt wordt opgesteld. Een bord met enkele referentiepunt wordt voor de camera gehouden. Door middel van deze referentiepunten en het algoritme voorgesteld door Zhang [2] is het mogelijk de homographie van het bord te bepalen. Deze homographie maakt het mogelijk de diepte voor elk punt op het bord te bepalen. Vervolgens gaan we de scène met het bord belichten met het patroon en maken de nodige opnames. Wanneer we over de nodige afbeeldingen beschikken berekenen we de θ waarde op de voorgenoemde wijze. We beschikken nu over zowel de θ waarde als de diepte voor elk punt hetgeen genoeg informatie verschaft voor een opzoek tabel op te stellen. Om nu de diepte voor een scène te schatten berekenen we voor elk punt de θ waarde op de zojuist besproken manier. Wanneer we voor elk punt over de θ waarde beschikken mappen we deze naar de corresponderende diepte waarde met behulp van de zojuist geconstrueerde opzoek tabel.

Stereo view diepte schatting

Stereo view diepte schatting algoritmen maken gebruik van twee afbeeldingen gemaakt vanuit verschillende standpunten om een schatting te verkrijgen van de diepte map. Er wordt verwacht dat de afbeeldingen genomen zijn door middel van twee camera's, gebruik makend van een specifieke configuratie, genaamd epipolair gerectificeerd. Epipo-

lair gerectificeerd afbeeldingen hebben bepaalde eigenschappen die uitgebuit worden door de stereo diepte schatting algoritmen. We zeggen dat twee afbeeldingen epipolar gerectificeerd zijn wanneer corresponderende pixels tussen de twee afbeeldingen dezelfde y-coördinaten hebben. Wanneer we spreken over corresponderende pixels hebben we het over de pixels in beide images die hetzelfde punt in de scène voorstellen. Aangezien de afbeeldingen aan deze stricte configuratie voldoen is het zoeken naar corresponderende pixels gelimiteerd tot één dimensie. De beslissing of twee pixels corresponderend zijn wordt bepaald door het berekenen van een matching kost. De pixel combinatie met de laagste matching kost worden corresponderende pixels genoemd. Een voorbeeld van zo een matching kost is het berekenen van de SAD waarde formule (3.2) en registreert het verschil tussen de RGB waarden van beide pixels. Aangezien het bepalen of pixels corresponderend zijn afhankelijk is van zijn RGB waarden, presteert het algoritme slecht in bepaalde gevallen. In textuurloze gebieden zijn er meerdere pixels met dezelfde kleurwaarden en is het dus niet mogelijk te bepalen welk van deze pixels de juiste is. In gebieden die zichtbaar zijn in één afbeelding maar niet in de andere levert dit algoritme ook problemen aangezien er geen corresponderende pixel aanwezig is. Wanneer er toch twee corresponderende pixels gevonden zijn, wordt de diepte gedefinieerd als zijnde het verschil tussen de x-coördinaten. De x-coördinaten zijn direct gerelateerd aan de diepte door het feit dat objecten op de voorgrond een grotere translatie ondergaan in de afbeeldingen bij het verschuiven van de camera, dan objecten op de achtergrond. Stereo diepte schatting algoritmen kunnen worden opgedeeld in verschillende klasses waarvan de belangrijkste de lokale en globale algoritmen zijn. De lokale algoritmen maken gebruik van de zojuist besproken techniek terwijl de globale algoritmen een optimalisatie probleem oplossen. Bij een globaal algoritme wordt de optimale diepte map gezocht die een globale kostfunctie minimaliseert.

diepte mappen van een sequentie afbeeldingen

Deze methode maakt gebruik van een sequentie (meer dan 2) afbeeldingen, genomen vanuit verschillende standpunten en is vergelijkbaar met stereo diepte schatting algoritmen. Kang and Szeliski [3] stellen 2 methodes voor om meerdere diepte mappen te verkrijgen voor de gefotografeerde scène.

- Methode 1 berekent voor elke afbeelding uit de sequentie (of een subset ervan) een diepte map door gebruik te maken van meerdere input afbeeldingen. Meerdere diepte mappen worden verkregen door het algoritme meerdere malen uit te voeren,

steeds met een andere reference image. Er worden verschillende technieken toegepast om het algoritme beter te laten presteren bij occlusies. De technieken die gebruikt worden zijn temporele selectie, spatiaal aanpasbare windows en het berekenen van een globale kostenfunctie die gebruik maakt van een aangepaste versie van het graph cut algoritme[4].

- De tweede methode maakt gebruik van een globale kostenfunctie die meerdere diepte mappen gelijktijdig bepaald en omgaat met zichtbaarheid.

Methode 1

Deze methode maakt gebruik van een lokale diepte schatting techniek om de diepte map te bepalen voor een reference image door gebruik te maken van meerdere input afbeeldingen. Aangezien deze technieken slecht presteren bij occlusies, textuurloze gebieden en bij diepte ongelijkheden, wordt het algoritme uitgebreid om beter te presteren in deze gevallen. De gebruikte uitbreidingen worden in de rest van deze sectie besproken.

Spatiaal aanpasbare windows

In het standaard geval worden de matching kosten opgesomt over een window, gecentreerd rond de pixel waarvoor de matching kost wordt berekend. Een alternatieve aanpak is als matching kost voor een pixel de minimale waarde te kiezen van alle vensters die de pixel bevatten. Door het toepassen van deze techniek gaat het algoritme betere resultaten opleveren voor voorgrond objecten bij diepte ongelijkheden.

Temporele selectie

In het ideale geval zouden we in plaats van de matching kost over alle frames op te tellen, enkel de frames gebruiken waarin de pixel zichtbaar is. In plaats van gebruik te maken van alle frames gaan we slechts gebruik maken van een vast percentage van frames. Enkel de frames die de laagste matching kosten opleveren worden gebruikt. Deze techniek, net als de vorige, zorgt er voor dat het algoritme beter presteert bij diepte ongelijkheden.

Globale Optimalisatie

De laatste techniek gaat de verkregen diepte map iteratief verbeteren door een globale energie functie te minimaliseren. De globale energie functie is gebaseerd op het graph-cut algoritme voorgesteld door Boykov et al. [4] en uitgebreid met enkele technieken, voorgesteld door Kang en Szeliski [3].

Bij de eerste techniek krijgen de pixels die verborgen zijn in een andere afbeelding een label toegekend. Het label geeft aan dat de pixel een matching kost heeft die hoger is dan de hoogste, correcte matching kost voor corresponderende pixels. Deze pixels kunnen gedetecteerd worden door de hoge lokale matching kost in een post-processing stage. De pixels die dit label dragen krijgen een sanctie toegekend. Niet verborgen pixels, die grenzen aan een pixel die dit label draagt, krijgen ook een sanctie toegekend.
De volgende techniek berekent de zichtbaarheid van een punt in de tweede afbeelding en gebruikt deze kennis bij het bepalen van de matching kost.
De diepte map wordt dan als volgt verbeterd. De diepte map wordt meegegeven aan de globale energie functie en het graph-cut algoritme wordt uitgevoerd. Zichtbaarheid wordt berekend en betrokken bij de berekeningen. Dit levert een betere disparity map op die dan opnieuw als input gebruikt wordt voor de globale functie, dit levert op zijn beurt weer een betere zichtbaarheid schatting waarna we deze procedure opnieuw kunnen herhalen. Om er voor te zorgen dat het algoritme convergeert bevriezen we elke iteratie 15% van de alle dieptes die niet meer veranderd worden in de volgende iteratie. De pixels die bevroren worden zijn gekozen omdat ze de laagste matching kost opleveren.

Methode 2

Zoals eerder vermeld berekent de tweede methode, voorgesteld door Kang and Szeliski [3], meerdere diepte mappen gelijktijdig. Uit de set van input images wordt een subset van keyframes gekozen waarvoor de diepte map bepaald wordt. De data term van de globale energie functie wordt aangepast zodat we nu optimalizeren over alle diepte mappen gelijktijdig. De globale energie functie wordt uitgebreid zodat er rekening gehouden wordt met de compatibiliteit tussen de naburige diepte mappen. Net zoals in de vorige methode wordt ook hier gebruik gemaakt van een zichtbaarheidsterm die bepaald of een pixel zichtbaar in de referentie afbeelding ook zichtbaar is in een andere afbeelding.
Het algoritme is opgedeeld in twee fases
De eerste fase is de initialisatie fase en berekend voor elke image onafhankelijk een diepte map, gebruik makend van bijvoorbeeld het graph cut algoritme, zonder rekening te houden met zichtbaarheids- of de compatibility term.
In de tweede fase worden de initiele diepte maps gebruikt om de zichtbaarheid te berekenen en de globale energie functie wordt geminimaliseerd en levert nieuwe diepte mappen op die gebruikt worden in de volgende iteratie. Deze procedure wordt herhaald en levert elke iteratie een betere visibility schatting en dus een betere diepte map op.

Resultaten

Diepte van defocus

Het diepte van defocus algoritme toont diepte mappen die zeer goede resultaten opleveren bij diepte ongelijkheden. De resultaten tonen echter wel problemen bij donkere pixels en schaduwen die zichtbaar zijn vanuit het camera standpunt. Het algoritme kan ook niet overweg met spiegelende oppervlakten. Het algoritme toont zeer nauwkeurige resultaten zoals te zien is bij gebogen objecten zoals een bloempot. De pixels in het midden van deze bloempot zullen donkerder verschijnen als de pixels aan de zijkant.

Stereo view diepte schatting algoritme

Het basis SAD algoritme werd in het implementatie gedeelte uitgebreid met twee performantie verbeteringen. Eén van deze verbeteringen zorgt enkel voor een betere geheugenorganisatie hetgeen een noemenswaardige vermindering van de uitvoertijd van het algoritme teweeg brengt. De tweede uitbreiding noemt glijdende vensters en slaat gegevens op in het geheugen zodat deze niet meerdere keren berekend moeten worden, maar enkel uit het geheugen hoeven opgehaald te worden. Deze uitbreidingen is vooral duidelijk zichtbaar wanneer we grote vensters gebruiken. Wanneer van beide verbeteringen gebruik gemaakt wordt kan dit in sommige gevallen de uitvoertijd met 80% verminderen zonder de kwaliteit van de diepte mappen te beïnvloeden.
Verder zijn er twee uitbreidingen geïmplentereerd die de kwaliteit van de diepte mappen beïnvloed. Eén daarvan selecteert pixels waarvoor geen diepte berekend kan worden, in plaats van deze pixels verkeerde diepte waarden toe te kennen. De andere uitbreiding gebruikt een mediaan filter om ruis uit de diepte mappen te verwijderen.

Conclusie

In deze thesis werden verschillende diepte schattingsalgoritmen besproken die verschillende soorten van input gebruiken. diepte van defocus maakt gebruik van afbeeldingen, genomen vanuit één standpunt terwijl de scene belicht wordt door een projector. Een ander soort algoritmen, stereo view diepte schatting algoritmen, maken gebruik van twee epipolair gerectificeerde afbeeldingen om een diepte schatting te berekenen. Een ander algoritme maakt gebruik van afbeeldingen gemaakt vanuit meer dan twee verschillende standpunten.
Diepte van defocus algoritmen presteren zeer goed bij diepte ongelijkheden en zijn zeer nauwkeurig. Ze leveren echter slechte resultaten wanneer er donkere of spiegelende

objecten aanwezig zijn in de scene. Stereo view diepte schatting algoritmen maken gebruik van het verschil tussen kleurintensiteiten om de diepte te bepalen. Om deze rede presteren deze algoritmen minder goed bij gebieden die textuurloos zijn of die verborgen zijn in één van de twee afbeeldingen. Het algoritme is echter wel in staat in real-time de diepte map te berekenen.
Wanneer we afbeeldingen genomen van meer dan twee standpunten gebruiken als input, beschikken we over meer informatie. De extra informatie maakt het mogelijk om betere resultaten te verkrijgen bij de probleemsituaties uit het vorige algoritme. Dit algoritme is echter niet in staat om real-time diepte mappen te bepalen.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Depth estimation algorithms are used to retrieve depth information from a scene. The goal of this thesis is providing an overview of different ways to handle this problem. Different algorithms will be discussed and compared to each other. Both advantages and disadvantages for each algorithm will be provided. With this information available we are able to determine which type of depth estimation algorithm performs best for a specific scene.

The input data to depth estimation algorithms can vary. Some depth estimation algorithms use input data from only one point-of-view while others use images from two or more viewpoints. When images are used from only one point-of-view, the scene can be illuminated by a projector to provide us with extra information. When images from more than one point-of-view are being used, depth can be estimated by using correspondence between multiple images.

Using images from only one point-of-view of the scene makes it hard to derive depth for each pixel in the input images. By using a projector to illuminate the scene with different illumination patterns we are provided with extra information. The extra information gained by this illumination is an amount of blur for each pixel in the input images. It is possible to derive depth estimations for each pixel by using the amount of blur, observed at the pixel.

Stereo depth estimation algorithms usually use two input images which are epipolar rectified. The meaning of epipolar rectification is discussed later. For now it suffices to know that epipolar rectified images are images acquired from cameras that have a particular geometric configuration. Given two epipolar rectified images, the algorithms derive a disparity map for one of the two input images. A disparity map represents the difference in images obtained from the two different cameras. These disparity maps are

often used by other computer vision applications. Because these applications build their algorithms based on data from this disparity map, it is desirable that the disparity map is as correct as possible. Sometimes it is more important to retrieve a raw estimate of the disparity map generated in real-time. For example when a robot has to interact with the environment it is desirable that this can be done in real-time. When we look at stereo view depth estimation algorithms, four steps can be distinguished. The steps that are being performed depends on the stereo algorithm being used.
Other algorithms make use of images taken from more than two viewpoints and disparity maps are determined for a subset of these images. Because images are used from more than two points-of-view, multi view depth estimation algorithms are able to perform better than stereo view depth estimation algorithms.

Chapter 2

Depth From Projection Defocus

Depth from defocus algorithms obtain images from the scene from only one point-of-view. As presented by Zhang and Nayar [1] the scene is being captured by a digital camera while it is being illuminated by a digital projector. Different illumination patterns are projected onto the scene and the radiance at each pixel is studied while illuminated by these patterns. The digital projector produces a focused image on a specific depth. When a surface point is closer to the projector than the depth on which the projector is focused, the projected image appears blurred at this specific point. By analyzing the retrieved data we can model a projection defocus kernel using a linear system. By using this model we can recover depth at each pixel by estimating the parameters of its defocus kernel in frequency domain. The defocus kernel is scene-independent in contrast to camera defocus. This property comes from the fact that projector defocus happens on the projector plane while camera defocus happens on the scene surface.
In contrast to most depth recovery algorithms, this method is more accurate near depth discontinuities and depth is estimated at each pixel, independent from its neighboring pixels, without missing parts in the depth map.
The algorithm presents us with a simple linear model for projector defocus. A frequency domain method is then used to estimate the defocus kernel which is used to recover three dimensional information of the captured scene.

