UNIVERSITY OF OULU

FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

Heikki Kaarlela

EDGE ADAPTIVE FILTERING OF DEPTH MAPS FOR MOBILE DEVICES

Master's Thesis
Degree Programme in Computer Science and Engineering
October 2019

Kaarlela H. (2019) Edge adaptive filtering of depth maps for mobile devices. University of Oulu, Degree Programme in Computer Science and Engineering. Master's thesis, 57 p.

ABSTRACT

Mobile phone cameras have an almost unlimited depth of field, and therefore the images captured with them have wide areas in focus. When the depth of field is digitally manipulated through image processing, accurate perception of depth in a captured scene is important.

Capturing depth data requires advanced imaging methods. In case a stereo lens system is used, depth information is calculated from the disparities between stereo frames. The resulting depth map is often noisy or doesn't have information for every pixel. Therefore it has to be filtered before it is used for emphasizing depth. Edges must be taken into account in this process to create natural-looking shallow depth of field images.

In this study five filtering methods are compared with each other. The main focus is the Fast Bilateral Solver, because of its novelty and high reported quality. Mobile imaging requires fast filtering in uncontrolled environments, so optimizing the processing time of the filters is essential.

In the evaluations the depth maps are filtered, and the quality and the speed is determined for every method. The results show that the Fast Bilateral Solver filters the depth maps well, and can handle noisy depth maps better than the other evaluated methods. However, in mobile imaging it is slow and needs further optimization.

Keywords: photography, stereo camera, depth perception, Fast Global Smoother, bilateral grid, Fast Bilateral Solver

Kaarlela H. (2019) Reunatietoinen syvyyskarttojen suodatus mobiililaitteilla. Oulun yliopisto, tietotekniikan tutkinto-ohjelma. Diplomityö, 57 s.

TIIVISTELMÄ

Matkapuhelimien kameroissa on lähes rajoittamaton syväterävyysalue, ja siksi niillä otetuissa kuvissa laajat alueet näkyvät tarkennettuina. Digitaalisessa syvyysterävyysalueen muokkauksessa tarvitaan luotettava syvyystieto.

Syvyysdatan hankinta vaatii edistyneitä kuvausmenetelmiä. Käytettäessä stereokameroita syvyystieto lasketaan kuvien välisistä dispariteeteista. Tuloksena syntyvä syvyyskartta on usein kohinainen, tai se ei sisällä syvyystietoa joka pikselille. Tästä syystä se on suodatettava ennen käyttöä syvyyden korostamiseen. Tässä prosessissa reunat ovat otettava huomioon, jotta saadaan luotua luonnollisen näköisiä kapean syväterävyysalueen kuvia.

Tässä tutkimuksessa verrataan viittä suodatusmenetelmää keskenään. Eniten keskitytään nopeaan bilateraaliseen ratkaisijaan, johtuen sen uutuudesta ja korkeasta tuloksen laadusta. Mobiililaitteella kuvantamisen vaatimuksena on nopea suodatus hallitsemattomissa olosuhteissa, joten suodattimien prosessointiajan optimointi on erittäin tärkeää.

Vertailuissa syvyyskuvat suodatetaan ja suodatuksen laatu ja nopeus mitataan jokaiselle menetelmälle. Tulokset osoittavat, että nopea bilateraalinen ratkaisija suodattaa syvyyskarttoja hyvin ja osaa käsitellä kohinaisia syvyyskarttoja paremmin kuin muut tarkastellut menetelmät. Mobiilikuvantamiseen se on kuitenkin hidas ja tarvitsee pidemmälle menevää optimointia.

Avainsanat: valokuvaus, stereokamera, syvyyden havaitseminen, nopea globaali siloittaja, bilateraalinen ruudukko, nopea bilateraalinen ratkaisija

TABLE OF CONTENTS

FOREWORD

Sometimes you need more than one view of the world to capture its depth.

This thesis was written to examine the simulation of depth in photos captured with mobile cameras. It's one of the milestones on the road to achieve a DSLR-level photo quality with a phone. Maybe soon everything works automatically in the software, you just have to point a phone at something and it will give you the best image, tailored for your preferences. But before that there are some roadbumps.

This project was a diploma thesis position at Visidon Oy. Working with the thesis here was a smooth experience, as the project progressed steadily without any major setbacks. Thanks are in order to my coworkers, and especially to the thesis supervisor Olli Silvén and the technological advisor Jari Hannuksela.

I would also like to thank those who have supported me during all these years on my way to this point. I was born to the best family, so that was a good start. And my friends at the various student organizations were a constant source of support, especially during this year of working with the thesis. They made sure that I also learnt a lot about life and people during my active student years. Love you, go to work, let's put everything on the red.

Oulu, Finland, 16th October, 2019

Heikki Kaarlela

ABBREVIATIONS

CMOS	Complementary Metal Oxide Semiconductor
RBG	Red-Green-Blue -color model
YUV	Luminance (Y) and chrominance (UV) color model
D (e.g. 2-D)	Dimension
SAD	Sum of Absolute Differences
PDAF	Phase-Detect Auto-Focus
LIDAR	Light Detection and Ranging
SIFT	Scale Invariant Feature Transform
SURF	Speeded-Up Robust Features
ORB	Oriented FAST and Rotated BRIEF
FAST	Features from Accelerated and Segments Test
BRIEF	Binary Robust Independent Elementary Feature
SLR	Single-lens Reflex Camera
AR	Augmented Reality
m/s	Meters per second
IEEE	Institute of Electrical and Electronics Engineers
FGS	Fast Global Smoother
WLS	Weighted Least Square
FBS	Fast Bilateral Solver
VR	Virtual Reality
MATLAB	Matrix Laboratory
Mpx	Megapixels
SoC	System-on-a-Chip
CPU	Central Processing Unit
GPU	Graphics Processing Unit
SIMD	Single instruction, multiple data
MPI	Max Planck Institute
KITTI	Karlsruhe Institute of Technology and Toyota Technological Institute
ETH3D	Eidgenössische Technische Hochschule Zürich's 3-D Benchmark
png	Portable Network Graphics
pfm	Portable Floatmap
RMSE	Root mean square error
MAE	Mean absolute error
VIF	Visual Information Fidelity
NV21	A YUV-format with subsampled and interleaved U/V-chroma planes
MC-CNN	Matching Cost with a Convolutional Neural Network
HD	High-definition
4K	Horizontal resolution of approximately 4,000 pixels

1. INTRODUCTION

The number of photos taken each year has grown during the era of smartphones, with over a trillion photos taken in 2017. When people take photos, they want to capture an aesthetically good-looking 2-D image from the 3-D world within the camera's vision. While aesthetics are usually in the eye of the beholder, there are guidelines to accomplish this. The common way to frame an image is to focus on certain parts of the image, which means blurring the other parts depending on their distance to the focused parts. This way of framing also simulates the human vision. Human eyesight sees clearly only from the middle of the eye and compensates for the poorer peripheral vision by saccadid eye movements which scan the whole scene. The result is that only a part of the vision is in focus at any time [1].

A weakness inherent to the short focal length of smartphone cameras can be that everything in the image is in focus, taking attention away from the intended target. Photography uses the depth of field as a way to get artistic foreground and background handling. A proper camera lens with a possibility for a large aperture can create a suitable small depth of field for a portrait, so the target is in focus and the rest of the picture is not. There's a term, bokeh, coming from a Japanese word meaning "out-of-focus blur", which is used to describe the way the lens renders these out-of-focus points of light. However, phone cameras usually have thin lenses which can't create a shallow depth of field optically. In that case it has to be simulated in post-processing to acquire the desired effect.

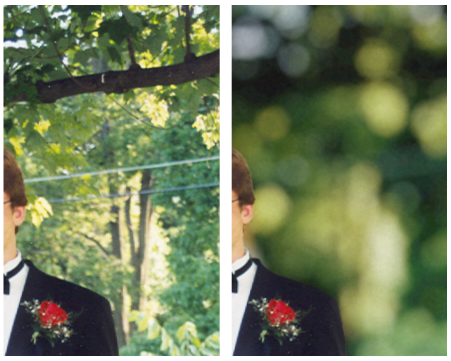


Figure 1. An example of simulating bokeh to the image in post-processing (Pictures from user BenFrantzDale at Wikipedia Commons[1].

There are ways to modify a photo taken by a phone camera so that it simulates a shallow depth of field. With the current growing trend of machine learning, one way is to use the machine to recognize which parts of the image should be in focus (like a person in a portrait) and blurring other parts. Other ways depend on capturing or finding depth information for the captured image. With a single objective camera, the depth information can be captured using other sensors with the camera. There's also a new method of dual-pixel disparity that uses the camera's autofocus feature to count the disparity for the pixels of the image.

[1]https://commons.wikimedia.org/wiki/File:Faux-bokeh-original.png, licensed under CC BY-SA 3.0, accessed 7.10.2019, and https://commons.wikimedia.org/wiki/File:Faux-bokeh-final.png, licensed under CC BY-SA 3.0, accessed 7.10.2019.

Starting with the HTC Evo 3D in 2011, there have been smartphones with two lenses for capturing a shallower depth of field. Nowadays phones with even more lenses have been released. Having multi-lenses is beneficial not just for the depth of field but also for the high dynamic ranges and the level of detail captured, as the information from the captures can be united for a more high-quality and aestethic image. The depth information can be calculated by using two pictures captured from different angles. However, the resulting disparity map can be very noisy and need further processing so it's smooth and the end result doesn't have in-focus or out-of-focus parts where it should not. Also, the border between an object in focus and an object not in focus has to be sharp to look natural, so this smoothing needs to be edge aware. Figure 2 shows the importance of smoothing (Original images from the dataset by Canessa et al. [2]).

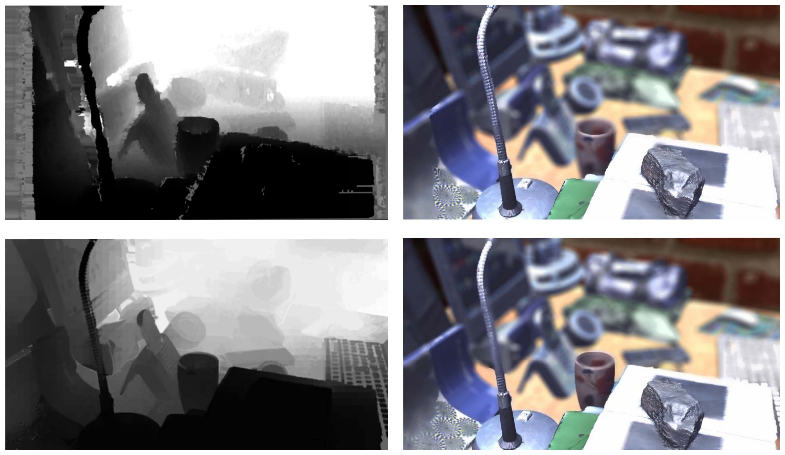


Figure 2. A comparison showing the difference between an unfiltered diversity map and its final result (above), and one filtered with Fast Bilateral Solver and its final result (below).

The goal of this thesis is to evaluate methods to achieve this edge-aware smoothing of depth maps, with the smartphone cameras being the focused use. There are several methods already in wider use, which will be discussed further in the following chapters. The end goal is to have a clear and accurate depth map for photo post-processing, so the end result is an aesthetically good-looking image. However, processing a depth map this way adds processing time to the algorithm, so part of the goal is to determine if the methods are worth it on smartphones, where the processing of the images shouldn't encumber the photo capturing too much.

Chapter 2 of this thesis recounts the principles of capturing depth information while taking a photo. Chapter 3 is about how that depth information is used to enhance images. Chapters 4 and 5 present the different methods that are used to enhance the calculated depth map and form the main focus of this thesis. These methods are linear and bilateral, and they are evaluated in Chapter 6, which presents the gathered results. The last chapters discuss and conclude these results.

2. DIGITAL PHOTOGRAPHY WITH DEPTH PERCEPTION

Photography is about capturing the light in a controlled way. The light refracts while going through a lens and focuses on to an image plane containing light-sensitive material. In the case of digital photography, this material is made of CMOS sensors, which perform image processing and output the image information as digital values [3].

The process of light being let to the image plane through a quick opening and closing of the shutter is called exposure. The light photons entering the camera during the exposure go to an array of cavities. The intensity of photons in a cavity determines the value of the pixel. The color images need at least three color samples for each pixel. This is accomplished by filtering the light so that only the specific wavelengths get through to specific light cavities, as seen in Figure 3. When using the Bayer RBG array pattern, the color green has a larger sample rate than blue or red, as the human eye is more sensitive to the green wavelengths of the visible spectrum. This helps to reduce the noise. This also means that half of the cavities capture the green color, and the remaining two quarters capture red and blue. As every pixel now only comprises of one color, the rest of the color values for the pixels are estimated by a demosaicking process. There are other possible patterns, like cyan-magenta-yellow-green (CMYG), and also dual-pixel systems which are discussed in Section 2.2 [4].

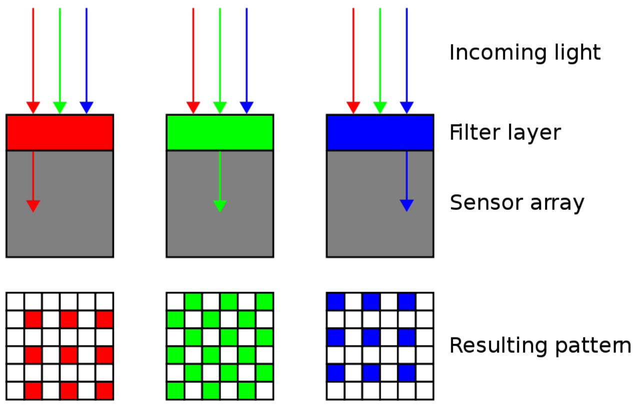


Figure 3. Filtering light to Bayer pattern array (Picture from user Cburnett at Wikimedia Commons[1]).

The lenses are used to refract the light, and they have a big impact on the captured image. Focal point of a lens is the point where the light rays originated from the object converge. When taking a picture, the region of interest is desired to be in this distance, so it shows up in focus in the captured image. Focal length is the distance from the lens to the focal point, and focal plane means the 2-D plane in the focal length's distance from the lens. Outside of this focal plane the objects get more and more blurry the farther they are, but on both sides of the focal plane there is an area which will be sharp enough in the image. This area is called the depth of field, and it can be shallow or wide.

[1]https://commons.wikimedia.org/wiki/File:Bayer_pattern_on_sensor_profile.svg, licensed under CC BY-SA 3.0, accessed 7.10.2019.

A lensless pinhole camera, *camera obscura*, has just a small hole for the light to get in, and so it has an almost infinitely wide depth of field and everything appears in focus. So the smaller the hole, the wider the depth of field is. A lens' size can't be changed without changing the lens itself, but with aperture blades it can be partly covered. The aperture blades cover the lens from its sides so that light only gets to the lens through a hole in the cover. Aperture, or aperture stop, is this hole. The size of the aperture stop affects the depth of field. This size is usually expressed as a ratio of lens focal length to aperture diameter, or an f-number which is the ratio's denominator and so a larger f-number means a smaller hole. A large aperture hole is needed for a shallow depth of field. A smaller one has a larger depth of field and so a larger area of the image is in focus. Figure 4 shows these effects.

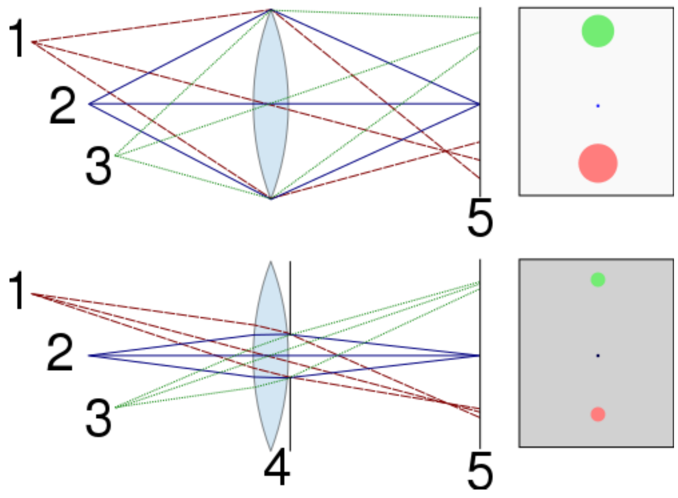


Figure 4. Effect of aperture on blur and the depth of field (Picture from user Ben-FrantzDale at Wikimedia Commons[1]).

In the upper image of Figure 4, the aperture hole is large and the f-number small. This depth of field is shallow and so points of light closer and farther than 2 have large circles of confusion. These show up in the photo as large discs. With a smaller aperture hole, like in the lower image, the defocused points create smaller circles of confusions. This makes for a wider depth of field, as the smallest discs are indistinguishable from points and so they don't blur the image.

Equation 1 shows how the depth of field T is affected by the focused distance u, f-number N, the circle of confusion C and the focal length f of the lens [3].

$$T = 2u^2NC/f^2 \tag{1}$$

2.1. Capturing depth information

Depth map of the picture can be acquired by several different methods, depending on the available hardware [5]. The focus of this thesis is in the stereo-lens method,

[1]https://commons.wikimedia.org/wiki/File:Depth_of_field_illustration.svg, licensed under CC BY-SA 3.0, accessed 7.10.2019.

but other methods are discussed in brief. Multiple lenses capture multiple frames at the same time, whereas using one lens needs either some supplementary information about the captured frame, multiple frames taken with the same lens, or intelligence to recognize what's in the image and segment it.

One lens can be used to capture several photos from different angles by using user action, like in Google's Lens Blur method [6], where the user moves the camera while capturing frames. The visual features of these frames are then tracked across the images, and their 3D positions are computed. The differences in positions are found by using the Sum of Absolute Differences (SAD) [7], and the most similar pixels are used to create an optimized depth map. The problem with this method is that the user has to work extra and move the phone themselves. Google has since launched Portrait Mode, which uses machine learning -powered semantic segmentation and dual-pixel autofocus [8]. This method is described in detail in Section 2.2.

Semantic segmentation works by dividing the picture into the focused area, like a person, and blurring everything else. This makes the resulting out-of-focus areas have a constant blur, instead of one that changes magnitude depending on its distance to the region of interest. It may also look unnatural, if there are other parts of the image within an equal distance to the focused part but the segmentation doesn't take them into account. For example, an in-focus person standing next to a completely blurred car.

2.2. Single lens method using autofocus

Wadhwa et al. introduced a system which combines portrait segmentation with dual-pixel -autofocus, or uses just one of them when using both is not feasible. Google has used this method on single lens cameras, when the portrait segmentation by itself doesn't give adequate results [8].

Dual-pixel -autofocus is used in cameras to automatically focus on the region of interest. It adjusts the lens position until the disparity value in the focus region is zero. In this Phase-Detect Auto-Focus (PDAF) technology the region of interest is determined based on the high-level objective, which can depend on the user input and face recognition [9]. The light rays entering the image sensor are divided into subpixels, depending on which direction they are coming from, left or right. This forms two phase images which have only detected light from one side. When the object in the camera's view is behind the lens' focal plane, the left phase image is a right-shifted version of the right phase image. When the object is closer than the focal plane, the left phase image is a left-shifted version of the right phase image. The phase shifts' magnitude indicates the distance to the focal point [10]. If the focal point is not in the region of interest, autofocus changes the lens position and iterates this until it is.

In a dual-pixel sensor each pixel is similarly split in two, as shown in the Figure 5[1]. The light emitted by the out-of-focus object is refracted through the lens with the aperture L to the sensor, which is farther than the focal length distance D_{i}. Half of the light is refracted on the left half of the image and half to the right side. This splits

[1]Figure republished with permission of Association for Computing Machinery, from Synthetic depth-of- field with a single-camera mobile phone by Wadhwa et al, ACM Transactions on Graphics Volume 37 Issue 4, August 2018; permission conveyed through Copyright Clearance Center, Inc.

the image into two halves (summing these halves together produces an image which doesn't differ much from an image taken without dual pixels). The sides have differences, with the disparities proportional to the blur size on the sensor. Compared to a stereo-lens view these disparities are very slight. This is because their viewpoints are only the diameter of the lens from each other, which is a small distance in smartphone cameras [11].

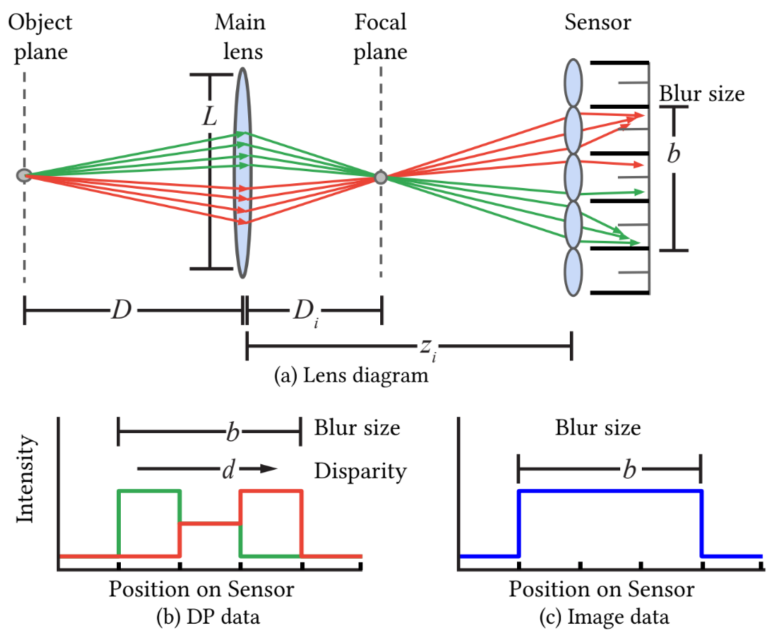


Figure 5. A thin lens model showing the relationship between depth D, blur diameter b, and disparity d on a dual-pixel sensor.

Dual-pixel captures can be compared to light field image cameras, which capture not just the 2-D image but also the direction of where the light is coming from. In a lenslet light field camera a microlens array is placed in front of the photosensor array. Each microlens in the array forms an image of the main lens aperture from a different angle, and from their displacements the depth can be calculated [12, 13]. However, a dual-pixel sensor creates only a two-angle light field, which is not enough for light view methods to calculate depth sufficiently. Using more angles decreases the spatial resolution of the image, as the pixels are divided between the angles.

Estimating depth using a dual-pixel method has several advantages compared to a multi-lens stereo. As there is only one capture from one lens, the compared elements have the same white balances and distances. The use of fewer lenses mean that the material costs are lower. The resulting disparities are also very small, which makes for a small search range. But this can also be a disadvantage, as the sub-pixel precision is vulnerable to noise, especially in low-light environments [14]. There is also an ambiguity created by the unknown scale and offset, because the focus distance varies. To counter this affine ambiguity, Gaug et al. have presented a method which incorporates convolutional neural network training to dual-pixel depth estimation [15].

2.3. Supplementary sensors

Other sensors can be used in addition to the single light-capturing camera. Mobile depth sensor systems can be grouped into passive and active systems. Stereo vision is a popular passive method. This means a system consisting of a stereo camera which determines the depth by triangulation. It is discussed in depth below. A simple passive stereo system can struggle with extracting depth information from texture-less surfaces, like one-color walls. Active systems have an active illumination of the object and include structured light and time-of-flight techniques [16].

In structured light systems there's a projector projecting additional texture to the view, and typically one camera to capture the image. More cameras are also possible, as this texture can also be used to improve the reliability of stereo vision [17]. Structured light systems can be divided into spatially and temporally structured light. Spatial techniques use a single pattern for single shot depth estimation. Temporal techniques take multiple captures while varying the projected dynamic pattern. This sequential projection makes for a uniquely encoded signal, from which estimating depth is quick to compute, but taking many captures at different times creates motion artefacts [18]. Sequential projection techniques use temporally changing binary and grey patterns and phase shifting, whereas single-shot spatially varying techniques use color information or unique segment patterns in the light projection [19].

The time-of-flight sensors measure the depth directly, by sending a pulse of light and measuring the time it takes for the light to travel to the object and back. The sinusoidally modulated light wave is emitted and the distance information is gathered from the phase delay between the original and the received light signal [20]. These sensors supplement a camera capture, and have low resolution compared to the images captured by the cameras. Some pulses used in directly measuring the depth are infrared light, sonar and Light Detection and Ranging (LIDAR). Infrared is the one used in smartphones.

The common disadvantage of these active sensor systems is that they are often expensive, power-hungry or limited to certain environments [15]. Structured light and time-of-flight sensors may have a hard time decoding the received signal if the background light, reflections or other signals have blended into it [16]. Their working distance is also small for outdoor shooting, as infrared impulses have safety restrictions.

2.4. Stereo vision

Stereo camera systems have become more common in smartphones, and they are the main focus of this thesis. Stereo view is a passive method to capture depth, and it simulates the human depth perception. It has two cameras with some distance from each other, and both capture an image at the same time. The differences in these two images are used to estimate the depth map. Figure 6 shows the pieces of this depth map estimation process.

Epipolar geometry formulates the relationship between the cameras and the scene. Figure 7 describes the desired setup. The two cameras are set some distance from each other, and the baseline is this distance between their center points, O_L and O_R. X is a

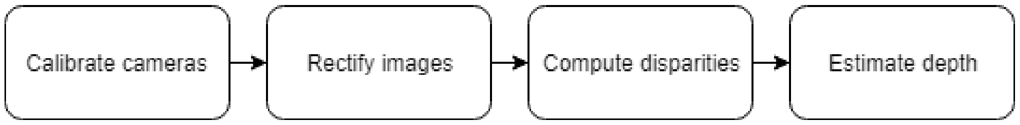


Figure 6. A simple pipeline for creating a depth map from stereo camera capture.

point in the captured scene, and it's projected onto the image planes as points x_L and x_R. All these points lie in the same epipolar plane. Epipolar line is the line on the plane that is parallel with the image plane (line e_R-x_R in the figure). The points e_L and e_R are in the intersection of the baseline and the image plane, and so are epipoles. As the projection of X on the left image plane, x_L, is on an epipolar plane, the projection x_R is on the epipolar line on the right image plane. So the same point is located on both epipolar lines of the same plane. This is the epipolar constraint. Now, using triangulation, when the locations of x_L and x_R are known, the location (and so the distance) of X can be calculated [21, 22].

This geometry of epipolar lines needs a calibrated setup and rectified image planes. This allows for a quick epipolar line checkup in disparity computation. The point x_R, corresponding to the point x_L, should be found from the epipolar line. If it is not recognized, the distance X can't be reliably calculated, and any of the points X_1, X_2 or X_3 could also be a right one. This is a correspondence problem and further described below.

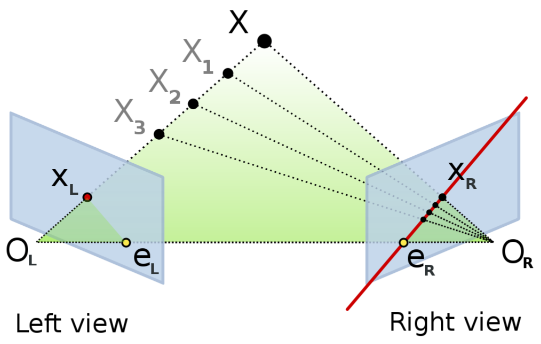


Figure 7. Epipolar geometry (Picture from user Arne Nordmann at Wikimedia Commons[1]).

2.4.1. Camera calibration

The two cameras of the stereo system have to be calibrated, meaning that their internal and external parameters are estimated. Internal parameters describe how the camera forms an image (these parameters include focal length, optical center of the lens, pixel sizes and radial distortion coefficients, among others), and the external parameters

[1]https://commons.wikimedia.org/wiki/File:Epipolar_geometry.svg, licensed under CC BY-SA 3.0, accessed 7.10.2019.

define the camera's position and orientation [23]. In a stereo system the external parameters also describe the relations between the cameras. Together they form a model of the correspondence between the real world coordinates and image coordinates.

This model is calculated to diminish systematic distortions in the image, caused by the lenses. These distortions include radial and decentering distortion. Radial lens distortion displaces image points radially in the image plane, creating changes in image magnification depending on the distance to the optical axis (e.g. barrel distortion). Decentering distortion is caused by mismatching centers of curvatures of lens surfaces. Modeling these distortions reduces them in the captured images [24].

2.4.2. Image rectification

After the cameras have been calibrated, the two frames are captured by exposing the pixel cavities for a short time. The differences in the shooting angle causes the objects to be in different positions in the horizontal dimension (or vertical if the two lenses are set vertically, like when the phone is held in 90 degrees). If the stereo camera captures the frames at the same time, the differences caused by the movements and the elapse of time are minimal, but there are other ways for the frames to have considerable differences. Since the perspective is different, the object's surface can be foreshortened in the other image and so it doesn't have the same measurements for stereo matching [25]. Another difference can be in the lenses themselves. The focal length of the lens determines the angle of view, so if the two lenses have different focal lengths, their view of the target may be different.

Before the disparity calculation, the images are transformed to minimize the extraneous differences. In the case of different focal lengths, the image with a smaller focal length (and which so looks like it's shot from farther away) is cropped and warped to fit the other image. This rectification process determines the transform so that the epipolar lines become collinear and parallel to the image axis [26]. This helps with the comparisons, as all the corresponding parts of the picture should now be in the same scanline. Figure 8 shows an example. The orange image planes have misalignment. The yellow image planes are after the rectification, and have collinear epipolar lines, allowing for matching pixels to be in one scanline. The image borders show signs of warping.