2.1 Depth From Defocus

Depth from defocus algorithms use images, observed by a camera, to determine the blur for each pixel. The camera is focused on a plane behind the scene and scene points

closer to the camera appear more blurred. The amount of blur can thus be related to the depth of the scene points. These methods are based upon the relation described by Langer et al. [5] which is illustrated in figure 2.1 and formula (2.1). The defocus principle shows us that whenever a scene points is closer to the camera that observes the scene, the radiance of the pixel influences multiple camera pixels. When a scene point is in focus it will only be observed by one camera pixel.

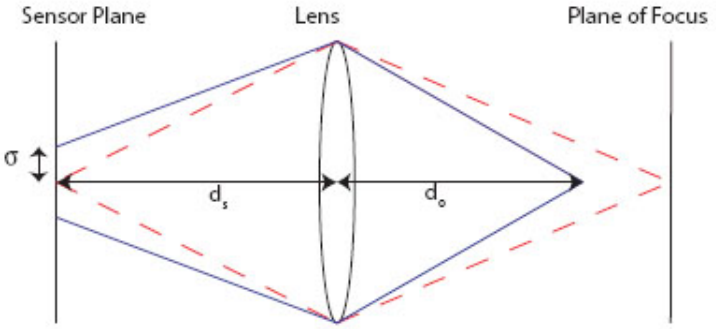

Figure 2.1: Fundamental relation underpinning depth from defocus. d_s is the distance between the lens and the sensor plane, d_o represents the distance between the scene point and the lens and σ is the radius of the blur circle. Image courtesy of Langer et al. [5]

$$d_0 = \frac{f d_s}{d_s - f - F\alpha} \tag{2.1}$$

In this formula f denotes the focal length of the lens, d_s represents the distance between the lens and the sensor, F refers to the lens aperture number and α is the radius of the blur circle created by a point at distance d_0. The lens aperture is a camera setting that influences the depth of field. A smaller lens aperture produces a longer depth of field which determines the range of scene points to be in focus at the same time.
When we have a measure for the blur radius at a scene point, we can use formula (2.1) to determine the distance of that point to the lens. When we have the distance from the point to the lens we obviously have a depth of the point in the scene.
Figure 2.1 shows us that whenever a point is located closer to the lens than the plane on which the camera is focused, the blur radius on the sensor plane grows larger. This is the principle of depth from defocus. The reverse projection from defocus principle is also presented by Langer et al. [5] and is illustrated in figure 2.2.

where $\hat{\delta}$ is the blur radius in the scene which is related to the lens-object distance,d_0,

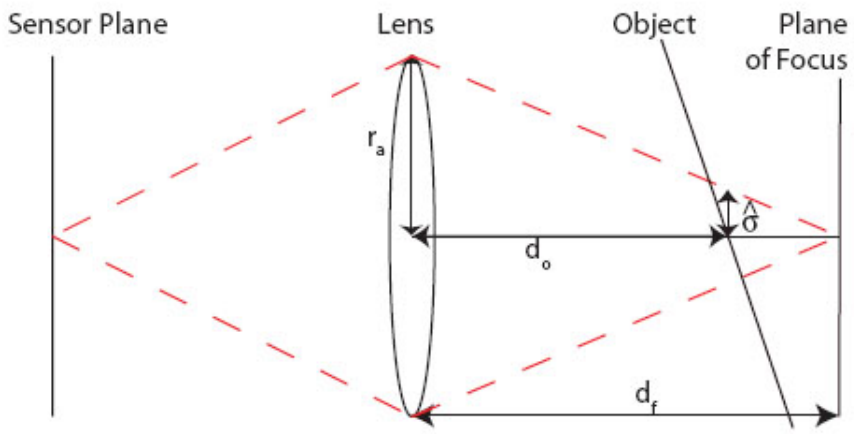

Figure 2.2: The reverse projection principle. Image courtesy of Langer et al. [5]

and d_f and r_a are the lens-focal plane distance and aperture radius respectively.
This figure shows us that a cone can be constructed between the lens and the point on the plane of focus, visible from a point on the sensor plane. The principle tells us that the radiance observed at the point on the sensor plane, depends on the radiance from all scene points within this cone. The size of the cone radius depends on the distance from the scene point to the lens.
When we use a projector instead of a camera, the sensor plane is replaced by the image that is projected onto the scene. This shows us that whenever a object is out of focus, one pixel in the illumination image is projected onto multiple scene points. This reverse projection principle is the basic principle on which the algorithm, discussed further in this chapter, relies.

2.2 Temporal Defocus Analysis

To determine the defocus kernel, the function used to represent the defocus, the scene is illuminated by means of a projector. The projector is focused behind the scene as discussed by Zhang and Nayar [1] and seen in figure 2.3.

For a scene point that is in focus for the projector, the scene point observed by the camera is illuminated by a single pixel on the image projected on the scene. When the scene point is out of focus the irradiance is illuminated by several pixels on the projected image. The irradiance for such a point equals the convolution of the defocus kernel and the illumination pattern. The radiance for a scene point on an opaque surface can be

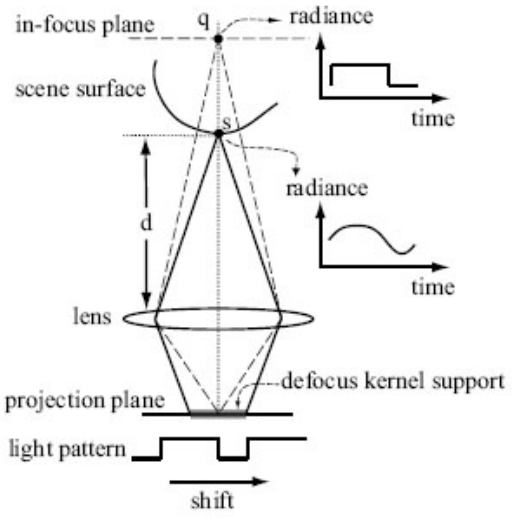

Figure 2.3: Principle of depth from defocus. Image courtesy of Zhang and Nayar [1]

written as

$$I = \alpha f(x, z) * P(x) + \beta \tag{2.2}$$

where I denotes the radiance at the scene point along any given outgoing direction, α is a factor that depends on surface reflectance, $f(x, z)$ is the defocus kernel, $P(x)$ is illumination pattern and β is the radiance due to ambient light [1]. This formula represents a linear system with the illumination pattern as its input parameter and the irradiance as its output. So what formula (2.2) shows us is that the irradiance depends on the defocus kernel. This means that if we have the irradiance from a pixel, it is possible to derive the defocus at that point.
The camera-projector configuration used by Zhang and Nayar [1] is shown in figure 2.4.

The beam splitter allows the light from the projector to be illuminated on the scene, while the light exitant from the scene is reflected onto the camera. This allows the camera and projector to be coaxial.

2.3 Frequency Domain

Jean Baptiste Joseph Fourier stated that every function whose area is finite under the curve can be expressed as the integral of sines and/or cosines multiplied by a weighting function [6]. This integral is called a *Fourier Series*. The Fouriers series can be recovered

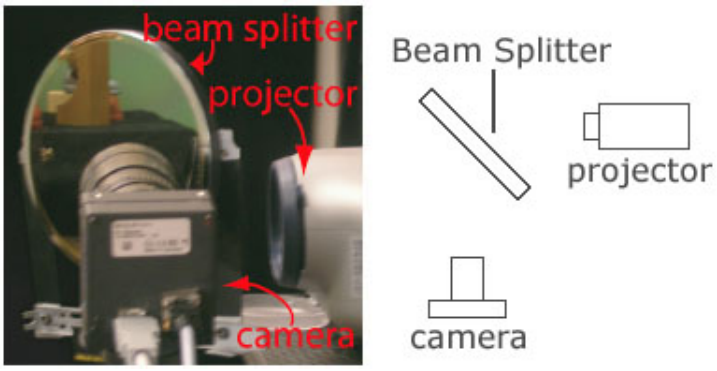

Figure 2.4: Camera-projector configuration using a beam splitter. Image courtesy Zhang and Nayar [1]

completely without loss of information via an inverse process. This allows us to work in the *Fourier Domain*, apply some changes and go back to the original domain without loss of information. In the Fourier transform, each term is composed of the sum of all values of the function. The values of the function are multiplied by sines and/or cosines of various frequencies. For this reason, the domain over which the resulting values of the Fourier transform range, is called the *Frequency Domain*.
The reason why the frequency domain is explained here is because an image can be seen as a function $f(x, y)$ where x and y represent the pixel coordinates and are called *spatial variables* . We can go from this spatial representation to the Fourier transform $F(u, v)$ where u and v are the *transform* or *frequency variables*. The reason why we would go from spatial to frequency domain is because some information is more obvious in frequency domain or some filters are easier to apply in this domain than in the spatial domain.
To apply a blur or defocus effect to an image, a low-pass filter in frequency domain can be applied to the image. A low-pass filter is a filter that attenuates high frequencies and passes low frequencies. Because the high frequencies are attenuated we would expect a low-pass filtered image to have less sharp details than the original image. An example of a low-pass filter in frequency domain is shown in figure 2.5.

2.4 Depth Estimation

Equation 2.2 can be seen as a linear equation where its input is represented by the illumination pattern and the output defined by the pixel radiance. Zhang and Nayar [1]

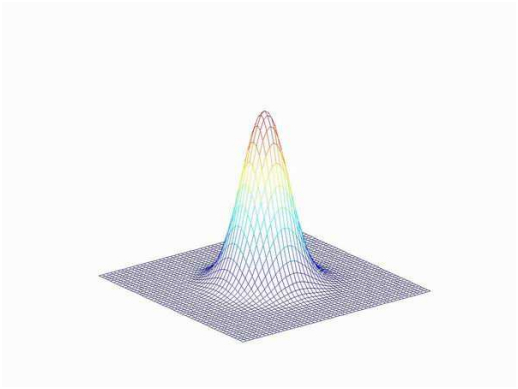

Figure 2.5: An example of a two-dimensional low-pass filter in frequency domain. Image courtesy of Favaro [7]

state that the radiance of a surface point over time is the response of its defocus kernel to the excitation by the illumination pattern, projected on the scene. The illumination pattern, as seen in figure 2.6, is constructed with a wide range of frequencies and is shifted over the scene. The illumination pattern is a binary periodic sequence, 011011011011... where each bit represents a 8-pixel wide stripe. The pattern is shifted one pixel at a time, 24 times, to retrieve 24 different images from the scene.

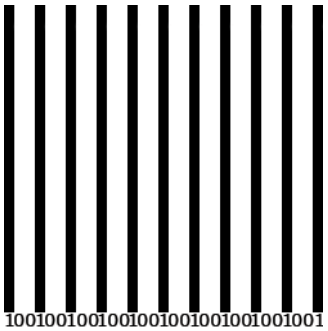

Figure 2.6: The illumination pattern that is projected onto the scene. Image courtesy of Zhang and Nayar [1]

By capturing the pixel radiance every time we shift the pattern, we now have a temporal radiance sequence $I_l, l = 0, ..., 23$, for a scene point.
In the next step a discrete-time Fourier series is defined by using the following formula

$$I_l = A_0 + \sum_{k=1}^{L-1} A_k cos(\omega_k l - \phi_k) \tag{2.3}$$

where $L = 24$ and is the number of times the pattern is shifted over the scene, $\omega_k = \frac{2k\pi}{L}$ and $\phi_k = arctan(B_k, C_k)$. A_k represents frequency k and consists out of two terms B_k and C_k. These terms are defined as follows

$$A_k = sqrt(B_k^2 + C_k^2) \tag{2.4}$$

$$B_k = \frac{1}{L} \sum_{l=0}^{L-1} I_l sin(\omega_k l) \tag{2.5}$$

$$C_k = \frac{1}{L} \sum_{l=0}^{L-1} I_l cos(\omega_k l) \tag{2.6}$$

Because of the fact that the defocus kernel is a low-pass filter, the amount of defocus is measured by how quickly A_k diminishes with k. Because the depth determines how much defocus is visible on a pixel we will be using this data to determine a disparity estimate for this pixel. Because A_0 depends on the ambient light, β, it cannot be used to estimate depth. We use the ratio of A_1 and A_2 to determine how severely the kernel attenuates the second-order harmonic with respect to the first-order one. So the following formula is used as a measure of depth

$$\theta = \frac{A_2}{A_1} \tag{2.7}$$

where $A_1 > A_2 > 0$ and $\theta \in [0, 1]$ because the kernel is a low-pass filter. The next step will be mapping θ to the pixel depth.

2.5 Mapping From Defocus To Depth

The mapping from θ to z happens in three steps

1. In the first step the correspondence between projector and camera pixels is computed. An algorithm to do so is presented by Scharstein and Szeliski [8]. The algorithm uses structured light to uniquely label each pixel. This can be done by using gray-code patterns or sine-waves [9].

2. When the correspondences between camera and projector pixels are determined, a foam board with four markers is tilted in front of the system and depth is computed for each pixel. To do this, an image is taken from the board and the homography

from the board to the projection plane is calculated. The homography allows to estimate the position and orientation of the board. When this data is known it is easy to determine the depth for each pixel. Zhang [2] proposes an algorithm to determine the desired homography.

3. The last step involves shifting the illumination pattern across the scene and computing the δ value for each point on the foam board. We now have both the θ value and the depth, or z-value, for each pixel. With this information it is possible to construct a look-up table that enables us to map a θ value to its corresponding disparity.

We now have the depth map and θ-values for the image of the board. Because the defocus kernel is assumed to be vertically but not horizontally invariant, a look-up table is being constructed for each column which maps a θ-value to its corresponding z-value.

2.6 Depth Recovery

Now that the mapping from θ to z is known, recovering depth for a scene is straightforward. 24 images are taken from the scene while shifting the illumination pattern, as discussed in section 2.2. The radiance values for all pixels are stored and used to calculate A_1 and A_2. This data is used to compute the θ-value for each pixel. Once we have all θ-values they are mapped to the corresponding z-values by using the look-up table, constructed as discussed in section 2.5.

Chapter 3

Stereo View Depth Estimation

Stereo view depth estimation is comparable to the principle of the human depth perception. To be able to see a scene in three dimensions the brain is provided with two images of the scene as visible from the left and the right eye. The point-of-view from both images is slightly different and correspondence between images is used to determine the depth for the scene. Stereo correspondence is a complicated and heavily investigated topic in the computer vision domain. Stereo View depth Estimation algorithms are algorithms that work with two frames/images that are acquired using a particular camera geometry called epipolar rectified, as discussed in section 3.1.2. These algorithms, given two epipolar rectified images, generate a disparity map that coincides with one of the input images. This disparity map relates a depth to every pixel.
Some computer vision algorithms use the disparity map as input data. For example some image warping algorithms use the disparity map for calculating a new view given an existing view. Hence, the result of this warping algorithm is related to the quality of the disparity map.
Stereo depth estimation algorithms often make assumptions, the most frequently used assumptions are:

Lambertian surfaces Surfaces are assumed to be Lambertian. This means that the appearance of a scene point does not vary with viewpoint.

Smooth surfaces Surfaces are often assumed to be smooth. This means that there is little difference between neighbouring pixels on the same surface. Great differences between intensities often refers to the presence of an edge.