2.4.3. Disparity computation and depth estimation

In a simple stereo system, or after rectification, the image planes are parallel to each other and the lenses' focal lengths are the same, and so the epipolar lines coincide as well. This is a simple case of calculating the disparity by triangulation. Disparity is the inverse of the depth, being larger when the objects are closer and there's a larger difference between the frames. In Figure 9 the difference between $X1$ and $X1'$ is larger than between $X2$ and $X2'$. The relative depth $Z1$ can be calculated with the following formula:

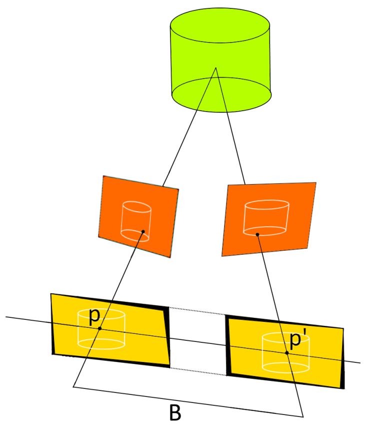


Figure 8. Rectification.

$$Z1 = \frac{B * F}{X1 - X1'} \tag{2}$$

where B is the baseline of the camera, F is the focal length and $X1$ and $X1'$ are the lines' x-axis distance at focal length from O and O', respectively.

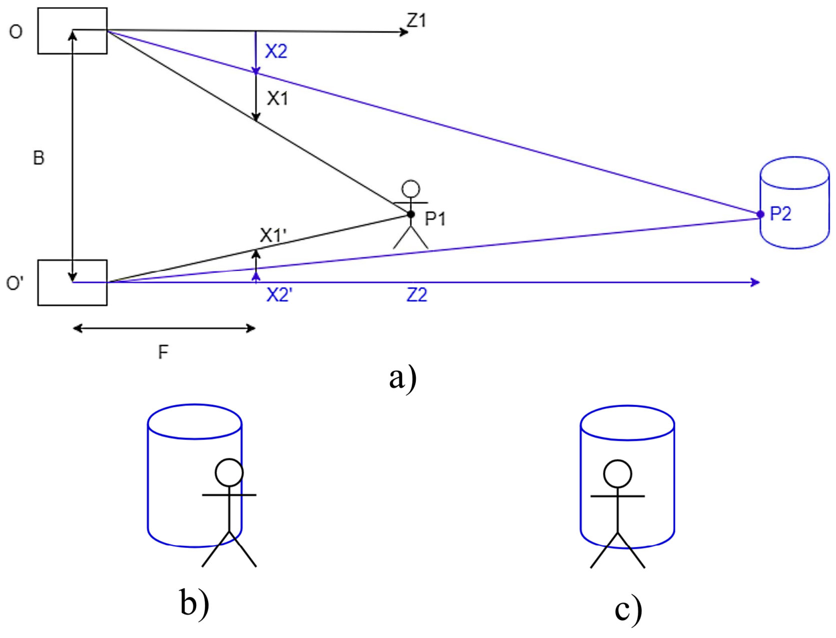


Figure 9. Disparity in a simple stereo system a), with b) and c) being the views from cameras O and O', respectively.

Of course, before the disparity can be calculated, first we have to know which part of the image is the same part in the other image. Finding out these common elements is a correspondence problem. To solve this problem, there exist two main classes of stereo matching algorithms, correlation-based and feature-based. In correlation-based algorithms the other image is checked for intensities that look like the intensities of the location in the first image, and the result is a dense disparity map. Schrastein and Szeliski reviewed dense stereo methods in the beginning of the century [27].

In feature-based methods the image is divided into features and the other image is checked for a subset of these features. Some used features are edge points, line segments and corners, which are first detected as interest points and then their neighborhoods are described by distinctive feature vectors. The result is a sparse disparity map which had to be interpolated to get a dense disparity map.

In 2004 the Scale Invariant Feature Transform (SIFT) [28] was introduced. It makes a large number of features which cover the image densely. These features are located by using a difference-of-Gaussian function. They are invariant to orientation, scale and location, and allow for distortion and change in illumination, to counter foreshortening and other perspective-related problems. One image is transformed into features and then the other image is matched with it by finding possible matching features and comparing their Euclidean distance.

SURF, Speeded-Up Robust Features, is a computationally more efficient method, as it uses box filters and integral images instead of difference-of-Gaussian [29]. It also uses Haar wavelet components to describe feature vectors. Later ORB [30] was introduced. ORB means Oriented FAST (Features from Accelerated and Segments Test) and Rotated BRIEF (Binary Robust Independent Elementary Feature). These are the detector and the descriptor used in ORB. The regular FAST and BRIEF are unstable when there is rotation, so orientation components were added to increase invariance.

Karami et al. compared these image matching techniques [31]. While ORB was the fastest algorithm by far (an order of magnitude faster than SIFT), SIFT performed best in most scenarios. SURF was 2-3 times more time-consuming than ORB but had better results outside heavily distorted or noisy images. Tareen and Saleem came to a similar result: SIFT was most accurate and ORB most efficient when seven different image matching techniques were compared [32].

Comparing stereo camera techniques to other methods described above, there are positives and negatives. In an uncontrolled environment, a passive system like a stereo camera works better than active light or infrared sensors, even though it has trouble with repeating textures or texture-less surfaces [16]. But compared to a mono view camera, a second camera takes up more physical space and consumes more power. Dual-pixel-autofocus cameras create a stereo view depth map with a single camera, and so don't have the physical negatives, but the sub-pixel precision and the ambiguity created by the autofocus create problems [14, 15]. The next chapters discuss depth maps computed with a stereo-camera process.

3. EMPHASIZING DEPTH PERCEPTION

Once the depth map has been acquired, it needs to be handled according to the need. These needs vary, and some of them are presented briefly in Section 3.4, but the use case focused in this thesis is to emphasize the blurriness of out-of-focus areas to create a shallower depth of field. This means controlling the bokeh of the image. However, the depth map has to be processed to be as clear as possible, so the manipulated end result image doesn't have irregularities in the objects' edges.

3.1. Bokeh

Bokeh is a Japanese word meaning the aesthetic of the out-of-focus blur. The term is used to describe the way the out-of-focus areas are handled in the image, by the lens or in post-processing. "Good" bokeh is very much in the eye of the beholder, as different kind of blurs work better depending on the purpose of the image, and that's why comparing the final images is a subjective matter [33]. Figure 10 shows four different approaches to bokeh.

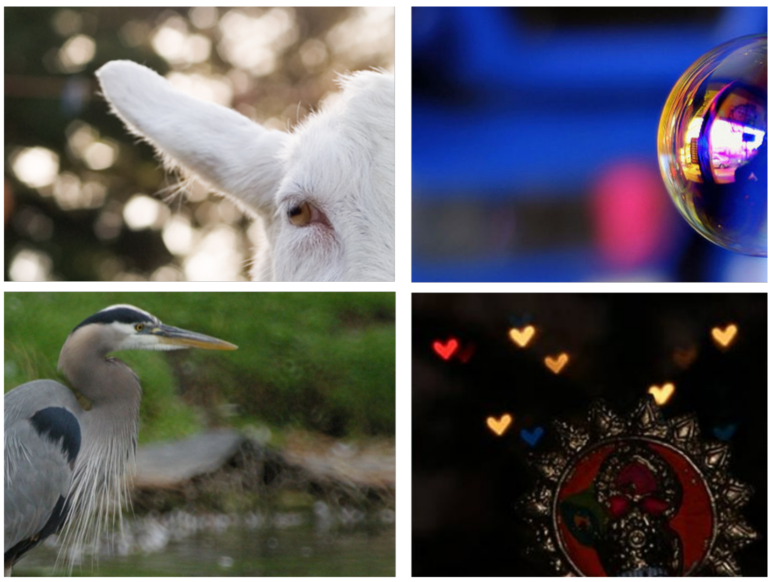


Figure 10. Some bokeh types seen in pictures at Wikimedia Commons, clockwise from upper left: 5-blade iris bokeh (from user Steve Ford Elliott[1]), smooth bokeh (from user brokenchopstick[2]), custom-designed heart-shaped bokeh (from user Arunus[3]) and a donut bokeh from catadioptric lens (from user Mark Wagner[4]).

[1]https://commons.wikimedia.org/wiki/File:5blade_bokeh.jpg, cropped from original, licensed under CC BY 2.0, accessed 7.10.2019.

[2]https://commons.wikimedia.org/wiki/File:Bubble_brokenchopstick.jpg, cropped from original, licensed under CC BY 2.0, accessed 7.10.2019.

[3]https://commons.wikimedia.org/wiki/File:Custom_Designed_Bokeh.jpg, cropped from original, licensed under CC BY-SA 4.0, accessed 7.10.2019.

[4]https://commons.wikimedia.org/wiki/File:Heron_and_donut_bokeh_20070607.jpg, cropped from original, licensed under CC BY-SA 2.5, accessed 7.10.2019.

The reason depth is emphasized in photos is to make them look more natural and focused, like a simulation of human vision. The human vision works by glancing around and changing the pupil's physical features to gather visual information for the brain, as only the center of the vision sees with full color and precision. The resulting formation of the vision does have similarity with a bokeh image, as peripheral vision creates circular highlights from out-of-focus light sources.

Bokeh doesn't have to be natural-looking. For example, when shooting with camera lenses, the highlight shapes in bokeh can also be something else than discs. The aperture blades covering the lenses while capturing the image can create polygon-shaped bokeh. Covering the lenses with a cover that has a distinctly-shaped hole makes the out-of-focus small light sources look like this shape. The size and noticeability of the bokeh shapes can also be inflated for artistic purposes [34].

3.2. Showing depth in a stereo image

Once the distances in the digital image are known, the pixels in it can be blurred by using a suitable filtering method. The simplest method is just taking an average from the neighbors and the pixel itself (box blur filter), weighing the amount of the blur by the distance from the focused distance. That way the depth of field has natural fading when moving farther or closer to the camera from the focal point. Blurring by the distance needs an accurate depth map.

Depth maps are one-channel images, or 2-D arrays, where pixels corresponding to the pixels in the original image tell the distance to the camera. In a stereo image's case, this depth map has been calculated from the disparities between the two images, and it corresponds with one of the images. With a depth map, the focused distance and the depth of field can be simulated to the image in post-processing. The depth maps can be visualized in grayscale, or in pseudocolor as in Figure 11. The original grayscale depth map showing the ground truth of the scene was from the dataset by Canessa et al. [2].

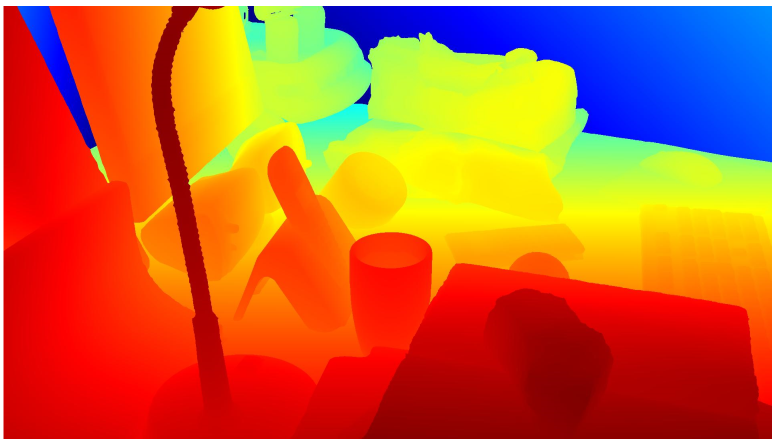


Figure 11. A pseudocolored depth map.

Bokeh effects, this meaning the enlarged points of light, can be rendered during this processing. A suitable kernel for the desired bokeh is chosen to go over these points of light. A smooth bokeh doesn't have clear circular or polygonal highlights and can be accomplished with a Gaussian filter [35]. Busy bokeh has large highlights, which in images taken by SLR system cameras may have (desired or undesired) ringing effects, making the center of the highlight darker than the edge. These can be simulated with the filter kernels. A naive approach with a 2-D-kernel can approximate different shapes, but its performance is poor. A separable approach by McIntosh et al. [36] uses multiple passes and rotating the image between the passes so the kernel used is 1-D. This can be used to approximate polygonal highlights.

3.3. Edge-preserving blurring

The borders between blurred and unblurred areas of the image are a problem in post-processing blur techniques. When the edges are not processed correctly, the sharp objects leak onto the blurry backgrounds. This results in halos around the sharp objects.

Bokeh effects added in post-processing can have artefacts which may not look exactly wrong but differ from the kind of bokeh originating from a SLR camera. One of these artefacts is sharp silhouetting. Instead of the out-of-focus foreground objects having partial occlusion, meaning blurry half-transparent edges when they are in front of a focused object, the edge is sharp because the sharp pixel doesn't sample from the foreground object. These depth discontinuity artefacts are an inverse problem from the sharp objects creating halos to blurry backgrounds [36]. The Figure 12 shows these artefacts.

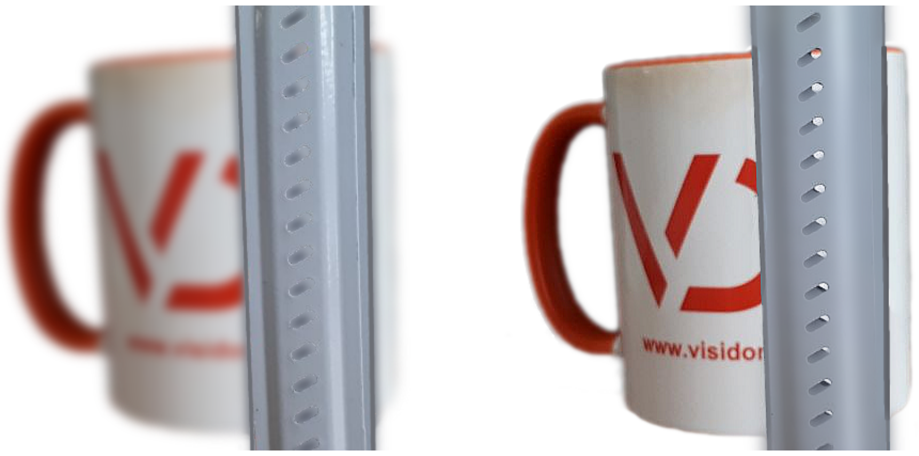


Figure 12. An illustration of edge-blurring artefacts, the left one shows halos as the sharp object leaks to the background, and the right one shows sharp silhouetting in a blurred foreground object.

There artefacts can be diminished with edge recognition. The depth map includes the information about the edges relevant to the blurring, and by identifying the disparity discontinuities these edges can be located. A mask can be created from the edges, and these masked areas of the image can then be handled with more care to get sharper edges without losing much efficiency.

The depth maps themselves have to be filtered with edge-awareness before they are used to guide the image's blurring. That part of the process should remove noise, smoothen the parts with the same depth and clear the edges in the depth map (like in Figure 2). The next section will discuss several edge-aware smoothing techniques suitable for this purpose.

3.4. Other applications for depth extracting

Depth maps and edge-aware filtering have other applications besides helping to blur images. Augmented reality (AR) is used more and more in mobile devices to modify the image the user sees. However, most smartphone apps using AR don't yet change the digital object in the environment according to its distance from the camera. In Pokémon Go and similar AR game apps the digital content is superimposed on the back-facing camera's view. The content is at a fixed length from the camera and doesn't take depth into account [37].

Gyroscope and accelerometer are used to show the Pokémon's position in the AR world, and it has a fixed 2-D position, but its size doesn't change when the user moves. This has a gameplay reason, as the player is often moving. For a more commonplace and ubiquitous AR a 3-D vision is necessary so the digital object can fit more naturally into the world seen through a mobile camera. This also allows real objects and humans to interact with the virtual objects better [38].

Snapchat and similar apps which can change the look of faces seen through the front camera, like beautification and face swapping, use facial recognition. Lately facial recognition methods used in smartphones have started to take depth into account. This makes the recognition more reliable and allows for authentication applications [39].

Haze and fog in images create a hindrance to the view that is dependent on the distance to the camera. Haze is caused by atmospheric particles, which absorb and scatter light. Fog is condensed humidity in the air. In both cases the direct transmission of light to the camera is disrupted, and so the image's contrast and the visibility of the far away objects is diminished [40]. The different methods to dehaze an image were surveyed by Li et al. [41]. The surveyed disparity map -based methods used stereo cameras and multiple images with different haze densities, but the single-image-based methods using priors gave better results. Fattal managed to dehaze single images without any additional information by using the surface shading in addition to the transmission function when modeling the haze[42]. A result of the method is seen in Figure 13[1].

Colorization is a computer-assisted process for adding color to grayscale images. This process takes some user input or outside information (what segments of the image should be assigned which color) and then colorizes the image. For the end result to look natural, image segments need to have precise edges [43]. This is an optimization problem similar to the depth map's edge-preserving smoothing. The fast bilateral solver, further discussed in Chapter 4, can be used to solve this optimization problem and colorize the grayscale image with some user input [44].

[1]Figure republished with permission of Association for Computing Machinery, from Single Image Dehazing by Fattal, Raanan, Proceedings, Association for Computing Machinery Volume 27 Issue 3, November 2008; permission conveyed through Copyright Clearance Center, Inc.

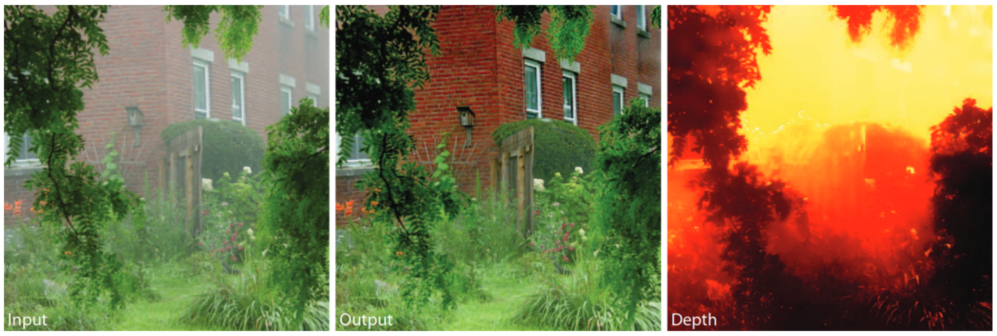


Figure 13. Dehazing based on a single input image and the corresponding depth estimate.

Object trajectory estimation computes the direction and the speed of the moving object in three-dimensional space. To achieve this, it needs to also see the object's distance in depth. When the camera is itself in a moving object, such as in a self-moving drone or robot, the path of the autonomous mover is computed based on the seen obstacles in the route. The depth extraction for this navigational computing can be done by using the methods described in this chapter (on their own or multimodally), active sensors [45], single image capturing with machine learning [46], or using stereo vision. A system introduced by [47] used pushbroom stereo vision to only search the disparities up to a certain distance, and was able to avoid some natural obstacles at speed up to 14 m/s without external sensors or computation, but had trouble avoiding tree canopies when there were many of them.

4. EDGE ADAPTIVE FILTERS

At this point in the process there exists an unfiltered depth map showing the disparities between the stereo images. The noise in this map has to be smoothed away before it's used to blur the image. However, the edges in the depth maps are the key features. They show the disparity dissonance between the objects in focus and not in focus. They need to stay sharp during the filtering to help the blurring. Sharpening an image is emphasizing the high frequencies of the image. Smoothing diminishes these same frequencies. Balancing the filtering to have both sharp edges and smooth surfaces is a very delicate act, and many techniques have been proposed to handle it.

Smoothing filters can be divided between linear and non-linear systems depending on whether their outputs are linear combinations of the inputs or not. Gaussian blur, a linear technique, blurs the image by a Gaussian function. Median filter, a non-linear technique, replaces each pixel with the median of neighboring entries. These techniques are relatively simple, but as simple methods they output inferior results compared to later methods. The Gaussian function blurs the edges and the median filter's edge-preservation is worse than that of linear filtering [48].

The discussed methods in this chapter are categorized into linear and nonlinear methods. At a high level, the filtering techniques can also be divided by whether they are local or global; do they compute a solution to a spatial piece of the image by sliding a window over it or finding a globally optimal solution to the image. Both linear and nonliteral categories have representatives of both local and global methods.

4.1. Linear methods

In linear filtering the output is a linear combination of the input. Linear systems are homogeneous and additive. Homogeneity means that an amplitude change k in the input $x(n)$ changes the output $y(n)$ correspondently ($kx(n) \Rightarrow ky(n)$). Additivity means that inputs can be filtered separately and the outputs added together in the end, or the inputs can be added together before the filtering. The result output is the same either way [49]. Linear filtering can be done using convolution, or by multiplication in the Fourier domain.

4.1.1. Guided filter

The basic idea of a guided filter is that it uses a second image, a guidance image, to influence the filtering. This guidance image can be the image itself, a different color channel of the image, a second capture with a flash on, or anything similar which has edges in the same places as the image to be filtered.

The guided filter output q of an input p at pixel index i, j is a weighted average, where W_{ij} is the filter kernel.

$$q_i = \sum_j W_{ij}(I)p_j \tag{3}$$

Figure 14 shows how the edge in the guidance image I is transferred on the output q. The edge not clearly visible in the input is imprinted from the guidance image. This structure transference imprints also other structural features of I, if it's different from the input. The guided filter has a local linear model between I and q, in that the output is a linear transform of I in a window ω_k centered at the pixel k, shown in equation 4. a_k and b_k are linear coefficients, constant in the window. These coefficients are determined by solving a minimizing problem of the difference between q and p.

$$q_i = a_k I_i + b_k, \forall i \in \omega_k \tag{4}$$

In middle of a high variance area (edge) the value of the pixel is unchanged, and in the middle of a flat patch the value changes to the average of the nearby pixels. This makes the filter edge-preserving.

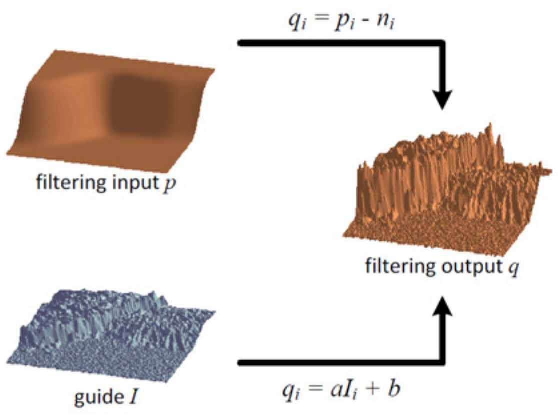


Figure 14. A guided filter uses a guidance image to influence the filtering of noise n. © 2012 IEEE

The original description of the guided filter, by He et al., was first introduced in 2010 [50]. In 2015, He and Sun presented a note for a faster performance [51]. By subsampling the smoothed maps and performing the box filtering part of the algorithm to these subsampled maps the complexity drops from $O(N)$ to $O(N/s^2)$, where s is the subsampling ratio. Since the guidance image, which has the edge information, is not subsampled, there's not a lot of visible degradation.

In a YUV color system, the guided filtering could be used with just the luminance channel Y, or using the chrominance channels U and V as the guidance images. In the case of depth maps, the luminance image is used as the guidance image to guide the depth map's filtering.

The guided filter was created as an alternative to the popular bilateral filter, which is described below. The original paper [50] compares the two and finds the guided filter faster and better at handling the edges.

The idea of using guidance images to help other filtering methods has been thought of before the actual guided filter. Two versions of a guided bilateral filter were introduced in 2004 [52, 53].

4.1.2. Fast Global Smoother

Guided filter, like the bilateral filters, is a local filter, and works by sliding a window over the image and handling a pixel at a time. Global filters try to find a globally optimal solution to the objective function, which works for all the pixels in the image. Fast Global Smoother (FGS) is an edge-preserving smoothing solver, which optimizes a cost function to calculate the smoothed result [54].

FGS is based on Weighted Least Squares, a model to estimate unknown parameters in a linear regression model. In data processing, Weighted Least Squares differ from Ordinary Least Squares in that it's the method needed when the data shows heteroscedasticity (the sub-populations have different variables), and the model has to take that into account.

Before FGS, WLS was used to reduce ringing while deblurring images [55], and more recently as an edge-preserving smoothing operator [56]. That latter article describes the smoothness weights depending on the input image's luminance channel and the designated sensitivity value to its gradients. The input is used to create a similarity function, represented by a sparse Laplacian matrix in a linear system. The filtered output is obtained by solving this large linear system.

FGS solves the system on a 2-D image by separating the algorithm into 1-D linear subsystems, as linear methods are additive. The Weighted Least Squares energy function for 1-D signal in dimension x is

$$\sum_x ((u_x^h - f_x^h)^2 + \lambda_t \sum_{i \epsilon N_h(x)} w_{x,i}(g^h)(u_x^h - u_i^h)^2), \tag{5}$$

where u^h is the output signal, f^h is the input signal, g^h the guiding signal, $N_h(x)$ a set of two neighbors for x ($x-1$ and $x+1$), and $w_{x,i}(g)$ is the weighting function. As the weighting function calculates the similarity between pixels, and takes the guiding signal as its definer, it works as an edge-aware smoother.

The linear system with an output u^h which minimizes the equation 5 is

$$(I_h + \lambda_t A_h)u_h = f_h, \tag{6}$$

where A_h is a Laplacian matrix, I_h an identity matrix and u_h and f_h are vector notations of u^h and f^h. Because the three-point Laplacian matrix is a $O(N)$-complex tridiagonal matrix solvable by Gaussian elimination, the linear system is $O(N)$ as well.

For a 2-D image, the global 1-D solver is applied as a separable smoothing algorithm. It goes through both dimensions, one dimension at a time, and in multiple iterations. Because the solver only takes one dimension's neighbors into account at a time, iterating two-dimensional images can create streaking artifacts to the result. To reduce these streaking artifacts, the amount of spatial smoothing is reduced between iterations by adjusting the smoothing parameter λ_t. A semi-global WLS method by Liu et al. [57] constructs the 2-D neighbors as a 1-D vector to remove this limitation.

4.2. Nonlinear methods

Nonlinear filters are defined as what they are not: They cannot be described as a linear relationship between an input and an output. They have a broader class of operations without explicit frequency domain transfer functions [58].

All the nonlinear methods described here are based on bilaterality. This means that when these methods search for the pixels close to each other, they take into account not just the spatial closeness but also the similarities of the pixels' values. On the feature edges these values differ a lot when going over the edge even though the pixels are spatially right next to each other. That kind of pixels are not deemed to be close for filtering purposes, and the edges stay unsmoothed [59].

4.2.1. Bilateral filter

Bilateral filter is a local method, counting the weighted mean of nearby pixels. It combines domain and range filtering to count both spatial and value closeness. Neither is enough on its own, as the domain filtering smooths edges and range filtering just remaps the color map. The combined filtering can be described with the equation

$$h(x) = k^{-1}(x) \int_{\infty}^{\infty} \int_{\infty}^{\infty} f(\xi)c(\xi, x)s(f(\xi), f(x))d\xi, \tag{7}$$

where $f(x)$ is the input image, $c(\xi, x)$ measures the geometric (domain) closeness, $s(f(\xi), f(x))$ the photometric (range) closeness and $k(x)$ is the normalization

$$k(x) = \int_{\infty}^{\infty} \int_{\infty}^{\infty} c(\xi, x)s(f(\xi), f(x))d\xi, \tag{8}$$

The closeness weighting is typically done with Gaussian kernels. When the filter's window is on a border between dark and light pixels, and the window's center pixel is on the dark side, the pixels from the light side are essentially ignored, and the dark pixels contribute according to their spatial distance [59].

The theoretical base behind the bilateral filter as a derivative of the Jacobi algorithm was later described by Elad [60]. One iteration of the Jacobi algorithm with a certain penalty function gives out the bilateral filter. Elad also suggests speeding up the filter by using Gauss-Seidel approach or iterating the gradient.

There have been many improvements proposed to the filter since its first description. Joint bilateral filter uses a version of the image with a better edge information for the range closeness calculation [52]. This usage of a guidance image is similar to the guided filter. Cross-bilateral filter works in a similar way and was proposed at the same time [53].

The practical solution evaluated in this thesis uses a guiding YUV image for the filtering of the depth maps. The depth maps often have holes, meaning swathes where the disparity wasn't computed right. These holes can be larger than just a couple of pixels, and so a bilateral filter that just looks for edges thinks that the edges of the holes are legit. A guidance image shows that there isn't supposed to be a hole there, and so it's smoothed over.

Using a bilateral filter may create artifacts to the output. Edges can have unwanted gradient reversal artifacts, creating halos around the edges. Another artifact is the staircase effect, resulting to flat cartoonish surfaces created to the homogeneous regions by the filter's weighted average. A 1-D signal example is shown in Figure 11. For each x, the filtered value is the average of the points y between $u(x) - h$ and $u(x) + h$, where h is a threshold. As this part of the signal is concave, in a) the number of points between $u(x) - h$ and $u(x)$ is larger than between $u(x)$ and $u(x) + h$, so the average becomes smaller than $u(x)$. At convex parts of the signal the average becomes larger, so in the parts where concave and convex parts meet, the result is a clear border between two comparatively flat areas [61].