Camera geometry Input images for stereo depth estimation algorithms are almost always assumed to be epipolar rectified.

3.1 Input Data

Stereo view depth estimation algorithms often require epipolar rectified input images. In this section you will be presented with a definition of epipolar rectification but first the pinhole camera model is explained because this model is later used to clarify the epipolar rectified configuration. The pinhole camera and epipolar geometry are defined by Hartley and Zisserman [10] but in this section the definitions are based on the work of McMillan [11].

3.1.1 Pinhole Camera Model

McMillan [11] defines a pinhole camera model as a model that collects intensities along rays that pass through a single point in space called its *center-of-projection*. These rays are contained within a bounded solid angle, defined by a solid planar section, the *image plane*. This solid angle is well defined as long as the center of projection does not lie on the extended image plane. As the boundaries of the image plane extend to infinity the solid angle boundary approaches to 2π steradians. This definition is illustrated in figure 3.1

In a pinhole camera model we consider the rays emanating from the center-of-projection with basis vectors $(\hat{i}, \hat{j}, \hat{k})$. In the image plane a local 2D coordinate system is being defined with vector basis $(\hat{s}, \hat{t})$ so we can identify each point by image coordinates.

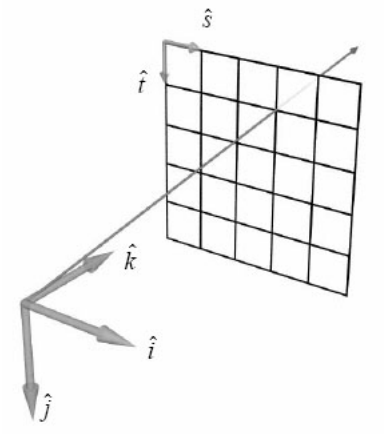

Figure 3.1: Pinhole camera configuration. $(\hat{i}, \hat{j}, \hat{k})$ are the basis vectors of the local coordinate system centered at the center-of-projection and $(\hat{s}, \hat{t})$ are the basis vectors of the local coordinate system of the image plane. Image courtesy of McMillan [11]

Each image-space point corresponds to one of the rays originating from the center of projection. The mapping from image space coordinates to rays can be done by the following planar mapping function.

$$\bar{d} = \begin{bmatrix} d_i \\ d_j \\ d_k \end{bmatrix} = \begin{bmatrix} a_i & b_i & c_i \\ a_j & b_j & c_j \\ a_k & b_k & c_k \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = P \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{3.1}$$

$\bar{d}$ represents the rays direction starting from the center-of-projection while (u, v) represents the image space coordinates.
We can interpret matrix P as having three columns that we represent as vectors $\bar{a}$, $\bar{b}$ and $\bar{c}$.
$\bar{a}$ and $\bar{b}$ are images of the image-plane base vectors $\hat{s}$ and $\hat{t}$, respectively, in the $(\hat{i}, \hat{j}, \hat{k})$ coordinate system. $\bar{c}$ is the vector connecting the origin of the rays with the origin of the image plane. see figure 3.2.

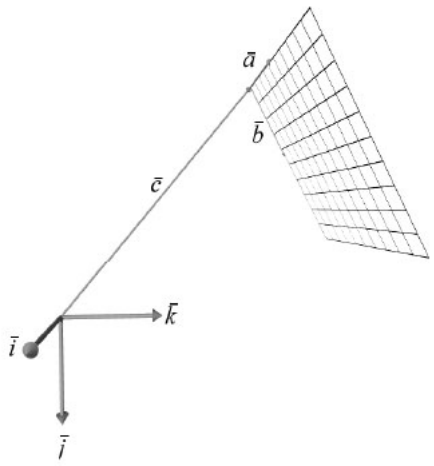

Figure 3.2: Meaning of $\bar{a}$,$\bar{b}$ and $\bar{c}$ vectors, used for ray/image-point transformation. $\hat{i}, \hat{j}$ and $\hat{k}$ represent the basis vectors of the local coordinate system, centered at the center-of-projection. Image courtesy of McMillan [11]

3.1.2 Epipolar Rectification

When we talk about epipolar rectified images we refer to a particular geometric configuration that exists between the images. Epipolar rectification can be achieved by a good camera alignment as discussed in section 3.1.2. Another possibility to achieve epipolar

rectified images is by rectifying the given images by means of a rectification algorithm. All rays originating from the center-of-projection from one camera model can be projected as lines on the image plane of the other camera model. The ray and the two centers-of-projection define a plane in 3D space, called an *epipolar plane*. The projection of such a plane results in a line on both images. All these projected lines intersect in the same point called the *epipole*, the epipole is the projection of the center-of-projection of one image on the image plane of the other image. An *epipolar line* is defined as the projection of the epipolar plane on the image plane as seen in image 3.3

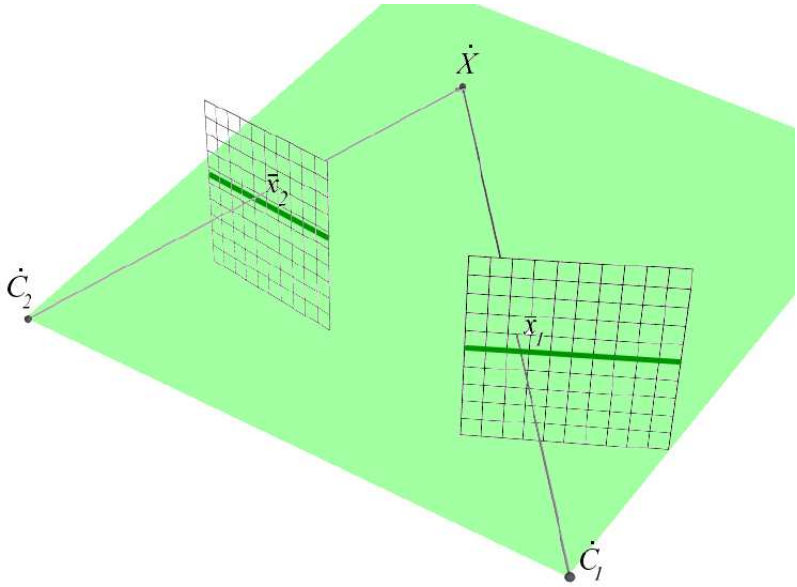

Figure 3.3: The projection of an epipolar plane (light plane) results in a epipolar line (dark line) on an image plane. $\dot{C}_1$ and $\dot{C}_2$ represent two centers-of-projection while $\dot{X}$ represents a scene point.

A epipolar rectified geometry can than be defined as a geometry for which all epipolar lines are parallel to one of the axes of the image planes. Most often the epipolar lines are parallel to the horizontal axes of the image plane.

By using pinhole camera configuration epipolar rectification can be defined as seen in figure 3.4.

In an epipolar rectified configuration, the vector connecting the two centers-of-

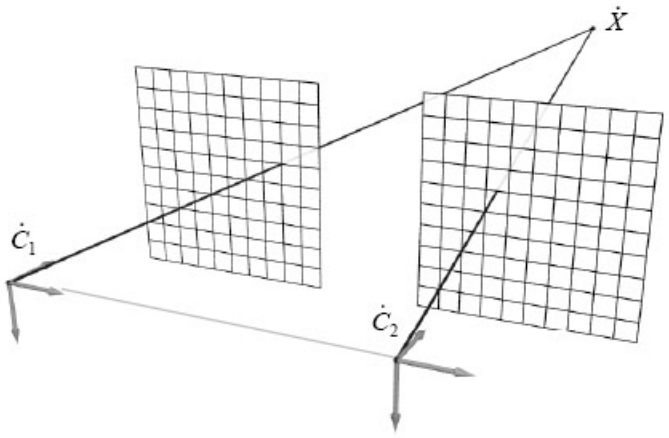

Figure 3.4: Epipolar rectified configuration where $\dot{C}_1$ and $\dot{C}_2$ represent two centers-of-projection while $\dot{X}$ represents a scene point. Image courtesy of McMillan [11]

projection is parallel to both image planes and the $\hat{i}$ basis vector of the camera space is parallel to the $\hat{s}$ basis vector of the image planes.

When images are epipolar rectified, the image-space coordinates of corresponding points differ only in one dimension. In other words this means that image-space coordinates of a pixel representing the same scene point for example have the same y-coordinate and is different in the x-coordinate.

When a depth estimation algorithm needs to search for corresponding pixels in both images, this configuration reduces the search area from the whole image to just one scan line in the image. This is a significant performance upgrade since the matching cost for a pixel (x, y) in the left image does not have to be calculated for all pixels in the right image. The matching cost only has to be calculated along one line instead of along the whole image. When pixel (x, y) corresponds pixel (i, y), the difference between x and i is related to the depth of the point in the scene. This value can be used to fill the disparity map.

An example of two epipolar rectified images often used to test depth estimation algorithms is the "Tsukuba" data set, see figure 3.5.

Figure 3.5: Two epipolar rectified images from the Tsukuba data set. Image courtesy of Scharstein et al. [12]

3.2 Design

According to Scharstein and Szeliski [12], stereo algorithms generally perform (subsets) of the following four steps in the process of generating a disparity map, given two epipolar rectified images.

Matching cost computation: For all pixels in a reference image, the matching cost is being computed for all candidate matching pixels. The matching cost is used as a means of measuring the correspondence between pixels. A typical way to determine the matching cost is to determine the RGB or intensity difference.

Cost (support) aggregation: This step combines the cost from multiple pixels to define a matching cost for a pixel in the reference image. For example by averaging the matching cost in the previous step over a neighbourhood. One way to aggregate the matching cost is to take the sum of the matching cost within a window of fixed size. Other methods use adaptive windows, shiftable windows,...

Disparity computation/optimization: Given a pixel and a possible matching pixel, a disparity is being estimated for the pixel in the reference image. When the images are epipolar rectified, the disparity used is the difference between the x coordinates of corresponding pixels in both the images. The pixel that is chosen to be a corresponding pixel is the one with the lowest matching cost.

Disparity refinement: In this step the disparity map from the previous stap is refined. An example of a refinement technique is applying a sub-pixel refinement stage or applying a median filter to remove noise.

The sequence of steps taken depends on the algorithm that is being used. Stereo depth algorithms can be divided into several classes. The two most popular classes are

Local Algorithms: Local (window-based) algorithms only use intensity values within a neighbourhood or window.
An example of a local algorithm is the traditional Sum Of Squared Differences algorithm, which is discussed in section 3.4, and can be divided in three steps.

1. The matching cost is calculated by applying the sum-of-squared differences formula on a pixel and its candidate matching pixel.
2. Aggregation is applied by summing up the matching costs within a fixed window around the pixel.
3. Disparity is then chosen as the difference in the x-coordinate between the reference pixel and the pixel with the minimum matching cost window.

Global algorithms: These algorithms make explicit smoothness assumptions and then solve an optimization problem. Mostly global algorithms do not perform an aggregation step but search for a disparity assignment to minimize a global cost function that combines data and smoothness terms. The main difference between global algorithms is the global cost function that is used.

Examples of other classes are iterative, cooperative and Dynamic programming algorithms.

3.3 Output

As discussed earlier the algorithms have as input two images of the scene, taken from two slightly different points of view. The goal of the algorithm is to provide the user with a depth estimation of the scene points, as seen from one of the reference images. The output of the algorithm should thus contain information that represents the third dimension, depth, or z-coordinate in 3D space.

3.3.1 Disparity Map

Disparity is the difference in images observed from the left and the right eye that the brain uses as a binocular cue to determine the depth of an object.
The output of depth estimation algorithms is often referred to as the *disparity map* or

disparity function.
The disparity function is defined as $d(x, y)$ which returns the disparity for pixel (x, y) in one of the reference images and the disparity map is a two dimensional array which contains the disparity for pixel (x, y) with respect to one of the reference images.
An example of such a disparity map can be seen in figure 3.6.

Figure 3.6: Disparity map for the left image of the Tsukuba data set, computed by a local stereo depth estimation algorithm. Red pixels represent pixels that are not assigned a depth value

3.4 Sum of Absolute Difference

3.4.1 Introduction

The Sum of Absolute Difference or SAD algorithm is a very simple algorithm. It's a local stereo depth estimation algorithm and uses corresponding pixels to determine depth for each pixel. As most stereo view depth estimation algorithms, SAD uses two epipolar rectified images as input and computes a disparity map for one of the two input images. A pixel in one image that represents the same scene point as a pixel in the other image are defined as being corresponding pixels. The goal of a local stereo depth algorithm is to find corresponding pixels between the two input images and use them to make a depth estimate. Because the images are epipolar rectified, corresponding pixels differ only in the x dimension. The difference between the x coordinate in the reference image and the x coordinate in the other image, is directly related to the depth of the scene point. So if we know which pixels in both images are corresponding pixels, we can use the difference between both x coordinates as their disparity estimate.

3.4.2 Matching Cost

The SAD algorithm uses color intensities to determine whether or not two pixels represent the same scene point. Each pixel in the reference image is compared to candidate pixels in the right image. The pixel pair with the lowest intensity difference, or matching cost, is selected as the corresponding pixel pair. The formula, used to compute the intensity difference is defined by Mühlmann et al. [13] as follows:

$$\begin{gathered} SAD(x,y,d) = \\ \sum_{i=-\frac{1}{2}(win_x-1)}^{\frac{1}{2}(win_x-1)} \sum_{j=-\frac{1}{2}(win_y-1)}^{\frac{1}{2}(win_y-1)} [\\ |R_L(x+i,y+j) - R_R(x+i+d,y+j)| + \\ |G_L(x+i,y+j) - G_R(x+i+d,y+j)| + \\ |B_L(x+i,y+j) - B_R(x+i+d,y+j)|] \end{gathered} \tag{3.2}$$

Chapter 4

Depth Maps From A Sequence Of Images

In stereo view depth estimation algorithms, a single depth map is estimated for one of the two input images. Because these algorithms depend on the correspondence between pixels in both images, they are known to have problems with occlusions and textureless regions. If regions are textureless, more than one possible corresponding pixel is found. If a region is visible in one image but is occluded in the other image, no corresponding pixel can be found. These two situations can lead to incorrect disparities.
Some techniques are presented to improve the algorithms in textureless regions or near occlusions. For example volumetric techniques, layered motion and stereo algorithms,... Despite the efforts, all current correspondence algorithms have their limitations.
A evaluation and comparison of multiple algorithms that use a sequence of input images is introduced by Seitz et al. [14]. In the paper by Kang and Szeliski [3], a method is proposed to determine depth by using a sequence of images from different points-of-view. Instead of computing a single depth map for one reference image, a depth map will be computed for each input image (or a subset of them). This technique should overcome most of these limitations. The paper proposes two methods to compute depth maps from a sequence of images.

1. In the first method, a depth map is computed for a single reference image, using multiple input images. this process is repeated for all images for which depth maps are required. Two complementary approaches are proposed to perform better near occlusions and therefore result in improved depth maps.
   The first approach uses spatially adaptive windows and selects a temporal subset of frames to match at each pixel. The second approach labels occluded regions by

using a global minimization technique based on graph cuts where good matching points are erased from the set of pixels that have to be matched later in the algorithm. The global optimization approach also explicitly takes visibility from pixels into account while computing the matching costs.
Both techniques can be combined in one single system.

2. In the second method, all depth maps are computed simultaneously with visibility handling.
The problem of extracting multiple depth maps is handled by using a global optimization over the unknown depth maps.

All images and results are originated from the paper by Kang and Szeliski [3]. Three image sequences are used throughout their paper and are shown in figures 4.1, 4.2 and 4.3.

Figure 4.1: 1st, 6th and 11th image from the eleven image flower garden sequence. Image courtesy of Kang and Szeliski [3]

Figure 4.2: 1st, 3rd and 5th image from the five image Tsukuba sequence. Image courtesy of Kang and Szeliski [3]

Figure 4.3: 1st, 3rd and 5th image from the five image symposium sequence. Image courtesy of Kang and Szeliski [3]

4.1 Method 1

In the first method a disparity map is calculated for each input image or a subset from these input images. A local algorithm is used to solve the problem. Local algorithms are known to have problems in textureless regions, near occlusions and depth discontinuities. Therefore the algorithm is upgraded using spatially shiftable windows and temporal selection. Although the algorithm discussed here is designed to use a sequence of input images, it will also produce reasonable results if only two input images are used, in the latter case the use of the temporal selection component is useless. After the depth map is calculated it is improved by using a global optimization approach, based on graph cuts.

4.1.1 Terminology

Before starting to explain the two methods used to compute depth maps given a sequence of images as input, we go through some terminology used throughout this chapter. The terminology and equations that are presented in this section are identical to the terminology and equations used by Kang and Szeliski [3]
Multi-view stereo algorithms use a collection or sequence of images as input. The collection of images is denoted as $I_k(x, y), k = 0...K$ where K is the total amount of images available. P_k refers to the camera matrix associated with image $I_k(x, y)$. The camera matrices are assumed to be computed elsewhere and are accurate enough to be used in the presented algorithms. $I_0(x, y)$ is the reference image for which we are about to calculate a depth map $d(x, y)$. $\hat{I}_k(x, y, d)$ is image I_k warped by a disparity map $d(x, y)$ and can be computed using the following formula

$$\hat{I}_k(x, y, d) = I_k(x + b_k d(x, y), y) \tag{4.1}$$

where b_k represents a scaling factor.
$E_{raw}(x, y, d, k)$ represents an initial raw matching cost between the reference image and image I_k warped with disparity $d(x, y)$ and can be computed using the following formula

$$E_{raw}(x, y, d, k) = \rho(I_0(x, y) - \hat{I}_k(x, y, d)) \tag{4.2}$$

$\rho(.)$ is a measure for the RGB difference between two images. An example of such a measure is the SAD value as can be seen in equation 3.2. So $E_{raw}(x, y, d, k)$ is the RGB difference between the reference image and another input image warped with disparity $d(x, y)$ over an unaggregated region. The task of multi-view stereo algorithms is to compute a disparity map $d(x, y)$ for which the raw matching costs are low for all images. Since the raw matching cost can be very noisy it is advised to aggregate over a region to improve the results. A simple aggregation technique is "sum of sum of squared differences" or simply SSSD. The following formula shows how to calculate the SSSD value.

$$E_{SSSD}(x, y, d) = \sum_{k \neq 0} \sum_{(u,v) \in W(x,y)} E_{raw}(u, v, d, k) \tag{4.3}$$

$W(x, y)$ is a square window centered around pixel (x, y). In words, we will sum up the raw matching costs over a window and sum these up for every image in the input collection except for the reference image. Local algorithms use the disparity with the lowest SSSD value. This performs well in textured regions and not near occlusions and depth discontinuities. In other cases local algorithms do not perform well. Kang and Szeliski [3] present two techniques to improve the algorithm in these cases. One technique uses spatially shiftable windows and the other technique uses temporal selection.

4.1.2 Spatially Shiftable Windows

Until now, if we want to calculate the matching cost between two pixels, the cost is aggregated over the windows centered around the pixels to be matched. Instead of only using the windows centered around these pixels, spatially shiftable windows use all the windows that include the pixels to be matched. This adds very little computation because the value of a shifted window is the same as that of a centered window at some neighboring pixel. This can be seen in figure 4.4. As can be seen in figure 4.5 this approach can improve the algorithm near depth discontinuities. The black pixel now also matches correct in both the middle and left image but not in the right frame because the scene point is occluded from this point-of-view. Trying to match the pixel in the right image will lead to erroneous depth estimations. To solve this problem, temporal selection is used, which is discussed in the next chapter.

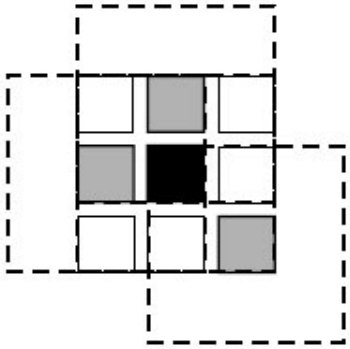

Figure 4.4: Working with all windows containing a pixel (black pixel in this image) is the same as taking the minimum matching cost of all centered windows around pixels in the window centered around the black pixel. This image shows three shifted windows of a 3x3 neighbourhood. Image courtesy of Scharstein et al. [12]

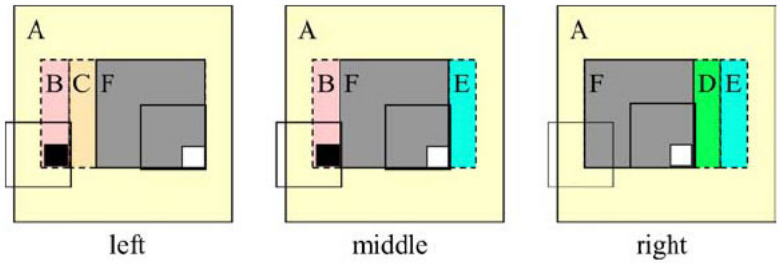

Figure 4.5: The white pixel now matches correctly in all frames. This would not be the case if we would use centered windows. The black pixel also matches correctly in the left and middle image. Image courtesy of Kang and Szeliski [3]

The results of applying spatially shiftable windows on the flower garden image sequence can be seen in figure 4.6. Differences are visible but they are not spectacular.

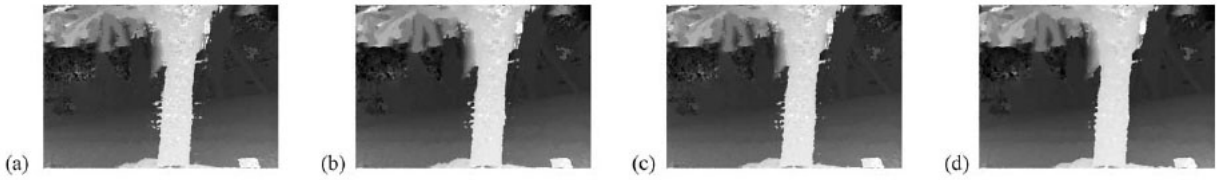

Figure 4.6: a) Result of using 3x3 centered windows, b) Result of applying 5x5 centered windows, c) Result of applying 3x3 shiftable windows, d) Result of applying 5x5 shiftable windows. Image courtesy of Kang and Szeliski [3]

4.1.3 Temporal Selection

Instead of using all frames to search for a match for a pixel it would be better to only use those frames in which the pixel is visible. For a camera moving along a continuous path, a pixel that is about to be occluded will often be occluded only in the preceding or the succeeding frames. Instead of just choosing between preceding or succeeding frames, a fixed percentage of frames can be chosen to be used. For example 50% of all available frames can be used. The frames we choose is based on the matching score the frame has. So for each pixel we compute the SSD value and then sum up the matching costs until we have used 50% of all frames. This approach is better for scenes in which for example a picket fence is present. The picket fence would make some pixels alternately visible and this technique would thus perform better by choosing frames from both succeeding as preceding frames instead of just one side. Figure 4.7 shows the results of applying temporal selection. The figure shows that temporal selection performs really well near depth discontinuities.

4.1.4 Global Techniques

As an alternative approach for dealing with the problems that arise in local depth estimation algorithms near occlusions,depth discontinuities and textureless regions, Kang and Szeliski [3] propose the use of a global technique. A global energy function normally consists of two terms as can be seen in the following formula.

$$E_{global}(d(x,y)) = E_{data} + E_{smooth} \tag{4.4}$$

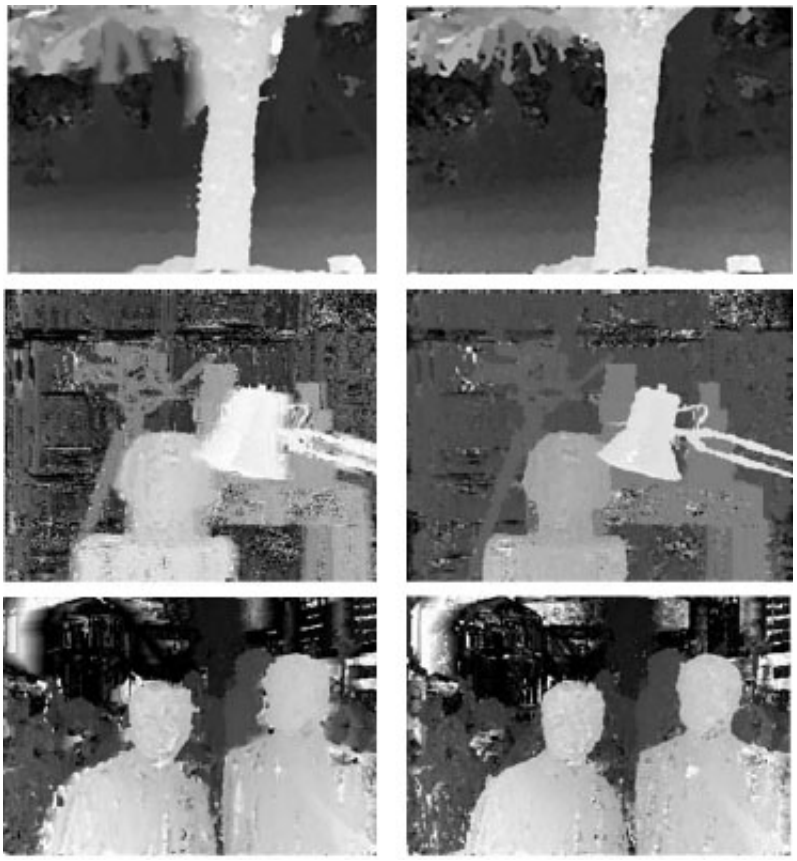

Figure 4.7: Results of applying the temporal selection principle on (from top to bottom) the Flower Garden, Tsukuba and Symposium sequence . Image courtesy of Kang and Szeliski [3]

The disparity $d(x, y)$ that minimizes this global energy function is chosen as the solution to the depth estimation problem. As said earlier formula (4.4) consists of two terms, a data term E_{data} and a smoothness term E_{smooth}. E_{data} is simply a summation of the local matching costs and is computed using the following formula.

$$E_{data} = \sum_{(x,y)} E_{SSSD}(x, y, d(x, y)) \tag{4.5}$$

Because smoothness is handled by the global smoothness term, no aggregation is used in the SSSD term. The smoothness term adds a certain amount to the global energy function so that disparity maps that are smooth have priority over disparity maps that have salt-and-pepper noise. The smoothness term is calculated as in formula (4.6).

$$E_{smooth} = \sum_{(x,y)} [s^h_{x,y}\phi(d(x, y) - d(x + 1, y)) + s^v_{x,y}\phi(d(x, y) - d(x, y + 1))] \tag{4.6}$$

Scharstein and Szeliski [12] define ϕ is a monotonically increasing function of disparity difference. ϕ can take different forms. It can be a quadratic function but may lead to poor results near object boundaries. Functions that do not exhibit this problem near object boundaries are called discontinuity preserving functions. $s^h_{x,y}$ and $s^v_{x,y}$ are horizontal and vertical smoothness strengths respectively which can be spatially varying. The graph cut algorithm introduced by Boykov et al. [4] is one that minimizes the full 2D global energy function. Next Kang and Szeliski describe two extensions on this global graph cut algorithm to better handle partial occlusions present in multi-view stereo. One extension uses pixel labeling and visibility computation while the other extension uses hierarchical disparity computation to improve efficiency.

Pixel Labeling

Pixels that do not have good matches in other images are still assigned a disparity, even if this disparity is erroneous. These pixels often have a high matching cost and are thus easy to detect. This problem can be solved by introducing a label d_{occl} which is assigned to pixels that are outliers or potentially occluded. A penalty E_{occl} is associated with every pixel that carries the d_{occl} label. The penalty E_{occl} should be assigned a value that is higher than the highest value associated with correctly matching pixels. All non-occluded neighboring pixels are assigned a smoothness penalty, Φ_{occl}. The matching cost $E_{SSSD}(x, y, d(x, y))$ in previous formulas can be rewritten as $E'_{SSSD}(x, y, d(x, y))$ where $E'_{SSSD}(x, y, d(x, y))$ is defined by the following formula:

$$E'_{SSSD}(x, y, d(x, y)) = \begin{cases} E_{SSSD}(x, y, d(x, y)) & \text{if } d \neq d_{occl} \\ E_{occl} & \text{if } d = d_{occl} \end{cases} \tag{4.7}$$

and smoothness function ϕ is modified to the following definition

$$\phi'(p-q) = \begin{cases} \phi(p-q) & \text{if } p \text{ and } q \neq d_{occl} \\ \Phi_{occl} & \text{if } p \text{ or } q = d_{occl}, p \neq q \\ 0 & \text{if } p = q = d_{occl} \end{cases} \tag{4.8}$$

Labeling of occluded pixels often fails to label occluded pixels in textureless regions because these pixels may match with pixels in the same region even if it is not the correct pixel. This a direct consequence of the fact that the region is textureless.

Visibility Computation

To use visibility inside the global optimization function Kang and Szeliski [3] make use of a visibility function. The visibility function is denoted as $v(x, y, d, k)$ and can be computed as a function of the disparity assignments at layers closer than d. $o(x, y, d')$ is called the opacity function and represents a binary image of the pixels assigned to depth d'. $s(x, y, d', d, k)$ is called the shadow function and is the shadow that the opacity casts onto another level d relative to camera k. This can be derived from the homographies that map between disparities d' and d, where homography $H_k(d)$ can be computed directly from camera matrices P_0 and P_k and depth d. Pixels in image k that are in a shadow region are pixels that are not visible from the reference image. The shadow function is defined as followed:

$$s(x, y, d', d, k) = (H_k(d)H_k^{-1}(d')) \circ o(x, y, d') \tag{4.9}$$

The visibility function can then be determined using the following formula:

$$v(x, y, d, k) = \prod_{d'<d} (1 - s(x, y, d', d, k)) \tag{4.10}$$

The formula runs over all depth levels smaller than d and determines if the pixel (x, y) in image k is in the shadow of one of these depths. If this is the case, the visibility algorithm returns 0 else it will return 1. The latter case means that pixel (x, y) in image k is visible from the reference image. To involve visibility in the global energy function, the raw matching cost can be replaced by

$$E_{vis}(x, y, d, k) = v(x, y, d, k)\rho(I_0(x, y) - \hat{I}_k(x, y, d)) \tag{4.11}$$

By using this formula we explicitly take occlusions and partial visibility in account. The problem with this approach is that it's complicated to minimize the global energy function. Kang and Szeliski [3] present us with a solution to this problem. To minimize

the global function a good way would be to start with all pixels visible. The graph-cut algorithm [4] is run as usual to get an initial depth estimation, $d(x, y)$. Visibility is recomputed for all pixels by using this disparity map. The disparity map is then improved by re-optimizing the modified energy function. The problem with this option is that this process may not converge. This comes from the fact that visibilities assumed for one iteration may be undone by a re-assignment of labels in that iteration. To make sure this approach converges a fixed percentage of all the pixels is frozen. For example we choose 15% from all the pixels and use their disparity in the final disparity map. We choose to freeze those pixels that have the lowest matching costs. The next step runs the graph cut algorithm again on the remaining pixels until depth is computed for all pixels. The result of applying this algorithm can be seen in figure 4.8. The most significant improvement of applying the global technique is visible in the dataset from the Tsukuba university.