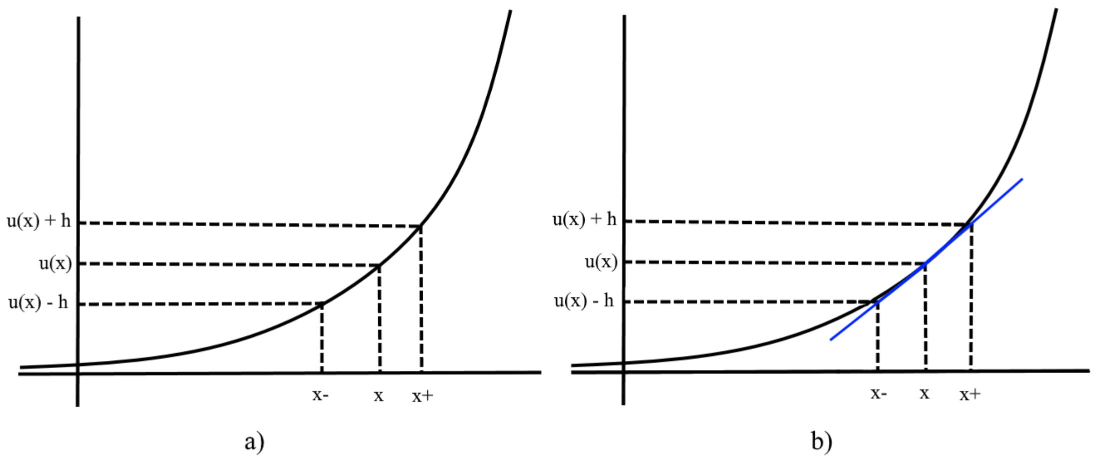


Figure 15. The staircase effect a), and the linear interpolation to correct it b).

The staircase effect can be countered with a regression correction, where a line is approximated as the signal's tangent. This linear regression gives a value that's closer to the signal at x [61].

4.2.2. Bilateral grid

Bilateral grid is a 3-D data structure representation for 2-D image data. The pixel's position in the grid is computed from its spatial placement in the image and its intensity. When the euclidean distance between two pixels in the grid is large, they are deemed too far from each other for smoothing purposes. So this presents the basic idea of the bilateral filter as a data structure. Using the grid helps to lower processing times, as e.g. bilateral filtering becomes a simple convolution in the bilateral space [62].

Figure 16[1] shows the grid's principle when created from a 1-D signal. The 2-D-grid is initialized to zero (blue cells) and the information (x, I), where x is the spatial (temporal) location and I the pixel's intensity, are inserted to the grid in a sampled position. The grid is sparse even with a large sampling rate. This grid is then processed

[1]Figure republished with permission of Association for Computing Machinery, from Real-time Edge-Aware Image Processing with the Bilateral Grid by Chen et al, ACM Transactions on Graphics Volume 26 Issue 3, July 2007; permission conveyed through Copyright Clearance Center, Inc.

by the chosen 2-D function. The filtering done to high-intensity area doesn't flood over to the low-intensity area and vice versa. After the processing the grid is transformed back in an operation symmetric to the grid creation. This process is called slicing.

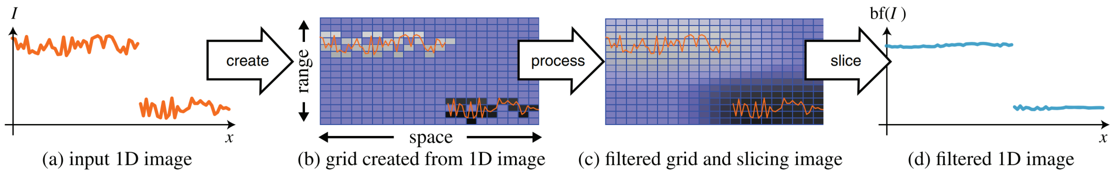


Figure 16. Bilateral filtering using a bilateral grid demonstrated on a 1-D example.

In a 2-D image case the insertion of the (x, y) pixels follows the equation

$$\Gamma([\frac{x}{s_s}], [\frac{y}{s_s}], [\frac{I(x,y)}{s_r}]) += (I(x,y), 1) \tag{9}$$

where s_s and s_r are subsampling rates for the grid's spatial and range dimensions, respectively. The result is a three-dimensional grid.

Using a three-channel color image creates a five-dimensional grid. This significantly raises the computing costs and the memory requirements, but the filtering is more precise. The grid is sampled to lower the number of cells and so the computing costs. Sampling the spatial axes affects the amount of smoothing, and sampling the intensity affects the amount of edge preservation.

Computing the edges with a guidance image is possible by decoupling the image data from the image edges. This results in two images, I and E. The data image defines the values and the edge image determines the positions on the grid. Dual-cross-bilateral grid [63] uses a modified grid for stereo-matching, by using both left and right input images as edge images, and the cost space as value source instead of the pixel intensity. It also tiles the 4-D bilateral grids into one large 2-D texture for quick computing.

4.2.3. *Fast bilateral solver*

Fast bilateral solver, FBS, solves a regularized least-squares optimization, and uses a simplified bilateral grid as its data structure. This method is global, as the optimization problem is solved simultaneously with respect to every pixel. In bilateral space, the local blurs between adjacent vertices correspond with non-local edge-aware blurs in pixel-space.

The solver uses a confidence image and a reference color image to guide the optimization. The optimization problem to be solved is

$$\underset{x}{\text{minimize}} \frac{\lambda}{2} \sum_{i,j} \hat{W}_{i,j}(x_i - x_j)^2 + \sum_i c_i(x_i - t_i)^2, \tag{10}$$

where x is the output, c the confidence and t the input. The bilateral affinity matrix W, which reflects the affinity (spatial and intensity closeness) between pixels i and j, is in a bistochastized form $\hat{W}$. This means that both rows and columns sum to the same constant and so the matrix is easier to manipulate.

The bilateral grid is constructed from the reference image. The affinity matrix is transformed to the bilateral space in a process called splatting. In this space, the optimization problem is now a sparse linear system

$$\underset{y}{\text{minimize}} \frac{1}{2} y^T A y - b^T y + c, \tag{11}$$
$$A = \lambda(D_m - D_n \bar{B} D_n) + diag(Sc), \quad b = S(c \circ t), \quad c = \frac{1}{2}(c \circ t)^T t,$$

which result y can then be sliced back to the pixel-space. S is the bilateral space transform (splat) matrix, B is the affinity matrix and D_m and D_n are bistochastization matrices.

FBS was introduced by Barron et al. [44], and it's based on a previous article about fast bilateral-space stereo [64] There bilateral filtering techniques are used to resample the optimization problem to a reduced bilateral space, and compute the optimization in that space. The method uses a simplified bilateral grid.

The simplified bilateral grid was compared to the usage of barycentric interpolation in high-dimensional filtering [65] and the result was that the grid is cheaper to construct and use. However, the grid's usage of hard assignments (the pixels are interpolated to vertices by using nearest neighbors) creates blocky artifacts to the filtered output. The optimization problem in the bilateral space stereo article is non-linear, but the one in FBS can be expressed in the bilateral space as a linear least-squares optimization problem.

An extension of FBS, a planar bilateral solver was introduced in a later article describing depth computing for smartphone AR [66]. The bilateral solver was used to post-process depth maps. Planar bilateral solver was implemented to remove the alleged problem FBS has, as it creates fronto-parallel surfaces of the depth maps. This means that the different depths are flat and face the camera like stage props in a theatre instead of being angled, like for example floors should be.

In the planar bilateral solver every pixel gets a three-dimensional linear system, where the solver is used to compute the matrix used in standard plane fitting. This creates a better recovering of angled surfaces but increases the computing.

Bilateral solver has also been used in video depth mapping for VR display [67]. Temporal dimension was added to the five dimensions FBS originally uses (two spatial dimensions and three for the color channel intensities) to optimize the solution over the entire video sequence.

4.3. Summary

There's an obvious incentive in showing that the proposed method is better than the methods currently in use, so articles usually include comparisons to other techniques.

As the bilateral filter and guided filter see frequent use in image filtering (both are e.g. included in MATLAB's Image Processing Toolbox [68]), they are often compared to proposed methods, and feature in many surveys. Bilateral filter works as a base model for edge preserving filtering, but it's seen as creating halo and staircase effects to the image. It also needs an efficient implementation to speed it up from the brute force's $O(Nr^2)$ complexity, where r is the kernel radius. Several implementations and extensions exist. A guided filter was devised to diminish the negatives of the bilateral filter. It's fast and doesn't have gradient reversal artifacts. However, halos are still possible, and as a local operator it may not recognize all the edges from noise. [69]

Fast Guided Smoother is based on the weighted least squares filter. WLS linear systems give high results, but solving an optimization problem can take a long time [70]. FGS works fast (its complexity is just $O(N)$) as a separable smoothing algorithm, but iterating a 1-D solver creates streaking artifacts, which have to be lessened with further iteration [54].

Bilateral grid creates a more high-dimensional grid of the image. In this bilateral space the filtering is linear. This makes for a much faster computing for bilateral filter and other filters, but higher dimensions (a color image becomes 5-D) eat more memory [71].

Because of its novelty, the bilateral solver has not yet featured in general comparison surveys or reviews. The papers it's featured in are limited to introducing the method and optimizing its use.

5. IMPLEMENTING EDGE AWARE FILTERS

The previous chapter handled the theory of these edge-aware filters. When implementing them, there are some platform-specific things to consider.

In mobile phones, the criteria with which to choose the most suitable method have a lot to do with the user experience. Generally speaking, if the image has even one big mistake, for example the foreground sharpness visibly overflows to the background, it doesn't matter if the rest of the image is perfect, it still looks bad.

Processing time works in a similar way. Getting the algorithm to work in real-time (enough frames per second that the human eye doesn't notice them changing) allows for its use in videos and in the preview image of the camera app. Too slow processing removes this possibility. This can be countered by leaving the time-consuming best filter to work behind the scenes when the actual photo is captured. Instead some faster filter is used in the preview.

In the worst cases, the filter may even slow down the image processing enough that using the filter with the best quality is not worth it at all. The user experience dampens too much. However, there are ways to optimize and speed up the processing by using parallel computing or trimming the unnecessary parts of the algorithm. If the filter's algorithm has parts that can be computed in parallel, it can be greatly sped up. In this chapter the algorithms are looked at from the hardware's point of view.

5.1. Optimization on mobile hardware

Phones are smaller than desktop computers, which shows directly in the hardware's architecture. The processing units are on a System-on-a-Chip (SoC), which integrates central and graphics processing units along with memory storage and other varied hardware.

Power consumption is another criterion on which the performance of a program is evaluated. It won't be discussed further here, but, as the phones are mobile devices, the maximization of battery life is an important consideration and the applications running on mobile phones may not get to use everything the hardware could offer when they run.

Central Processing Unit (CPU) performs the program's instructions. The CPUs are general-purpose, meant for performing a wide variety of applications. As the basic operation is subscalar (one instruction takes longer than a clock cycle), methods to parallelize its operations have been implemented. The current CPU's have several cores which can be used to divide the processing to different threads to speed it up. The CPU is optimized for handling tasks with a low latency and being able to quickly switch between operations [72].

5.1.1. GPU

Graphics Processing Units (GPU) are used in graphics-heavy applications, like image processing and rendering [73]. Compared to the CPU, the GPU's architecture can divide a task into far more threads. They contain a number of multi-processors, which contain processor cores and memory chips.

The GPU's have a lot of processing cores, a hundred times that of a multi-core CPU, and each core can run a separate thread. Loading data for processing can still be a bottleneck, as the individual memory chips are small, but the potential latency problems are suppressed when the GPU is processing enough computations at the same time.

For an algorithm to be well suited for a GPU optimization it has to be possible to parallelize it. Dividing an image to smaller blocks and performing each block in its own thread is an efficient way of using cores, but the requirement for that is that the data in the image is handled independently from other parts of the image [74]. For example, globally transforming separate blocks of an image to another domain, filtering them and then transforming them back easily creates discontinuities to blocks' borders.

5.1.2. SIMD

Single instruction, multiple data (SIMD) parallelizes the processing by performing the same instruction to multiple pieces of data simultaneously [75]. This is accomplished by handling the data in blocks, so a larger chunk of the data is loaded at once to fill the registry. For example, instead of a 32-bit processor performing same 8-bit commands one after another, it can perform four at the same time.

The SIMD optimization can be combined with multi-thread solutions. The SIMD instructions load larger chunks of data, but they are still single commands and a threaded routine performs them in separate threads for different calls of the routine. Inside one call the registry loads the same chunk of the data for the single instruction as it would without threading.

The instruction sets depend on the hardware used and include mostly just low-level commands. When using these commands, the developer has to take into account the register size and be especially careful with the variable types and sizes. This makes the code more complex for humans to read and write, but just by replacing the most iterated pieces of the code the processing can become several times faster.

Optimizing with SIMD is worth it when the algorithm has a heavily iterated routine which works with a large piece of data. This routine must be possible to re-articulate with low-level commands and handle data sequentially. As the image processing features similar calculations done to every pixel, SIMD can be used to work with several pixels at the same time.

There exist several architectures which extend the SIMD technology. ARM Neon is one such extension. It's used in several of ARM's processors. The Neon instructions can be replaced with Neon intrinsics in the code, which simplifies the writing from the assembly language [76].

5.2. Edge adaptive filters' optimization

Some of the filters described in the previous chapter can be optimized with relative ease, as they fulfil the requirements mentioned earlier. Others have more complicated computations, or the memory requirements are the more difficult part.

The guided filter's main computation is an evaluation of four box filters, all of them $O(N)$. Downsampling the image by a factor s lowers the complexity into $O(N/s^2)$, and also lowers the memory requirement. The box filters form the most computitave part of the algorithm. The computing time of a box filter depends of the box filters' kernel size. By lowering the kernel size, the time can be shortened, but it can lower the quality as well. Mezeni and Saranovac proposed that by adding a pre-filtering stage the quality can be improved without the complexity increasing. This pre-filtering stage applies a low-pass filter to the image before downsampling it, to prevent possible alialising [77]. Parallelizing box filters is easy, as the box filter calculates pixel locations one by one by using the nearby pixels and the kernel coefficients, and the result is saved to the output independent of the other pixels.

The Fast Global Smoother is a global filter, so it doesn't go through the image pixel by pixel. It calculates the globally optimal solution, which is done by computing a linear system. The 2-D image is filtered separately in both dimensions, first by rows and then by columns. Changing the dimension on which the image is traversed - and then changing it again for the next iteration - can increase the data handling time. When the image is traversed by rows, the pixel data is right next to one another, but when traversing by columns, they can be thousands of memory slots away and the memory access time increases. Transposing the image between operations just moves the computation there. This makes the SIMD optimization hard. Using multithreading works as the image can be divided into blocks by row or column, but this affects the global filter's result, as seen in Figure 17, which uses the stereo pair image Motorcycle, from the Middlebury 2014 dataset [85].