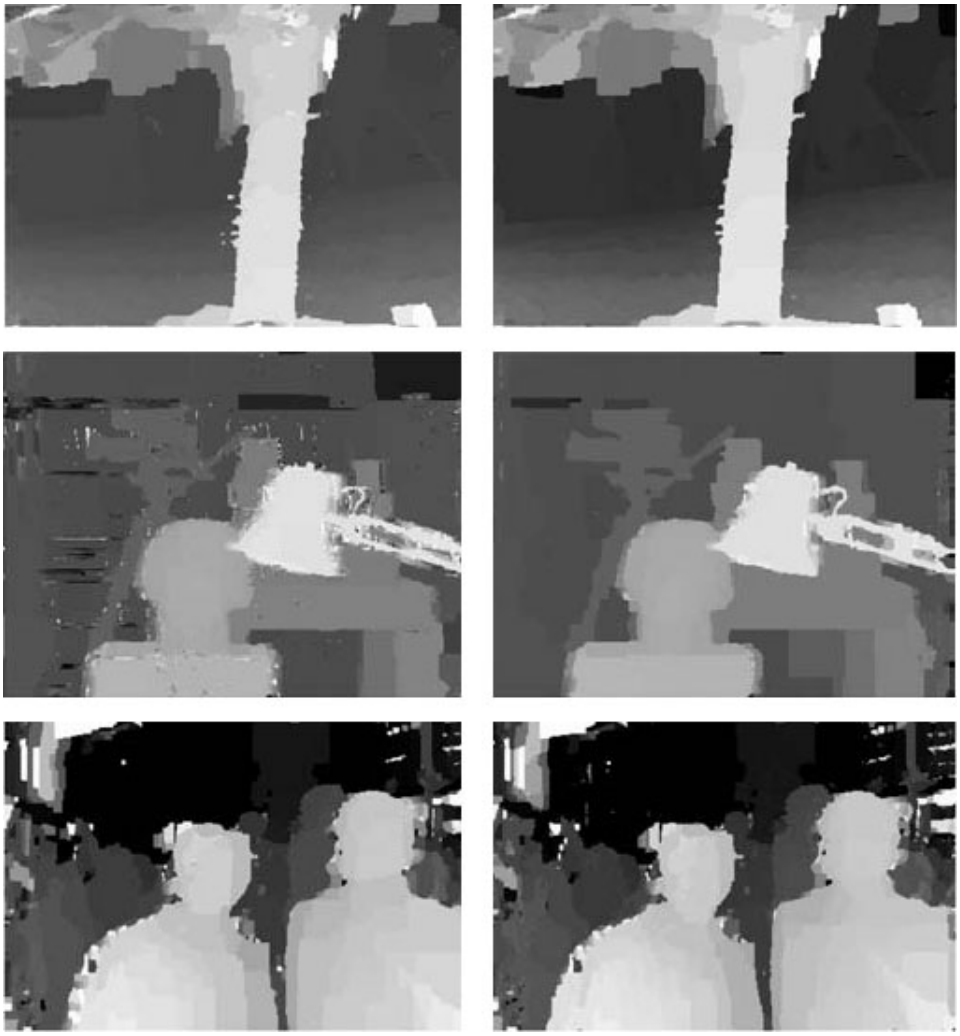

Figure 4.8: Result of applying the global visibility based graph-cuts algorithm on (from top to bottom) the Flower Garden, Tsukuba and Symposium image sequences. Image courtesy of Kang and Szeliski [3]

4.2 Method 2

This section describes a technique proposed by Kang and Szeliski [3] which computes multiple depth maps from a sequence of input images simultaneously. It is referred to as the multi-view stereo reconstruction framework.

4.2.1 Terminology

From the set of input images $I_k(x, y), k = 0...K$, a set S of keyframes is chosen. For these keyframes a depth map $d_l, l \in S$ will be estimated. The choice of which frames to be taken as being keyframes is problem dependent and is comparable to the selection of I and P frames in video compression techniques [15]. Because we now use a collection of reference images the formula for calculating a warped image has to be adjusted.

$$\hat{I}_k^l(x, y, d) = I_k(x + (b_k - b_l)d(x, y), y) \tag{4.12}$$

where b_k and b_l represent scaling factors. In the ideal case without occlusions, this warped image would coincide with reference image I_l. This leads to the fact that now, the raw matching cost depends on l as can be seen in the following formula.

$$E_{raw}(x, y, d, k, l) = \rho(I_l(x, y) - \hat{I}_k^l(x, y, d)) \tag{4.13}$$

This formula represents the raw matching cost between reference image I_l and the image I_k warped by disparity map d_l. When we include visibility in the raw matching cost formula we get the following

$$E_{vis}(x, y, d, k, l) = v(x, y, d, k, l)\rho(I_l(x, y) - \hat{I}_k^l(x, y, d)) \tag{4.14}$$

The visibility function $v(x, y, d, k, l)$ will be explained later. The data term of the global energy function to be minimized can then be defined as followed

$$E_{data} = \sum_{l \in S} \sum_{k \in \aleph(l)} w_{kl} \sum_{(x,y)} E_{vis}(x, y, d, k, l) \tag{4.15}$$

Image I_k with $k \in \aleph(l)$ is a neighboring frame of image I_l for which corresponding pixel colors agree. w_{kl} is a weight that determines how much the neighboring image contributes to the estimate of disparity map d_l. This formula shows that minimizing the global energy function involves minimizing over multiple depth maps simultaneously. For the multi-view stereo reconstruction framework, the global cost function is extended to consist out of three terms as can be seen in the following formula

$$E_{global} = E_{data} + E_{smooth} + E_{compat} \tag{4.16}$$

The formula consists out of the data term, which is explained above, a smoothness term and a compatibility term. This formula differs from formula (4.4) because it encapsulates an extra term. The other difference between the two methods presented are not directly visible in this formula but are integrated in each term. The smoothness term is the same as in 4.6 but now involves summing over all depth maps l.

$$E_{smooth} = \sum_{l \in S} \sum_{(x,y)} [s^h_{x,y} \phi(d_l(x,y) - d_l(x+1,y)) + s^v_{x,y} \phi(d_l(x,y) - d_l(x,y+1))] \quad (4.17)$$

The E_{compat} function is defined as follows

$$E_{compat} = \sum_{l \in S} \sum_{k \in S} w_{kl} \sum_{(x,y)} v(x,y,d,k,l) \rho_C(d_l(x,y) - \hat{d}^l_k(x,y,d)) \quad (4.18)$$

where ρ_C denotes the disparity difference and $\hat{d}^l_k(x,y,d)$ can be computed analogous as a warped image in formula (4.12). Instead of enforcing smoothness on a disparity map like the smoothness term, the compatibility term enforces compatibility between depth maps at different neighboring keyframes. The computation of The definition of the visibility function $v(x,y,d,k,l)$ is different from where we compute a single depth map. We first compute the warped disparity map $\hat{d}^l_k(x,y,d)$ and then define the visibility function as follows

$$v(x,y,d,k,l) = ((d_l(x,y) - \hat{d}^l_k(x,y,d)) \leq \delta) \quad (4.19)$$

where δ represents a threshold which accounts for errors in estimation. Whenever a pixel corresponding to (x,y) is out of the boundaries of the image dimension, $v(x,y,d,k,l)$ is set to zero.

4.2.2 Depth Estimation Algorithm

The following algorithm was proposed by Kang and Szeliski [3] and combines ideas from hierarchical estimation, correlation-style search and sub-pixel motion/disparity estimation.

The algorithm works in two phases. The first phase is an initialization phase and the second phase involves computing visibility and enforcing compatibility between disparity maps.

Phase One

This phase computes depth maps for each keyframe separately without enforcing the compatibility term nor computing visibility. The disparity maps in this phase can be computed by using a hierarchical algorithm like for example the graph cut algorithm described in section 4.1.4.

Phase Two

Once the initial set of disparity maps d_l is calculated, visibility can be computed and the compatibility term E_{compat} is used in the global cost function. The algorithm can be executed several times and will obtain better visibility and thus better depth results at each iteration.

Chapter 5

Implementation

This chapter will discuss the implementation of two depth estimation algorithms. In section 5.1, a depth from projection defocus algorithm is presented which is based on the work of Zhang and Nayar [1]. In section 5.2, a local stereo depth estimation algorithm is discussed, the implementation is based on the work of Mühlmann et al. [13]. First a basic SAD algorithm is implemented which is optimized with several performance and quality upgrades. The results of the algorithms can be found in section 6.

5.1 Depth From Projection Defocus

This implementation is based on the work from Zang and Nayar [1]. It uses a camera and a projector to estimate the depth for each pixel, captured from a real-world scene by the camera. The basic principle is to focus the projector on a plane, located behind the scene objects. An illumination pattern, as shown in figure 2.6, is shifted over the scene. We choose this illumination pattern because it contains a wide range of frequencies. The next step in the algorithm is to record the radiance at every pixel, illuminated by the projector. Because the objects are positioned closer to the projector than the plane on which the projector is focused, the illumination pattern is defocused on each object. The amount of defocus is directly related to the depth of the object. For this reason we analyse the radiance sequence for each pixel and determine the amount of defocus, introduced by the object on that point. The amount of blur for a sequence of radiance values is determined by converting the spatial domain values to frequency domain. In frequency domain, an image is blurred by applying a low-pass filter. An example of a low-pass filter can be seen in figure 2.5. How quickly the frequencies decrease represents the amount of blur. An amount for how fast the frequencies diminish is to take the

second and third frequency and compute their ratio. The obtained value is referred to as θ and this value is directly related to the depth of that pixel.

5.1.1 Camera And Projector Configuration

In the paper upon which this implementation is based, a beam splitter is used to obtain a coaxial camera-projector configuration. The camera and projector are made coaxial to avoid shadows in the obtained images. Because shadows are not influenced by the projection of the illumination pattern, we would not be able to determine depth for these pixels. In the original configuration, a projector was placed behind the beam splitter which allowed the light to pass. Light exitant from the scene was reflected in the beam splitter into the camera lens. The original configuration can be seen in figure 2.4.

Because no beam splitter was available, an alternative approach was adopted which is much cheaper but also approximate. If the projector and camera are not coaxial, shadows are visible through the eye of the camera. This principle can be seen in figure 5.1 and shows the case where the camera is placed on the right-hand side from the projector.

Each object illuminated by the projector casts a shadow, visible to the camera, to the right. When the camera is placed to the left of the projector, the shadow is cast to the left of the object. The farther the camera and projector are placed, relative to each other, the more shadow becomes visible. By placing the camera on the same x axis, above or below the projector, shadows are eliminated to both the right and left side. Because the camera is positioned below or above the projector, the camera will capture shadow below or above the object respectively. A configuration is chosen so that the camera is positioned under the projector. This will cast shadow under the objects in the scene. If objects are placed on a ground object, no shadows will be visible. This limits our scene to objects that are not hanging in the air, but standing on another object. The configuration can be seen in figure 5.2.

Figure 5.3 shows the example of a scene illuminated with the pattern as seen through the lens of the camera.

5.1.2 The Algorithm

The first step in the algorithm is to acquire images while shifting the pattern. The pattern is shifted 24 times with 1 pixel. Every time the image shifts, the camera is triggered to capture an image. Because the projector vibrates at a high frequency, its brightness is not stable over time which means the radiance changes slightly over time.

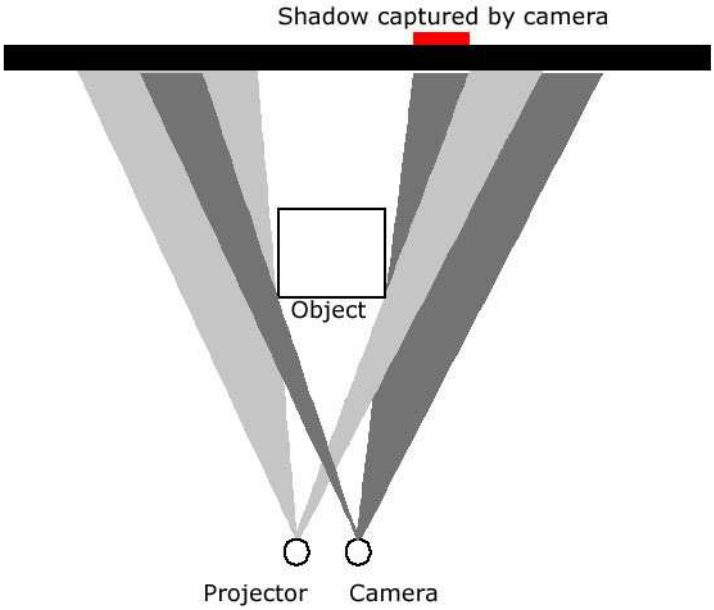

Figure 5.1: Principle of shadow, visible with the camera

Figure 5.2: Camera and projector configuration used for the implementation

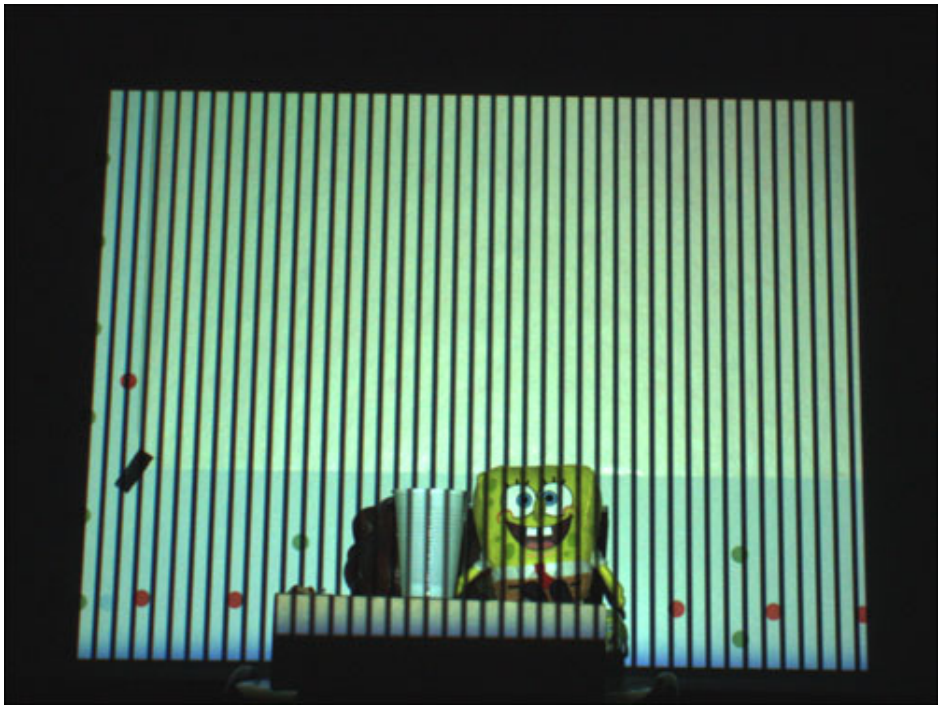

Figure 5.3: Example of a frame, captured by the camera while illuminated by the projector

For this reason multiple frames are captured while the scene is illuminated by the same pattern. After we acquire all the images, we take the average for each frame and use this to get the desired pixel radiance. The more images we take for each frame, the less the radiance will fluctuate.
To go from spatial to frequency domain, we convert the 24-bit color images to grayscale images. This is done with the following formula [16]:

$$Y = 0.3 * R + 0.59 * G + 0.11 * B \tag{5.1}$$

and is applied to the averaged frames.
We use the formulas defined in section 2.4 to retrieve the second and third frequency for each point. These frequencies are used to determine the speed at which the defocus kernel diminishes. The first frequency is not used because it depends on the ambient light β. The ratio between the second and the third frequency is called θ and is a measure for the amount of defocus at the specific point.
In the original paper a method is adapted to map theta to depth. This is done by building a lookup table for each column because the defocus kernel is assumed to be vertically invariant but has horizontal variation. The mapping between θ and depth is not a necessary step in the algorithm but will improve the results. Because θ is directly related to the depth it will only affect the result with a scale factor and will thus only serve as a solution for the horizontal variation from the defocus kernel.

5.1.3 Limitations

As discussed before, the algorithm will not be able to determine depth for each pixel with the discussed configuration if there are pending objects in the scene. These objects will create shadows that are visible through the camera. Because the projector does not illuminate the pixels they will have the same values for all frames and there will be no defocus visible. Because we use the defocus to estimate depth, it is not possible to determine depth for these pixels.
When a black scene point is illuminated with a pixel that is black in the illumination pattern, the point is observed as being black by the camera. When the same point is illuminated with a pixel from the illumination pattern that is white, the scene point is still observed as being black. The illumination of such a scene point does not give us extra information. Because the radiance of this pixel, observed by the camera, remains approximately the same, we can not derive the amount of defocus from the illumination pattern for this pixel. If we cannot determine the amount of defocus for a pixel it is not

possible to compute a disparity for it. This means that we are limited to scene objects that are not to dark.
If scene objects are not Lambertian, the light originating from the projector is reflected and can not be captured by the camera. This means that depth estimates for objects that reflect the light are erroneous depth values.
Obviously depth can only be estimated for regions that are illuminated by the camera. If the scene is to big to be illuminated by the illumination pattern, we can not estimate depth for all scene objects.

5.2 Stereo-View Depth Estimation Algorithm

The implementation of this local stereo depth estimation is based on the paper presented by Mühlmann et al. [13].
First the base algorithm will be discussed. This base algorithm will be upgraded step-by-step to gain in both quality and performance.
The resulting depth maps and computation times can be found in chapter 6.

5.2.1 Base Algorithm

The base algorithm is a simple SAD algorithm. A SAD algorithm is a local stereo depth estimation algorithm. For each pixel in the reference image, a list of candidate pixels is chosen. Between the reference pixel and all of its candidate pixels, a matching cost is computed. In an SAD algorithm this matching cost is determined by computing the SAD or Sum Of Absolute Difference value which is defined in Formula (3.2). Once we have this value for all candidate pixels, the pixel with the lowest matching cost is chosen to be the corresponding pixel and a depth estimate can be determined by using the coordinates of both the reference and corresponding pixel.
Depth will be estimated for the right image of the two input images. In the right image, objects in the scene will be positioned more to the left than the same object in the left image. This property can be seen clearly in figure 5.4. This limits our search for corresponding pixels to only one direction and leads to the fact that all candidate pixels for pixel (i, j) in the right image have x coordinates so that $x > i$.
As discussed before, the candidate pixels are limited to those pixels that have the same y coordinate. This comes directly from the fact that we are using epipolar rectified images as input.
To further limit candidate pixels it is possible to define two variables. One variable defines the bottom limit for an interval between which we are searching for corresponding

pixels, while the other defines the upper limit. If we know that the difference between corresponding pixels is at least a pixels and b pixels at max, it is useless to search for it outside this interval. If for example we define a as being 5 and b as being 30, the search interval for pixel with coordinate $(100, 100)$ is now limited to all pixels with coordinates (x, y) where $x \in [105, 130]$ and $y = 100$.
With all these limitations the search for the corresponding pixel is limited from the whole image to only a small interval.

Figure 5.4: The cones data set shows us that objects in the right image appear to be translated more to the left than in the left image. Image courtesy of Scharstein et al. [12]

The matching cost values between two pixels are stored in memory in a three dimensional array or *cuboid* as seen in figure 5.5. The dimensions of the cuboid are determined by the dimensions of the reference image and the disparity search range. The difference between pixel (x, y) in the right image and pixel $(x + d, y)$ in the left image is stored at position (x, y, d) in the cuboid.

While running over all pixels to compute their matching cost, for each pixel we keep track of the minimum matching cost that is found and store the related d-value. The d-value with the smallest SAD-value represents the pixel which corresponds the most to the reference pixel. The difference in x coordinates between the reference pixel and the corresponding pixel is used as disparity value.

The times needed to execute the base algorithm with different variable values can be

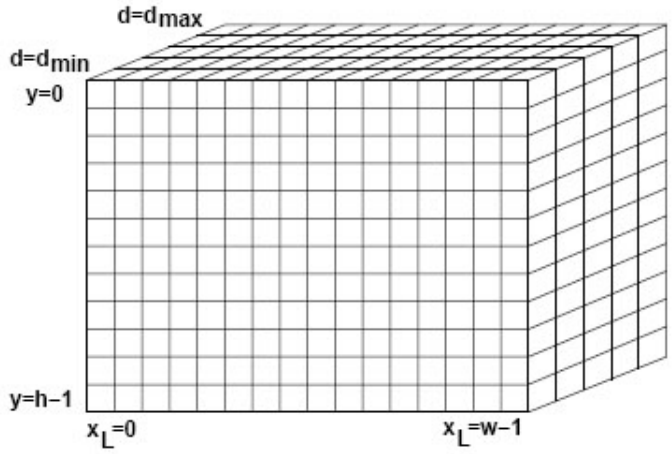

Figure 5.5: A three dimensional array, cuboid, where matching costs are being stored. The value at (x, y, d) represents the matching cost between pixel (x, y) in the reference image and $(x + d, y)$ in the other image. Image courtesy of Mühlmann et al. [13]

found in Table 6.1.

5.2.2 Memory Organization

Mühlmann et al. [13] proposed different ways to fill in the cuboid. The way the cuboid is filled affects the performance of the algorithm. The matching cost has to be computed for all pixels and all depths within a specified interval. To compute this, three loops are needed, one loop for each dimension of the image (x and y) and one loop for different depth values, d.

The best way to organize the loops is to make y the outer loop and d the inner loop. If we would use d as the outer loop we would need two runs through the volume. One run would fill in the volume while the other run is used to find the pixel with the smallest matching cost.

If we use d as the inner loop this can be done in one run. If we fill in the matching cost for (x, y, d) we check it against a variable that holds the current smallest value. If the new value is smaller than this value, we update the variable. When the d loop is done, we have the smallest d value stored in the variable. This value represents the disparity that corresponds that corresponds to the pixel in the reference image with coordinate (x, y).

When the loops or organized like this, advantages of the cache memory are better exploited [13].

The times needed to execute the base algorithm with optimized memory organization can be found in Table 6.1.

5.2.3 Sliding Window

Mühlmann et al. [13] also presented a technique which is called sliding window. This upgrade, just like the previous one, reduces calculation time. It is easy to see that every RGB difference between a pixel in the reference image and a candidate pixel is required in $win_x * win_y$ windows. We can exploit this by storing some data in memory and reusing it, instead of recalculating the difference every time it is needed.
If we assume for now that d is a constant value, we calculate the SAD value for a window around pixel (x, y) and the next window is centered around $(x + 1, y)$.
Figure 5.6 shows the principle of the sliding window in the x-direction. Each square represents the sum of SAD values in a column of the window. The final matching cost is computed by summing up these columns. We can get the correct matching cost for the next pixel by reusing the current matching cost. We do this by simply deducting the first column from the total value and adding the next column to this result.
Until now we have assumed that the d value is constant. But this is not the case in our implementation because the d loop is the inner loop. For this reason we store the necessary data in a two dimensional array. This array stores each column of the current window for each d value. If we now want to compute the matching cost for a pixel we request the matching cost from the previous pixel from the cuboid. Next we compute the matching cost for the new column and add it to the value from the cuboid. We now ask the first column from our two dimensional array and deduct it from our current value and use it as the current matching cost. We remove the column we deducted from the two dimensional array and add the new column so our window is up-to-date again.

The times needed to execute the base algorithm with optimized memory organization and sliding windows can be found in Table 6.1.

5.2.4 Uniqueness

In regions with repetitive or no texture, a local stereo algorithm does not provide a reliable depth estimate. We choose to mark these pixels as unreliable instead of just

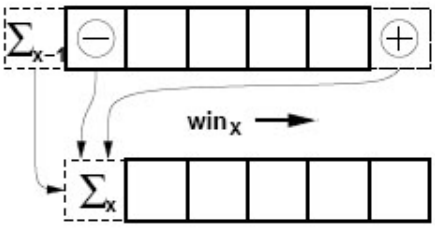

Figure 5.6: Sliding window principle in x-direction. Each square represents the sum of all SAD values in a column of the window. Image courtesy of Mühlmann et al. [13]

giving a wrong value. To determine whether or not such an estimate is reliable or not we determine its uniqueness. This is done by recording the three minimum SAD-values. The lowest SAD-value determines a threshold. If the third value is not higher than this threshold we say it is not unique and mark it. To determine the threshold, 5% of the total colorrange is added to the lowest value. Whenever a value is not unique we do not relate a depth to that pixel because the disparity we would apply to it probably is erroneous. Resulting disparity maps can be found in chapter 6, figure 6.7.

5.2.5 Median Filter

The next upgrade to the base algorithm is to apply a median filter on the acquired disparity map. In contrast to the previous upgrades, the median filter does not upgrade the algorithm's speed but improves the depth map quality. The median filter is a technique to remove salt-and-pepper noise from images. Salt-and-pepper noise refers to pixels which are significantly different from their surrounding pixels. In depth maps these pixels often are wrongly estimated disparity values. By removing these pixels the depth map will appear much smoother. To apply the median filter we run through the image and for each pixel we choose the median of its 3×3 neighbourhood to represents that pixel value. For each pixel the value is determined by the values of 9 pixels. It is possible in our implementation that a pixel is not given a value because it is not unique and we do not relate a disparity to the corresponding pixel. This can be the case in weakly textured regions. If at least 5 of 9 pixel values are given, we determine the median value, else we say the pixel is not unique. The fastest way to determine the median for a pixel is shown in figure 5.7. The dots in this figure represent nodes. Each node has an input of 2 pixel values and outputs the minimum of the 2 to the left and the maximum to the right.

The resulting disparity map can be found in chapter 6, figure 6.8.

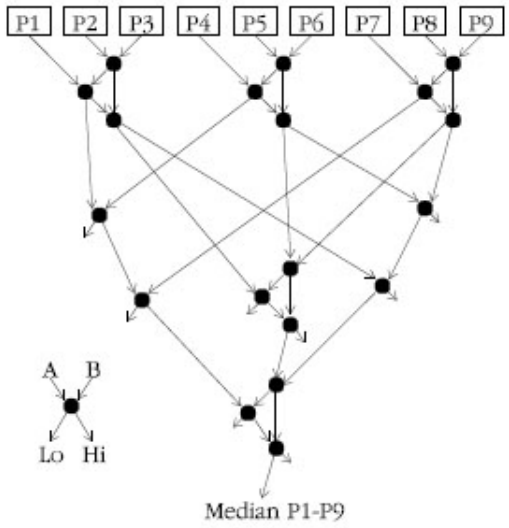

Figure 5.7: Minimum sorting network to calculate the median of nine elements. Image courtesy of Mühlmann [13]

Chapter 6

Results

This chapter shows the resulting depth maps, acquired from the algorithms discussed in section 5. The chapter also compares the performance upgrades, made to the local stereo depth estimation algorithm described in section 5.2.

6.1 Depth From Projection Defocus

The following results are acquired by executing the algorithm as discussed in section 5.1. Figure 6.1 shows us one frame of an image sequence, used to determine depth by using the depth from defocus algorithm. As discussed in section 5.1, multiple images are captured from the scene while being illuminated by the same pattern. The reason why this is done is because the radiance does not remain constant over time. When we take the average over a series of images from the same scene, the radiance difference between different frames will be minimized. Figure 6.2 shows the result of using different amounts of versions of the frame.

Objects that appear darker are positioned closer to the camera. The resulting depth map clearly shows us that the ball is positioned before the boxes. The table appears to be closer to the camera, which is erroneous and artifacts are visible in the lower right corner. The reason for this erroneous data originates from the fact that the table, used to place the object on, is reflective as can be seen in figure 6.1. This means that when we project the illumination pattern on the table, the pattern is reflected by the table. For this reason we can not capture the illumination pattern at those scene points and are unable to construct a defocus kernel for those points. Erroneous pixel depths are also visible in regions where shadows are visible as can be seen at the top of the frontal box.

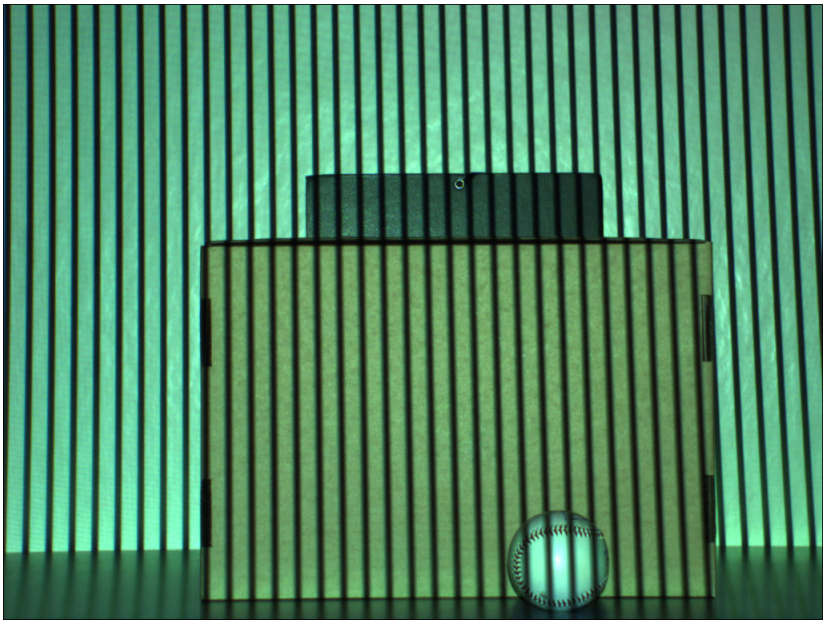

Figure 6.1: Shows the first frame of image sequence used as input to the depth from defocus algorithm.