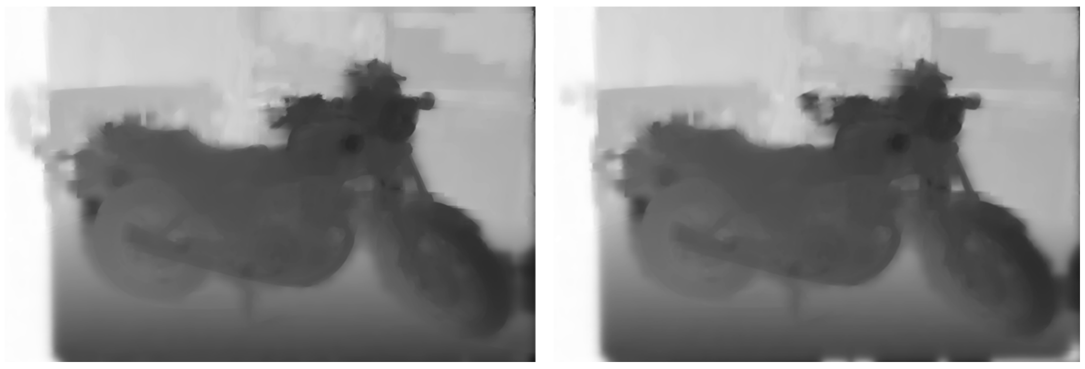


Figure 17. Fast Global Smoother filtering results when using one core (left) and four cores (right).

The basic bilateral filter uses Gaussian kernels to weight the closeness of the pixels' domain and range. A guided bilateral filter adds a guidance image, and it's used for range values' weighting. The complexity of this local filter is $O(Nr^2)$, where r is the kernel's radius. The filter goes through the image one pixel at a time, calculating the weights of the other pixels in the kernel. Lowering the kernel's radius is an obvious way to lower complexity, but that makes the edges harder to recognize and the filter more susceptible to the noise. Parallelization and SIMD are both possible.

The bilateral grid's complexity is $O(N)$, where N is the number of pixels in the image. Using just the luminance channel instead of all three color channels lowers the number of N. This affects the edges between two colors with the same luminance value. The memory is a larger concern, as with all the color channels the grid becomes 5-D. It however is very sparse and can be downsampled in the grid creation (too much and this affects the smoothing and the edge preservation). The grid is in bilateral space, and in that space the filtering can be done linearly, so it's possible to divide the filtering process into parallel blocks. SIMD is also a possible optimization method. Transforming the data into the bilateral space and slicing it back takes the majority of the filtering time. This part can also be parallelized.

The Fast Bilateral Solver's complexity lies in its data structure, and how it's used differently than in a simpler bilateral grid. The used 5-D bilateral grid is sparse, and its structure varies as a function of the input. This means that the memory access is unpredictable. The solver solves a minimization problem, which is a sparse linear system. These make the solver very hard to parallelize. Table 1 summarizes the implementabilities of these methods.

Table 1. Simple comparison of the implementability of the filtering methods

Method	Complexity	GPU	SIMD
Guided filter	$O(N/s^2)$	+	+
Fast Global Smoother	$O(N)$	+	-
Bilateral filter	$O(Nr^2)$	+	+
Bilateral grid	$O(N)$	+	+
Fast Bilateral Solver	$O(N)$	-	-

5.3. Fast Bilateral Solver's implementation and optimization

The Fast Bilateral Solver's algorithm can be divided into initialization and filtering phases. The initialization phase of the Fast Bilateral Solver constructs the splat and slice matrices, which are used to transform the images to and from the bilateral domain. They depend on the image and so can't be initialized in advance. This is the most time-consuming part of the initialization, but comparing it to the actual filtering, neither the matrix construction nor the bistochastization of the matrices is a bottleneck. Solving the global minimization problem in the filtering phase is. Figure 18 shows the pipeline of the FBS algorithm.

Barron et al. [44], who introduced the FBS, have provided a barebones algorithm of the method. The following goes through that implementation and looks for possible improvements. Mazumbar et al. tested a modified version of the FBS, which they proclaimed as more parallelizable [78].

The data structure used for the bilateral domain is a bilateral grid, and it's created in the initialization phase. The implementation takes as an input the depth map, and

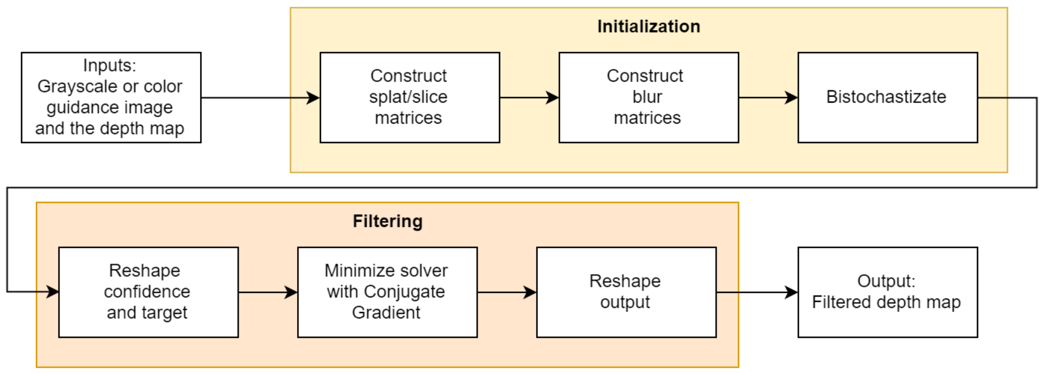

Initialization
Inputs: Grayscale or color guidance image and the depth map
Construct splat/slice matrices
Construct blur matrices
Bistochastizate
Filtering
Reshape confidence and target
Minimize solver with Conjugate Gradient
Reshape output
Output: Filtered depth map

Figure 18. Fast Bilateral Solver algorithm pipeline.

the original image as a guidance image. The depth map is used to create a confidence image, which represents how certain the values in the map are. In the filtering phase more confident values weight more.

The bilateral grid is formed from the guidance image by hashing the coordinates into unique values and using them to construct the splat matrix. An image multiplied with a splat matrix is now in the bilateral space, its pixels in bins. The splat matrix is very sparse. Its construction can be parallelized by dividing the image into blocks, but the hashed coordinates have to be combined after the parallel computing, which diminishes the optimization somewhat.

Subsampling the pixel's spatial locations or values lowers the memory requirement, but this affects the smoothing and edge recognition. Not using all the image's channels is a simple speeding up act, and so Mazumbar et al. dropped the color channels and used only a 3-D bilateral grid. This visibly downgrades the result, as the color channels show edges that can't be located just from the luminosity channel.

The blur matrices are created similarly. They represent the blurring kernel in the bilateral space, and so for every dimension, the hashed coordinates are iterated to look for the neighbors of every hashed coordinate. This is working with sparse matrices, and so not easily parallelized.

The bilateral affinity matrix is bistochastized. This operation iterates over the matrix a handful of times, but the operations are relatively simple and so nowhere close to a bottleneck compared to the solving itself. Still, an approximately correct bistochastization may be enough, in which case the computing can be made simpler and with less iterations.

After all this initialization the confidence and target images are reshaped into the bilateral space, where the actual solving occurs. The target image is the depth map to be filtered.

The minimization problem is a sparse linear system, as described in the Equation 11. Solving it is the most time-consuming part of the FBS. The solving is done by a conjugate gradient method, with preconditioning by a Jacobi preconditioner. The minimization is iterated several times to get an accurate enough result. The number of these iterations is the simplest way to change the time taken to solve the minimization problem. Iterating needed until the result is good enough varies by picture, so optimizing the number depends on the image.

Parallelization of the minimization is hard, because the steps in the conjugate gradient are global. Mazumbar et al. performed the optimization with local gradient descent with momentum, which increased the number of operations but made them easier to parallelize. After the filtering is done in the bilateral space, it is sliced back with a transposed splat matrix, with the output being the filtered depth map.

In the evaluation detailed in the next chapter, the used implementation of the FBS was based on the provided algorithm, with some further development.

6. EVALUATION

The filters' qualities and speed are evaluated in this chapter. The Table 2 lists the pros and cons for the five methods according to the literature in brief.

Table 2. Comparison of the filtering methods

Method	Pros	Cons	Other
Guided filter	Fast ($O(N/s^2)$), no gradient reversals	Halos	s is the subsampling rate
Fast Global Smoother	Fast ($O(N)$), no halos	Streaking artifacts	Based on WLS
Bilateral filter	Preserves edges	Halos, staircase effect, processing time ($O(Nr^2)$)	Can use a guidance image, r is the kernel radius
Bilateral grid	Faster implementation of the bilateral filter	Memory requirement	High-dimensional bilateral space
Fast Bilateral Solver	Accurate results	Memory requirement, hard to parallelize	Uses bilateral grid

Evaluating the filters' results needs a reliable ground truth. These ground truths are depth maps showing the exact scaled depth values for every pixel. The filter results' qualities can be compared to them. But while the result image is an important concern, the running speed of the program is an equally important measurement. An algorithm which creates a perfect depth map still can't be used in a phone if it takes too long to process the image in practice and there's no suitable way to optimize its implementation.

The evaluation algorithm outputs a depth map after the disparity computing, and again after the edge-aware filtering. The differences in these two depth maps, the filters' input and output, are compared to the ground truth to see how much the quality of the depth map was improved by the filter. The errors used in the metrics calculations are the differences between the ground truth and the test outputs.

While metrics show objective results, comparing two filter results in images is still a subjective ordeal in practice. And while some errors like halos and wrongly filtered parts can be seen clearly as errors by human eyes, many rankings of the differences between the image filtering results are in the eye of the beholder. Some glaring error cases are shown in the end of this chapter.

6.1. The test image set

There exist several stereo vision benchmark web sites for comparing depth map results. These web sites include stereo images with ground truths, usually giving out training sets and hiding test set ground truths for benchmarking purposes. The Middlebury

Stereo Vision Page [79] collects the benchmark results of many stereo correspondence algorithms. It was created to have a test bed for the quantitative evaluation of these methods [27]. The currently available evaluation results there include the Fast Bilateral Solver of the main methods described in this thesis.

KITTI Vision Benchmark Suite [80] has stereo image data gathered from vehicles moving outdoors in the traffic. The ETH3D benchmark [81] was created to have a diverse dataset, with both indoor and outdoor scenes with real-life challenges. It has much larger images (24 Mpx) than other benchmarks (e.g. Middlebury's images are 6 Mpx), and uses the same metrics as the Middlebury benchmark. There exist also computer-generated scenes, which have the advantage of precise measurements of depth. One of the computer-generated stereo vision test benches is MPI-Sintel Stereo Dataset [82], which was generated by using an animated open source movie's 3-D graphics source files to render two images from slightly different perspectives.

Asserting the ground truth to a real-life image, with pixel-level accuracy, requires that these distances have been measured precisely by some method, like using structured light. Even still the occlusion holes are a possibility. In one image the foreground object blocks the background from view and so the same part of the background can't be found from both images for the disparity calculation. These are usually masked as zero to the depth maps and look like shadows on the borders of the foreground objects.

When choosing the test sets, the necessary requirements of the images were that they have a stereo pair view of the pictured scene, and a ground truth depth map of either view. Because of this necessity, some datasets would have worked poorly in these tests. For example, while the ETH3D benchmark has large images from different real-world scenes, the multiview images don't have image pairs comparable to a phone stereo view, which is the use case. The viewing locations and directions are too far apart from each other. Middlebury Stereo Evaluation dataset was chosen because its images resemble the use case. It's also prevalent in similar studies.

6.1.1. Middlebury Stereo Evaluation datasets

The first Middlebury stereo set was created by using piecewise planar scenes (2-D pieces of posters or paintings set up in front of the camera) and labeling by hand. The depth maps were sub-pixel precise, but the scenes weren't complex enough to represent the real world [27].

Middlebury developed the capturing process and the next datasets were captured by using structured light to acquire both precise and complex depth maps [17, 83]. The scenes were illuminated by using binary Gray codes and 10 vertical and 10 horizontal patterns to uniquely encode each pixel. The total number of unique illuminated light patterns was 80. Gray codes have since been used in all the later Middlebury stereo evaluation datasets.

A more automated version of this method was used in 2005-2006 to gather 30 more stereo datasets. These datasets have seven views from different directions per scene, and two of the views can be used for stereo view testing [84].

The currently latest dataset, with 33 stereo view scenes, is from 2014 [85]. In its capturing process, instead of the standard Gray codes used previously, maximum min-

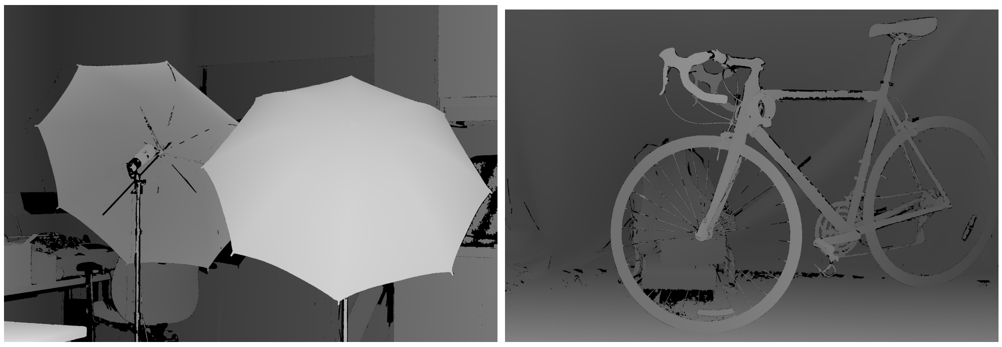


Figure 19. Two ground truth images, Umbrella and Bicycle1.

stripe-width Gray codes were used to better recover reflective surfaces. The number of different exposures was quadrupled.

The MiddEval set used in these tests was this 2014 set. Middlebury also provides an evaluation set meant for their stereo benchmark. This set includes 15 stereo pairs. Some of the pairs have images from different illumination conditions, which are irrelevant in this case. For example, one stereo image pair (PianoL) had one of the images shot with a much lower exposure. The used disparity calculator isn't suitable for use cases like this and so the initial depth map to be filtered was full of noise and nothing else. As this is outside the filters' use case, the evaluation set images were not used.

The stereo view images in the set are in .png-format, and their resolution varies, being generally about 6 Mpx. The ground truth depth maps are in the .pfm file format, where the data is in a 32-bit float format. The maps were converted into signed integers for comparing them with the filter results. This conversion approximated the sub-pixel values of the ground truth, but as the implementation approximates the outputs into integers the tests have to be done on integer level anyway. Figure 19 presents two of the 23 used ground truth depth maps.

6.2. Metrics

The metrics were chosen to highlight the differences between the methods, taking into account the subjectivity of the final results. The chosen perspectives were the average pixel value difference from the ground truth (which is here represented by the root mean square error), the severity of the highest differences (represented by 90-Percentile) and the visual similarity to the ground truth (represented by Visual Information Fidelity).