Another result from the algorithm can be seen in figure 6.3. This figure shows us that the algorithm works well, even for complexer objects like the toy. The figure also shows that the algorithm is not able to determine depth for pixels which appear black, because the illumination of these objects do not provide us with enough information.

Figure 6.4 shows us the effect of using pending objects when using the camera-projector configuration, presented in section 5.1. When the depth values are analyzed we find that in the middle of the pot, pixels appear darker than on the borders of the pot. This follows from the fact that the pot is not a plane but a curved object. This shows us that algorithm is able to detect small differences in depth.

6.2 Stereo Depth Estimation Algorithm

Table 6.1 shows us the execution times of the basic SAD algorithm with different input parameters as discussed in section 5.2.1. This table can be used show us the effects of the performance upgrades, applied to the base algorithm.

Table 6.2 shows us the results of running the SAD algorithm, upgraded with the memory optimization technique which is discussed in section 5.2.2. When we compare

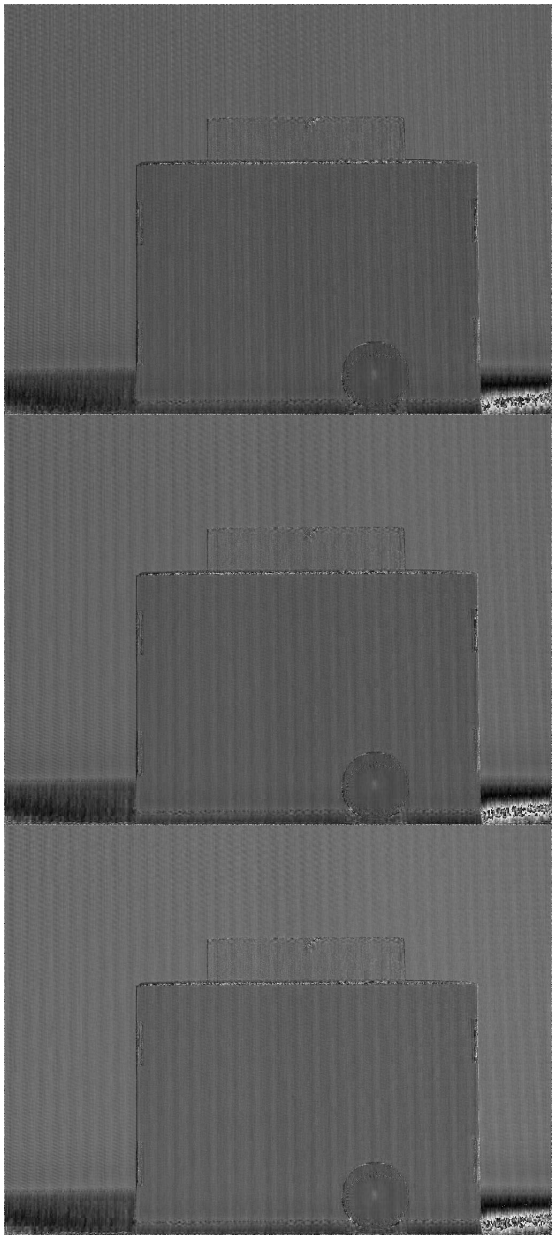

Figure 6.2: This figure shows the results of using different amounts of snapshots from the scene. The left image is the depth, estimated by using only 1 version of each frame. The result in the middle uses 150 version and the right image uses 300 versions.

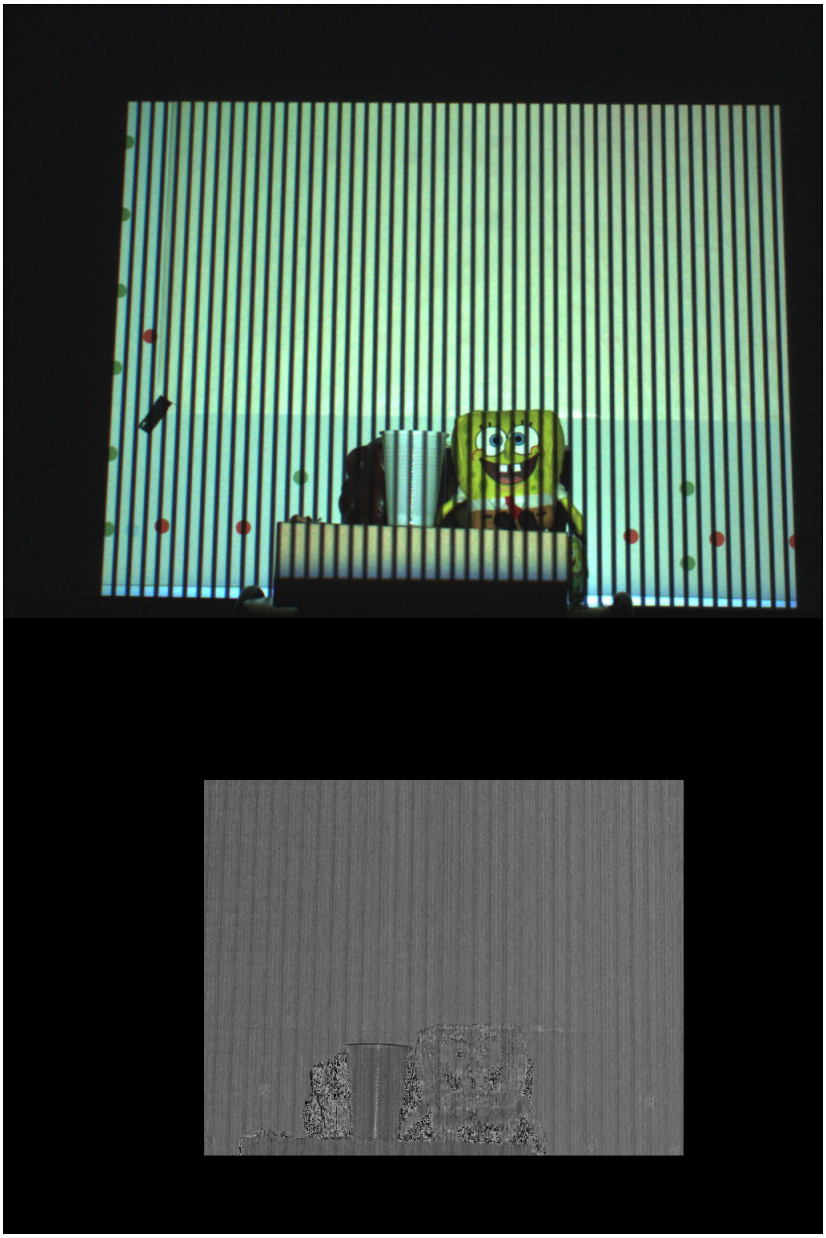

Figure 6.3: This figure shows us the result of using the projection from defocus algorithm when dark object are present in the scene.

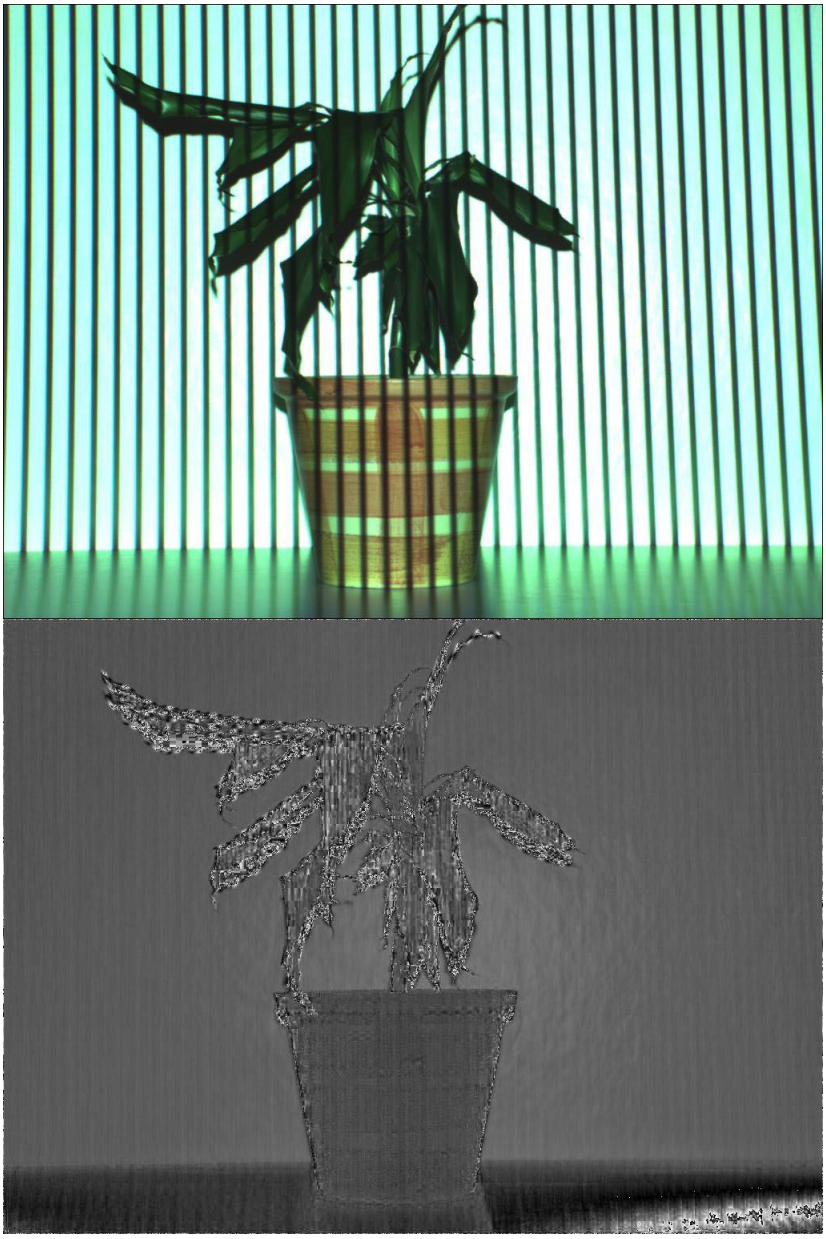

Figure 6.4: This figure shows us the result of using pending and curved objects.

Resolution	Window Size	Depth Range	Time
320x240	9	30	6453
320x240	9	10	2422
320x240	5	30	2375
320x240	5	10	735
160x120	9	30	813
160x120	9	10	422
160x120	5	30	453
160x120	5	10	203

Table 6.1: SAD Algorithm Time Table

Resolution	Window Size	Depth Range	Time
320x240	9	30	3344
320x240	9	10	1344
320x240	5	30	1390
320x240	5	10	531
160x120	9	30	750
160x120	9	10	359
160x120	5	30	344
160x120	5	10	172

Table 6.2: SAD + Memory Organisation (Optimization 1) Time Table

the execution times with table 6.1 we see that it is greatly reduced. This tells us that a good memory organisation is very important.

Table 6.3 shows the results of executing the basic SAD algorithm, upgraded with both the memory organization upgrade and the sliding window technique. When we compare this table with the previous two, it shows us that the upgrade is especially useful when we use larger windows.

By comparing the base algorithm with the fully upgraded algorithm we can see that in some cases the execution time is reduced with approximately 80% without any loss of quality.

The following figures show the resulting disparity maps with resolutions of 320×240 (figure 6.5) and 160×120 (figure 6.6). The execution times needed to compute these depth maps are visible in tables 6.1, 6.2 and 6.3. The upgrades that are applied to the

Resolution	Window Size	Depth Range	Time
320x240	9	30	1297
320x240	9	10	656
320x240	5	30	922
320x240	5	10	453
160x120	9	30	266
160x120	9	10	187
160x120	5	30	234
160x120	5	10	140

Table 6.3: Optimization 1 + Sliding Window (Optimization 2) Time Table

basic algorithm do not influence the resulting disparity map. The depth maps on the left side of the figures shown below are computed by using a depth range of 30 pixels and the ones on the right with a depth range of 10 pixels. The disparity maps at the top of the figures are the result of aggregating over a 9x9 window while the bottom depth maps or the result of aggregating over a 5x5 window.

The objects that appear darker are farther away from the camera. The red pixels in the images represent pixels for which no depth is estimated. When we look at figure 6.5 we see that the algorithm gives reasonably good results. When we use larger windows, the less noise is visible in the images but the object borders appear less sharp.

Figure 6.7 shows us the result of applying the "uniqueness" update. The red pixels represent those for which no disparity is computed. As can be seen more pixels are rejected in textureless areas and no erroneous depths are assigned to these pixels.

The next figure is the result of applying the median filter to the depth map. The application of the median filter takes about 32ms for a disparity map with a resolution of 320x240. The median filter eliminates a certain amount of noise but also makes the object borders appear less sharp.

The times needed to run the algorithms are all calculated on a Intel Pentium 4 2.8 GHZ portable PC with 256MB of RAM.