Besides these qualitative metrics, the time taken per megapixel was also calculated for every image. This measured time includes only the filtering time, as the algorithms are otherwise identical.

6.2.1. Root mean square error

The root mean square error (RMSE) shows the square root of the squared differences' mean, and is calculated as

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2}, \tag{12}$$

where e is the error in pixel i, and n is the total number of pixels. RMSE measures how much the results differ from the prediction, in this case the ground truths.

There has been a debate on whether RMSE or mean absolute error (MAE) is more accurate when calculating errors [86]. As RMSE squares the errors, it penalizes large errors more than MAE. As the errors in object's edges are large if there is an overflow or the edge is otherwise in a wrong place, RMSE suits these tests' purpose better.

6.2.2. Error quantile in pixels

Quantiles divide the samples into sets of equal size, after putting them into an order of ascending value. The value in a cut-off point is the measurement. The 2-quantile is the median, and 100-quantiles are called percentiles. 50-percentile means the same as the median.

90-percentile was used in these tests. The absolute differences between the ground truth and the filtering result were calculated for every pixel, and the error quantile was the value of the error of which 90 percent of the errors were below it. This metric gives information about the highest errors in the image while still excluding the largest anomalies, especially when comparing the 90-percentiles between the unfiltered and filtered depth map errors.

6.2.3. Visual Information Fidelity

Visual Information Fidelity (VIF) is an information-theoretic framework for comparing visual data. It measures the loss of image information, by quantifying the information that can be extracted from the reference image (ground truth depth map in this case) and then looking at how much of that information can be extracted from the image to be evaluated [87]. The value of VIF is between 0 and 1. 0 means that no similar information could be extracted, and 1 that everything from reference image was found in the evaluated image.

As VIF is based on modeling the human visual system, it was added to the metrics to get results which could map better to the subjective human vision. The creators of the metric evaluated it with other image quality assessment algorithms and deemed it the best [88].

6.3. The test setup

The filtering methods presented earlier were incorporated into a stereo image depth blurring algorithm. The usage of the filters in the algorithm is identical between them. They all receive the same input depth map and filter it, with the result being used to blur the out-of-focus areas of the final image. The algorithm was run on 8-Core AMD Ryzen 7 2700X.

The stereo images in the Middlebury dataset were in the png-format. The algorithm needed the images in a YUV-format, so they were first preprocessed into NV21. In this format the image planes are 8-bit, and the U/V-chroma planes are subsampled into a quarter of the image size and interleaved after the Y-luminosity plane.

These preprocessed NV21-images were the input to the stereo depth algorithm. There they were rectified, and their disparities computed using SURF and confidence based fill filtering. This computation outputs a depth map of the image (left view if not otherwise specified), and this depth map goes to the chosen filter for processing. Before that, it's saved into a file for evaluation.

One of the filter implementations is chosen for the depth map's filtering, and at this point the result becomes different based on the filter. The filters used in the tests were one implementation each of the methods described in Chapter 4. They have been developed and configured for a stereo view use case and have had some optimization for maximum quality per time used.

The filtered depth map is also output into a file. After this the variable type of the depth maps is changed from signed 16 bits to unsigned 8 bits. The depth map is also converted to a pseudometric scale using look-up tables. On this new scale the closer objects in the image are now lighter, when they were darker in the filtering phase.

The converted depth map is now used to create the bokeh blendmask. The bokeh mask is upsampled back to the image size. The original image (the one which the depth map depicts) is blurred with the bokeh mask's information, and this blurred image is the algorithm's output.

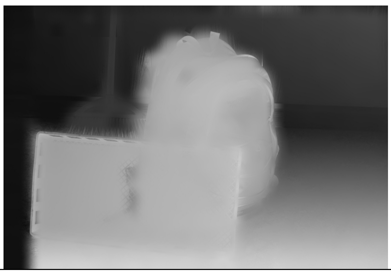
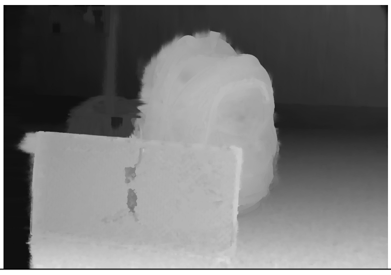
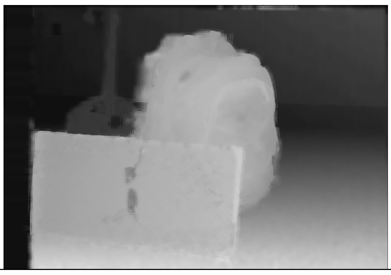
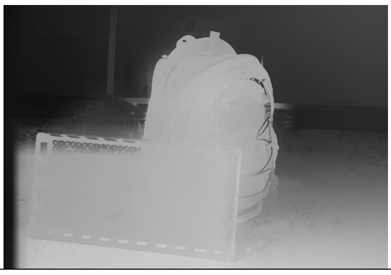
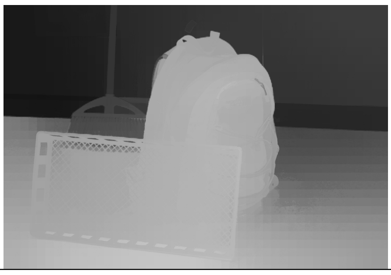
6.4. Results

The comparisons between the depth maps before and after filtering, against the ground truth, were done in MATLAB. The resulting depth maps were on a different disparity scale from the ground truth depth maps, so the input and output depth maps have had their histograms adjusted to better match to the ground truth's scale. As the evaluation is on the difference between the input and the output depth maps, this doesn't distort the comparison of the results.

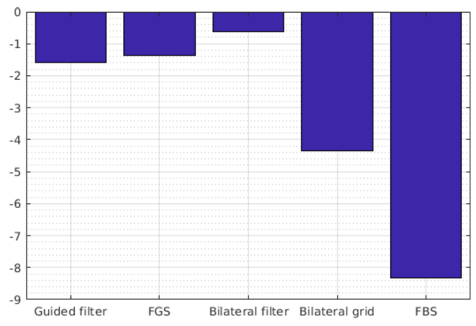
Table 3 shows an example of the filtered depth maps. In RMSE and Error Percentile changes, the lower number is better (as that means that the errors go down), and in the VIF change the higher number is better (as that means that more of the ground truth's information could be extracted from the result). The less time needed for filtering, the better.

The table shows that the numerically highest improvements were achieved with the FBS. The metrics fit with a visual inspection of the depth map, which has good-looking edges. The grid in the box has also been recognized, which the others weren't able to do. The visible blocks in the FBS-filtered depth map are because of parallelization. In addition to the good results, the time it takes for the FBS to filter the image is an order of magnitude larger than the next time-consuming method. These observations fit also on the average results of the test set, as seen in Figure 20.

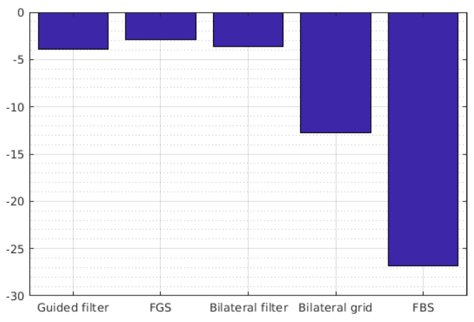
Table 3. Example results with the stereo image Backpack from the Middlebury 2014 dataset.

Method	Output	RMSE change	Error Per-centile change	VIF change	Time (ms/ Mpx)
Guided filter		-1.50	-6	-6.9x10-3	2.72
Fast Global Smoother		-1.26	-5	-5.2x10-3	8.59
Bilateral filter		-1.33	-5	-9.4x10-3	20.44
Bilateral grid		-5.35	-14	-7.9x10-3	12.8
Fast bilateral solver		-7.08	-11	18.7x10-3	183

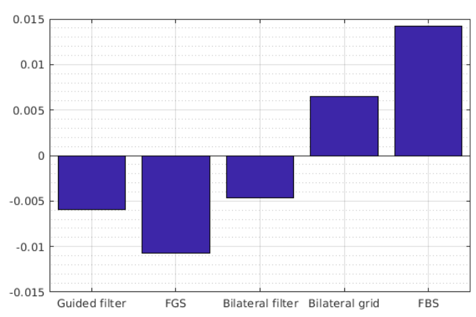
The black lines on the left side of the images come from the impossibility of calculating disparities between parts which can only be located in one image. The global filters can better extrapolate this area.



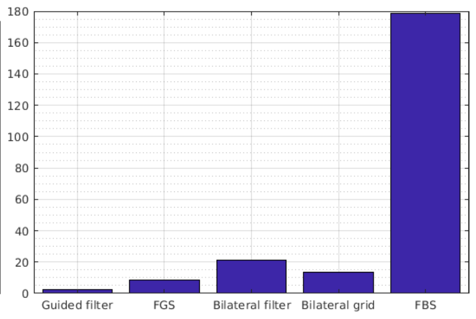
(a) RMSE



(b) 90 Error Percentile



(c) VIF

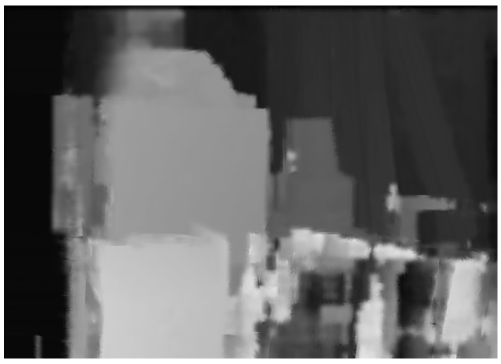


(d) Time per megapixel

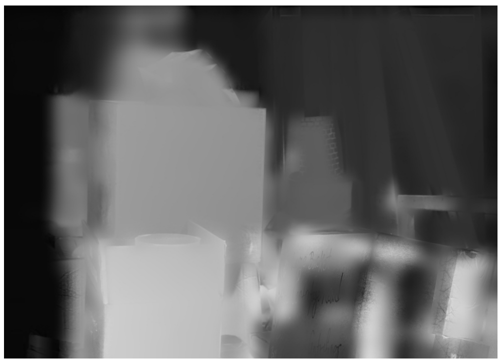
Figure 20. Average results from filtering the dataset with the five methods.

Modifying the parameters of these methods varies their output. This has some effect to the metric results, but generally they stay at the same level of magnitude. For example, changing the radius of the Guided filter's kernel affects the result, as seen in Figure 21, which uses the image Storage from the Middlebury dataset 2014 [85]. While the edges look sharper where they are in the picture, since they have been taken from the guidance image (the letters in the foreground box can be seen in the right image), they are surrounded by a glow.

Comparison of the metrics with these two radii is in Table 4. It shows that the glowier depth map gives out better results with the metrics (VIF is slightly worse in this picture, but on average over the dataset it's better). Figure 22 shows a detail in

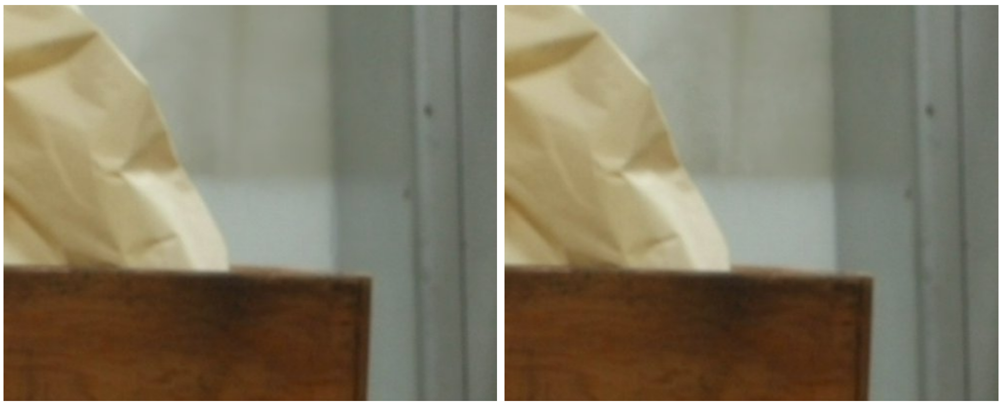


(a) Kernel radius 3



(b) Kernel radius 10

Figure 21. The difference in the guided filter's results when the depth map is filtered with different kernel radii.



(a) Kernel radius 3 (b) Kernel radius 10

Figure 22. A detail of the algorithm's output image of the image Storage, when the depth map has been filtered with the guided filter.

the final image where these depth maps were used in bokeh masking. The visual differences are negligible, showing a visible difference only in the background line above the box's corner, where the disparity calculation had given a wrong result in the lighter part of the wall, and the smaller kernel radius hasn't done much to it. The larger kernel radius has fuzzed the edge where it doesn't belong.

The results in the Figure 20 are taken with the kernel radius 10. The results are better, but they are still in the same class as FGS and the bilateral filter. The similar effect can be seen when adjusting other methods' parameters; unless they are adjusted completely wrong, the results are in the same ballbark as with other parameters. All the methods in this evaluation have been adjusted to show the best results on these images.

Table 4. Comparison of the radii in the guided filter's filtering of the image Storage from the data set.

Radius	Storage RMSE	Average RMSE	Storage 90-Percentile	Average 90-Percentile	Storage VIF	Average VIF	Storage time/ Mpx (ms)	Average time/ Mpx (ms)
3	-1.40	-0.73	-7	-2.74	-14x10-3	-7.9x10-3	2.57	2.60
10	-3.34	-1.57	-12	-3.9	-17x10-3	-6.0x10-3	2.56	2.65

The depth map of the image needs a lot of filtering, as the original depth map of the Figure 23 shows. The thin branches have become wide blocks. The Fast Global Smoother filters the edges, but while it manages to find them, the blocks leave glowing patches around the branches. A similar effect can be found in the guided filter's output, while a bilateral filter's output has halos. The output of the bilateral grid still has some glowing patches, but the branches are much more distinct from them. Still the branches have discontinuities. The Fast Bilateral Solver doesn't have either of these problems, the branches are continuous and their edges are clear.