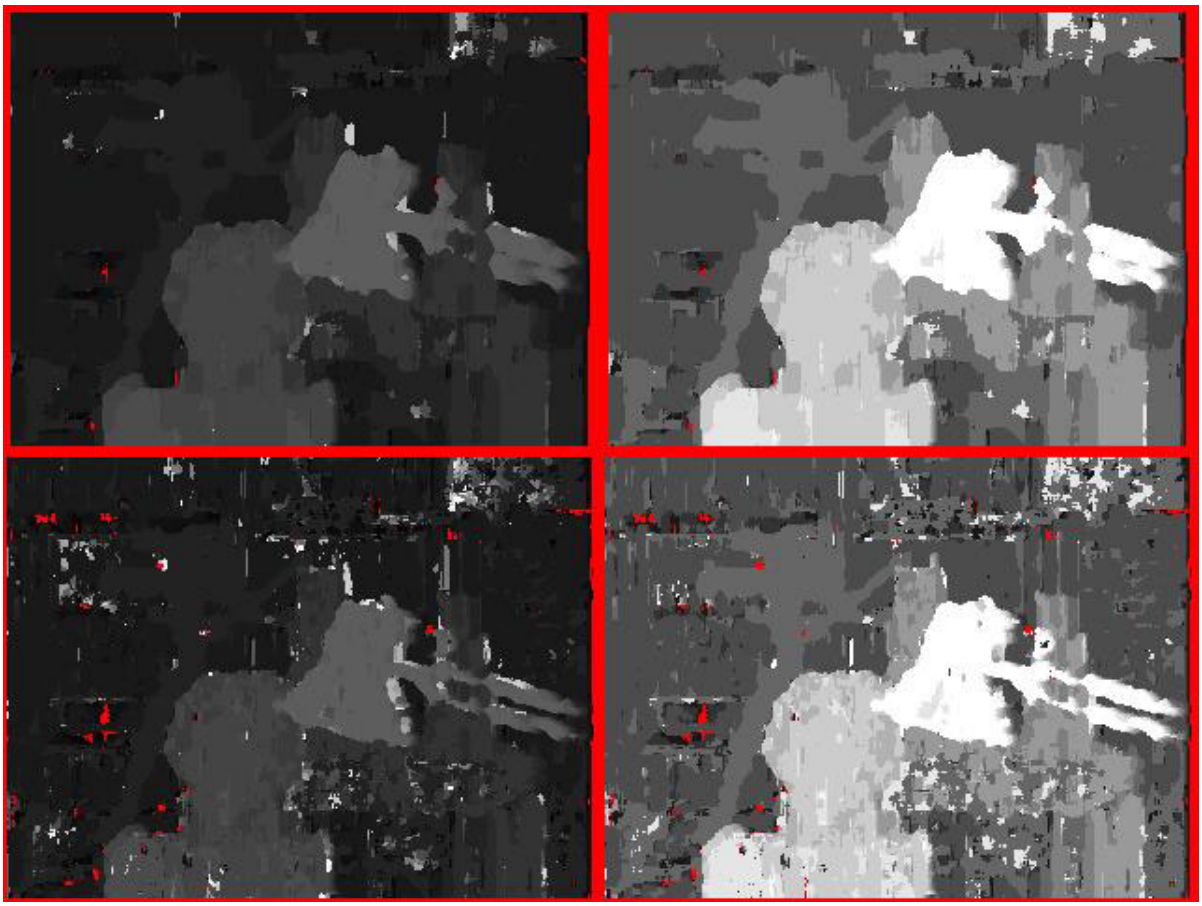

Figure 6.5: Disparity maps from local stereo algorithm with a resolution of 320×240 [left: depth range = 30, right: depth range = 10, top: window size = 9, bottom: window size = 5

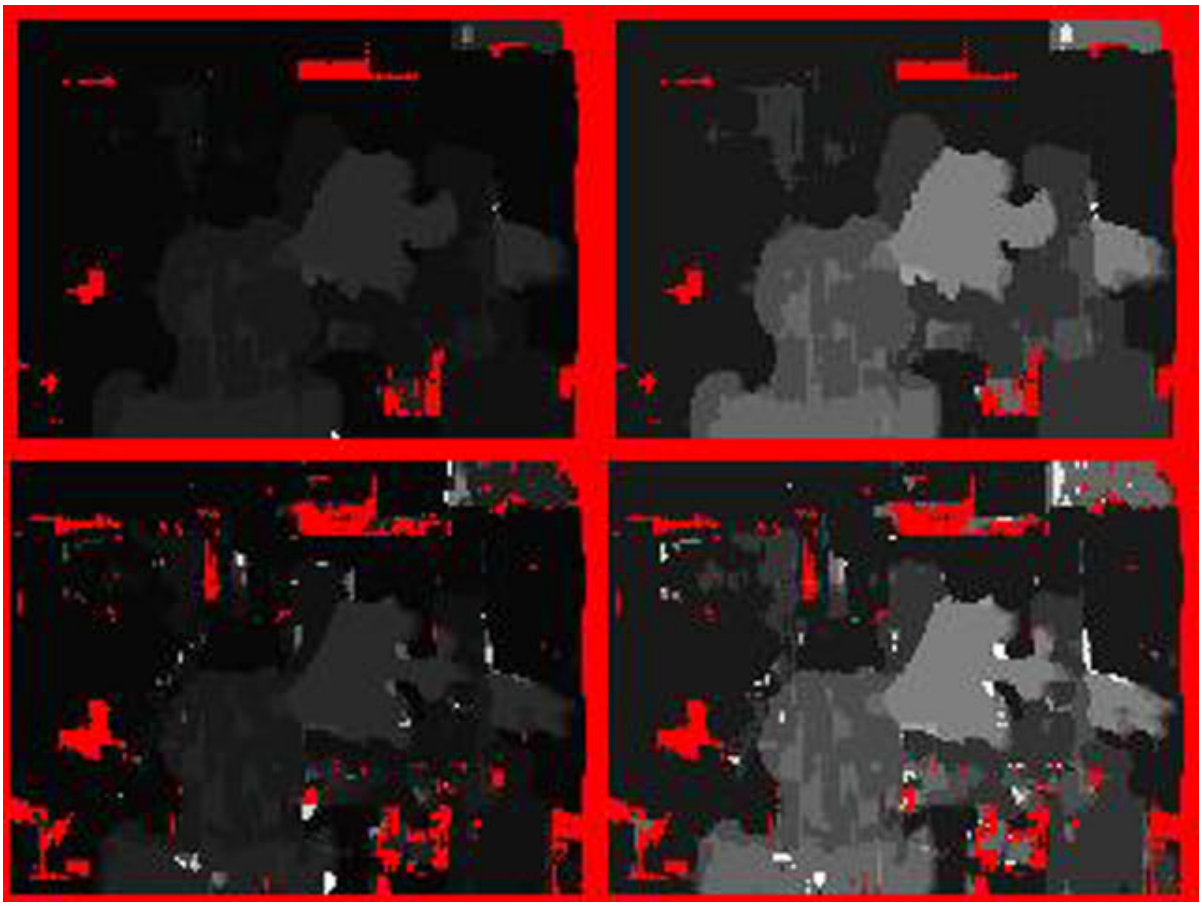

Figure 6.6: Disparity maps from local stereo algorithm with a resolution of 160×120 [left: depth range = 30, right: depth range = 10, top: window size = 9, bottom: window size = 5

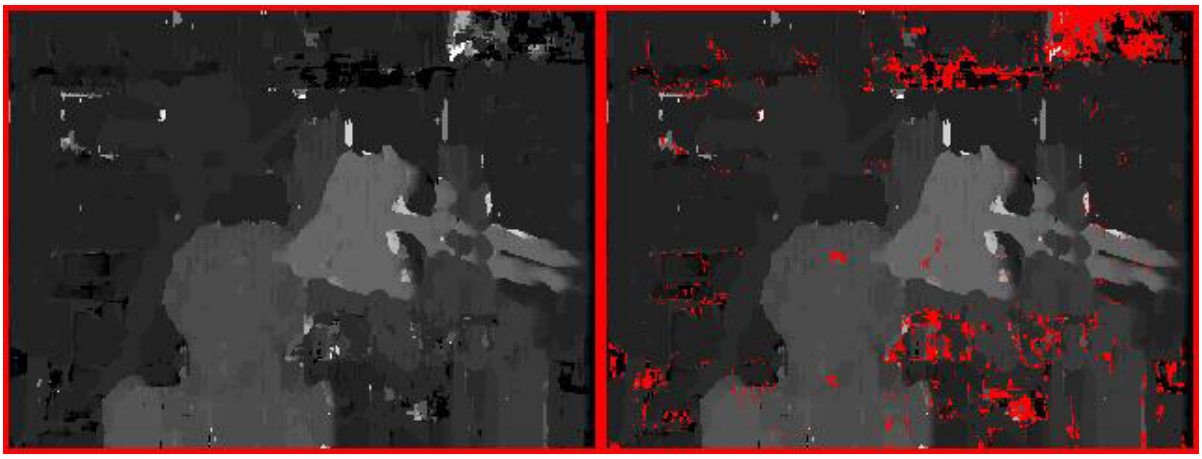

Figure 6.7: Result of applying the 'uniqueness' upgrade. As we can see, in the disparity map to the right, more pixels are rejected rather than applying a faulty depth to them.

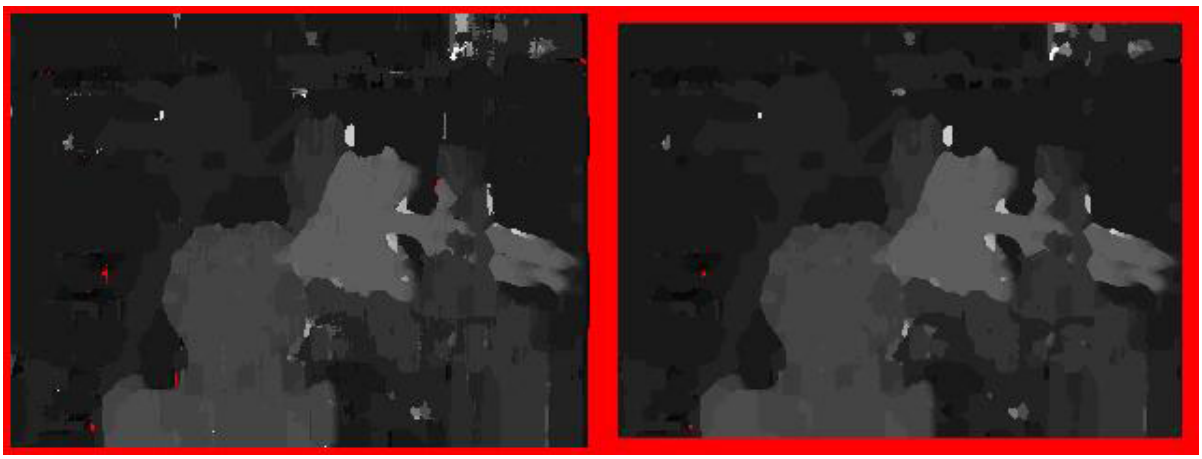

Figure 6.8: Result of applying the 'median filter' upgrade. As can be see, the disparity map on the right is smoother than the one on the left.

Chapter 7

Conclusion And Future Work

In this thesis different types of depth estimation algorithms have been presented. The depth estimation algorithms that were discussed differ in the input data that is used. The wide range of depth algorithms is divided in three major domains.
One domain uses only images from one point-of-view. Extra information can be provided by illuminating the scene by a projector. This information is used to estimate depth for each pixel.
Another domain uses two epipolar rectified images as input and is referred to as stereo view depth estimation. The third domain of depth estimation algorithms uses a sequence of images, taken from different viewpoints.
The stereo view depth estimation algorithms deliver reasonable depth maps with much noise. These algorithms do not perform well near depth discontinuities and textureless regions. Near depth discontinuities, a pixel visible in one image is not visible in the other and makes it impossible to find the correct matching pixel. In textureless regions, we have to many corresponding pixels and it is hard to determine which of these pixels is the correct one. The stereo view depth estimation algorithms allow us to determine depth maps in real-time, so when real-time depth estimation is required, stereo depth algorithms are the best choice.
When we compute depth from a sequence of images we have more information for each pixel in the reference images. When a pixel appears occluded in one image, it may be non-occluded in another. The quality of these depth maps is therefore better than the results from stereo view algorithms.
Depth from projection defocus algorithms are able to determine depth for all pixels in the scene. It does not have trouble near depth discontinuities and textureless regions because it does not depend on pixel correspondence and delivers accurate depth maps.

Problems with these algorithms arise when dark objects are present in the scene. Another problem is to determine depth when the camera and projector are not perfectly aligned.

To improve the results from the depth from defocus algorithm it would be a good idea to implement the mapping from θ to depth as discussed in section 2.5. This will reduce the artifacts visible in the current results. It would also be a good idea to use different types of illumination patterns. As can be seen in the resulting depth maps, vertical lines are visible, this is probably from the fact that we use an illumination pattern that shifts vertical lines. A good idea would be to shift some horizontal lines over the scene as well. An implementation of a global stereo algorithm would be interesting to compare results with the local algorithm and compare execution times.

Bibliography

[1] L. Zhang and S. K. Nayar. Projection Defocus Analysis for Scene Capture qnd Image Display. *ACM Trans. on Graphics (also Proc. of ACM SIGGRAPH)*, Jul 2006.

[2] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

[3] S. B. Kang and R. Szeliski. Extracting view-dependent depth maps from a collection of images. *Int. J. Comput. Vision*, 58(2):139–163, 2004.

[4] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. In *ICCV (1)*, pages 377–384, 1999.

[5] S. McCloskey, M. Langer, and K. Siddiqi. The Reverse Projection Correlation Principle for Depth from Defocus. *Proceedings of the 3rd International Symposium on 3D Data Processing, Visualization and Transmission*, 2006.

[6] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[7] P. Favaro. Depth from focus/defocus, June 2002.

[8] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. *cvpr*, 01:195, 2003.

[9] J. Salvi, J. Pages, and J. Batlle. Pattern codification strategies in structured light systems. *Pattern Recognition*, 2004.

[10] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2003.

[11] L. McMillan. *An Image-Based Approach To Three-Dimensional Computer Graphics*. PhD thesis, University of North Carolina, 1997.

[12] D. Scharstein, R. Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms, Dec 2001.

[13] K. Mühlmann, D. Maier, J. Hesser, and R. Männer. Calculating dense disparity maps from color stereo images, an efficient implementation.

[14] S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms, 2006.

[15] D. Le Gall. Mpeg: A video compression standard for multimedia applications. *Commun. ACM*, 34(4):46–58, 1991.

[16] D. W. Fanning. Convert rgb image to grayscale, 2002.

Auteursrechterlijke overeenkomst

Opdat de Universiteit Hasselt uw eindverhandeling wereldwijd kan reproduceren, vertalen en distribueren is uw akkoord voor deze overeenkomst noodzakelijk. Gelieve de tijd te nemen om deze overeenkomst door te nemen, de gevraagde informatie in te vullen (en de overeenkomst te ondertekenen en af te geven).

Ik/wij verlenen het wereldwijde auteursrecht voor de ingediende eindverhandeling:
Multi-View Depth Estimation
Richting: **Master in de informatica** Jaar: **2007**
in alle mogelijke mediaformaten, - bestaande en in de toekomst te ontwikkelen - , aan de Universiteit Hasselt.

Niet tegenstaand deze toekenning van het auteursrecht aan de Universiteit Hasselt behoud ik als auteur het recht om de eindverhandeling, - in zijn geheel of gedeeltelijk -, vrij te reproduceren, (her)publiceren of distribueren zonder de toelating te moeten verkrijgen van de Universiteit Hasselt.

Ik bevestig dat de eindverhandeling mijn origineel werk is, en dat ik het recht heb om de rechten te verlenen die in deze overeenkomst worden beschreven. Ik verklaar tevens dat de eindverhandeling, naar mijn weten, het auteursrecht van anderen niet overtreedt.

Ik verklaar tevens dat ik voor het materiaal in de eindverhandeling dat beschermd wordt door het auteursrecht, de nodige toelatingen heb verkregen zodat ik deze ook aan de Universiteit Hasselt kan overdragen en dat dit duidelijk in de tekst en inhoud van de eindverhandeling werd genotificeerd.

Universiteit Hasselt zal mij als auteur(s) van de eindverhandeling identificeren en zal geen wijzigingen aanbrengen aan de eindverhandeling, uitgezonderd deze toegelaten door deze overeenkomst.

Ik ga akkoord,

Stijn Brouwers

Datum: **22.05.2007**