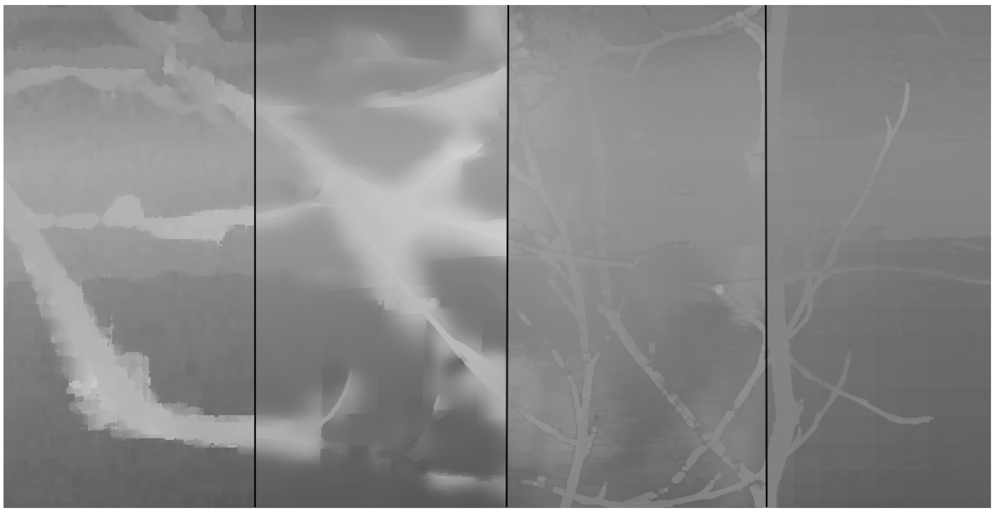


Figure 23. Details of the created depth maps of the stereo image Sticks, from left to right: The unfiltered depth map and filtered with the Fast Global Smoother, the bilateral grid and the Fast Bilateral Solver.

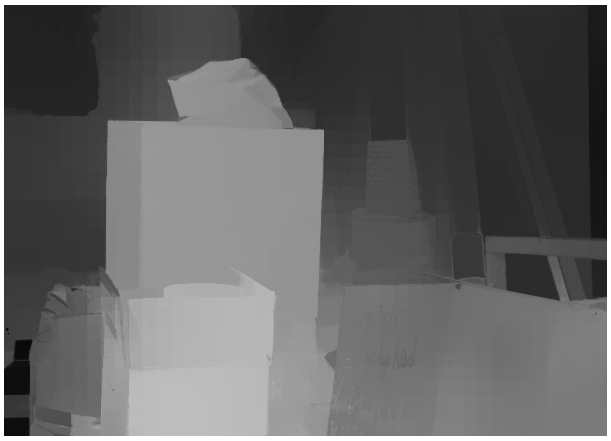
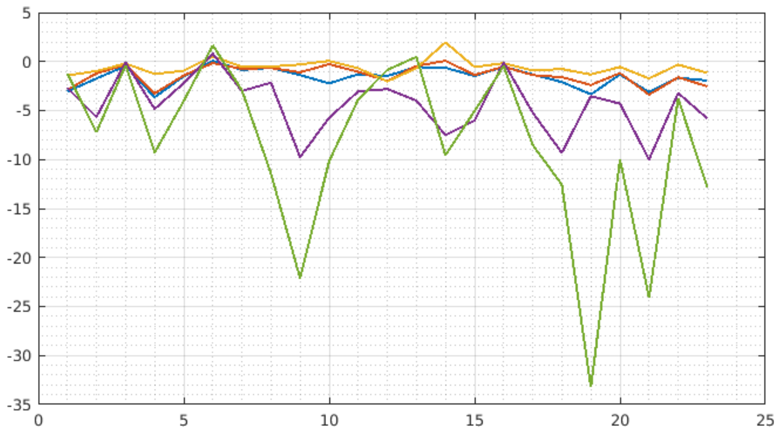
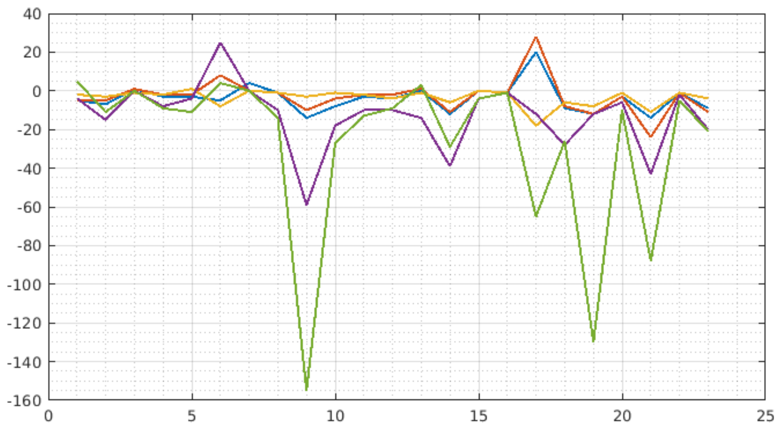


Figure 24. The depth map of the image Storage after the Fast Bilateral Solver.

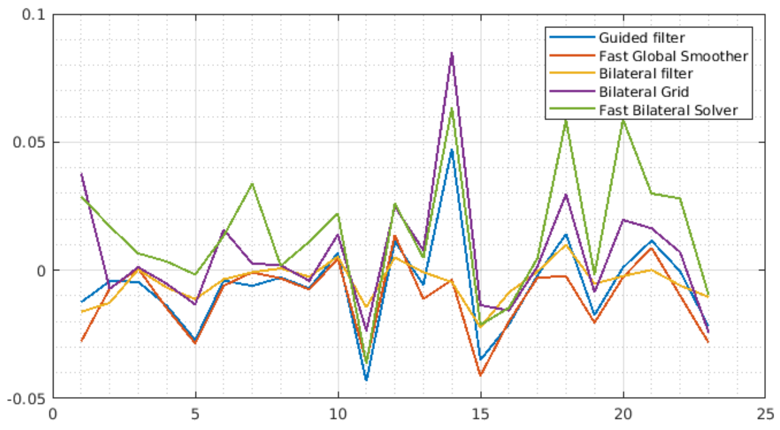
Figure 25 compares the method results for every image in the dataset separately. It can be seen that the FBS and the bilateral grid give almost consistently the best results. The stereo image Sticks, featured in the Figure 23, is the number 18. The results show that the FBS and the bilateral grid are the only methods which managed to significantly improve the result. Generally the images where the FBS shines the most compared to other methods are ones like these, which have thin objects blown up by the disparity calculation or an otherwise noisy depth map. An example is the image 19, Storage, where the box on the foreground (lower right) has gone noisy. The guided filter is unable to fix it, as seen in the Figure 21, but the FBS manages to smooth it, as seen in Figure 24. However, the noise makes it so that the box is thought to be much farther away than it is.



(a) RMSE



(b) 90-Percentile



(c) VIF

Figure 25. The difference in the metric results after filtering the computed depth maps.

7. DISCUSSION

The tests focused on the filters' abilities to create as clear depth maps as possible from flawed disparity images. The desired result was a depth map with visually pleasing sharp edges and smooth surfaces acquired in a reasonable time. The depth maps help to defocus the parts of the image with less important information. This is used to create images that better simulate the human visual system and images captured by single-lens reflex cameras.

The focus of the thesis has been on the Fast Bilateral Solver, because of its novelty and the reported result quality. The FBS's original paper [44] promises a lot, starting with the word “fast” in its name. The bilateral solver has been compared to a variety of stereo algorithms with the Middlebury Stereo Dataset, and shown to achieve superior outputs. However, the processing time was not reported, as "the runtime of [FBS] and all baselines is dominated by the time taken by the MC-CNN technique common among all entries".

As the reported numerical results with the FBS were good, the processing time was the biggest question in this thesis. The results submitted to Middlebury Stereo Evaluation web page have 140 milliseconds per megapixel as the average processing time. However, that includes the disparity calculator MC-CNN (Stereo Matching Cost with a Convolutional Neural Network), and so is not comparable to the current tests.

Instead, two of the result's qualitative metrics are comparable: RMSE and the 90-Percentile metrics. The results in the current test environment were worse than the ones reported to Middlebury. One of the likely culprits is the format of the ground truth depth maps. They may not scale perfectly to the data coming out of the used disparity calculation, which may also not be as accurate as with MC-CNN.

Based on performance metrics the filtering methods can be divided into two categories. The guided filter, the bilateral filter and the Fast Global Smoother improve the depth map's RMSE and 90-Percentile metrics, whereas the VIF gets worse. The bilateral grid and the Fast Bilateral Solver rank significantly higher than the others with this dataset. The FBS turned out to be the slowest algorithm.

The RMSE and the 90-Percentile results correlate in the results. The highest error values in the image also affect the squared error's value. The results of the VIF appear inconsistent. The three methods with the lower quality have their results in the same ballpark.

The clear outcome of the tests is that the FBS's results are good, but the processing time is a challenge. With the current implementation it takes nearly 200 milliseconds per megapixel to filter a depth map. Cameras on phones typically output Full HD -images, which have about 2 megapixels. Larger images are also becoming more common, especially in the 4K future. Currently the FBS is too slow to work in real-time with those. Single shots without as strict time requirements are possible, depending on what other processing is done to the image.

Of the other four tested methods the bilateral grid works well when its small processing time is taken into account. As its average execution time was less than 14 milliseconds per megapixel it could be used in the place of the other fast methods.

In the future optimizations of the FBS need to be explored. Currently the image was downsampled into a quarter resolution. More downsampling may be possible without losing too much of the solver's advantage. Dropping the color channels or implementing a more parallelizable substitute for the conjugate gradient solver are other possible paths. The optimization is a balancing act between minimizing the time needed and keeping the quality as high as possible.

8. CONCLUSION

The lenses of the mobile cameras can't create a shallow depth of field that can be compared to one created by a single-lens reflex camera. The simulation is achieved by blurring the parts of the image which would be out-of-focus in a shallow image. This needs an accurate and edge-aware depth map of the image.

The current state-of-the-art has several approaches for acquiring the depth map. A single-lens system needs either supplementary information about the captured frame (like structured light), multiple frames taken with the same lens, or artificial intelligence to segment the image's content. A stereo lens system which captures two frames from slightly different points of view has the advantage of working well in an uncontrolled environment.

The stereo frames can be used to calculate the depth map of the image by disparity calculations. Because of imperfections in the disparity data the depth map is noisy. This results in artefacts when the depth map is used to defocus parts of the image.

This requires filtering the depth map. Depth aware smoothing filters can be divided into global and local, or linear and nonlinear filters. The investigated filters, the guided filter, the Fast Global Smoother, the bilateral filter, the bilateral grid and the Fast Bilateral Solver, were representatives of all of these. Some of the filters were chosen because of their prevalence in image filtering, and some because of their reported quality. The relatively new but promising method FBS was the most interesting one.

The implementation of the filters on mobile phone architectures can be made faster by using the CPU and the GPU for parallel computing. The local filters, the guided filter, the bilateral filter and the bilateral grid are easily parallelized. As a global filter, the processing time of the FBS is high, even in the bilateral space.

The results showed that the FBS filters the depth maps really well, better than the other evaluated methods, but is also slow. The bilateral grid improved the depth maps almost as well as the FBS but needed only a fraction of the processing time. The input to the filters matters also a lot, as the noisier images improved the most when filtered with the FBS.

Without further optimization the FBS could be used for single-shot depth map filtering, depending on the specification, but the real-time use is far away. Downsampling, removing color channels or modifying the solver are some optimization angles.

9. REFERENCES

[1] Pentland A (1987) A new sense for depth of field. IEEE transactions on pattern analysis and machine intelligence PAMI-9(4): 523–531. DOI: https://dx.doi.org/10.1109/TPAMI.1987.4767940.

[2] Canessa A, Gibaldi A, Chessa M, Fato M, Solari F & Sabatini S (2017) A dataset of stereoscopic images and ground-truth disparity mimicking human fixations in peripersonal space. Scientific data 4(170034). DOI: https://dx.doi.org/10.1038/sdata.2017.34.

[3] Allen E & Triantaphillidou S (2011) The Manual of photography. Focal Press, tenth edition.

[4] Gunturk B, Glotzbach J, Altunbasak Y, Schafer R & Mersereau R (2005) Demosaicking: color filter array interpolation. IEEE Signal processing magazine 22(1): 44–54. DOI: https://dx.doi.org/10.1109/MSP.2005.1407714.

[5] Seitz S, Curless B, Diebel J, Scharstein D & Szeliski R (2006) A comparison and evaluation of multi-view stereo reconstruction algorithms. Proc. 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. 519–528. DOI: https://dx.doi.org/10.1109/CVPR.2006.19.

[6] Hernández C (2019), Lens blur in the new google camera app. Google Research Blog. URL: http://googleresearch.blogspot.mx/2014/04/lens-blur-in-new-google-cameraapp.html. Accessed 3.5.2019.

[7] Hamzah R & Ibrahim H (2016) Literature survey on stereo vision disparity map algorithms. Journal of Sensors 2016(2): 1–23. DOI: https://dx.doi.org/10.1155/2016/8742920.

[8] Levoy M (2017), Portrait mode on the Pixel 2 and Pixel 2 XL smartphones. Google Research Blog. URL: https://ai.googleblog.com/2017/10/portrait-mode-on-pixel-2-and-pixel-2-xl.html. Accessed 3.5.2019.

[9] Abuolaim A, Punnappurath A & Brown M (2018) Revisiting autofocus for smartphone cameras. Proc. 2018 European Conference on Computer Vision. 523–537. DOI: https://doi.org/10.1007/978-3-030-01267-032.

[10] Chan C & Chen H (2018) Improving the reliability of phase detection autofocus. Electronic Imaging 2018, 1–5. DOI: https://doi.org/10.2352/ISSN.2470-1173.2018.05.PMII-241.

[11] Śliwiński P & Wachel P (2013) A simple model for on-sensor phase-detection autofocusing algorithm. Journal of Computer and Communications 1(6): 11–17. DOI: https://doi.org/10.4236/jcc.2013.16003.

[12] Adelson E & Wang J (1992) Single lens stereo with a plenoptic camera. IEEE Transactions on Pattern Analysis & Machine Intelligence 14(2): 99–106. DOI: https://doi.org/10.1109/34.121783.

[13] Jeon H, Park J, Choe G, Park J, Bok Y, Tai Y & So Kweon I (2015) Accurate depth map estimation from a lenslet light field camera. Proc. 2015 IEEE conference on computer vision and pattern recognition. 1547–1555. DOI: https://doi.org/10.1109/CVPR.2015.7298762.

[14] Wadhwa N, Garg R, Jacobs D, Feldman B, Kanazawa N, Carroll R, Movshovitz-Attias Y, Barron J, Pritch Y & Levoy M (2018) Synthetic depth-of-field with a single-camera mobile phone. ACM Transactions on Graphics 37(4): 64:1–13. DOI: https://doi.org/10.1145/3197517.3201329.

[15] Garg R, Wadhwa N, Ansari S & Barron J (2019) Learning single camera depth estimation using dual-pixels. arXiv preprint arXiv:1904.05822.

[16] Zhang S (2018) High-speed 3d shape measurement with structured light methods: A review. Optics and Lasers in Engineering 106: 119–131. DOI: https://doi.org/10.1016/j.optlaseng.2018.02.017.

[17] Scharstein D & Szeliski R (2003) High-accuracy stereo depth maps using structured light. Proc. 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. DOI: https://doi.org/10.1109/CVPR.2003.1211354.

[18] Ryan Fanello S, Rhemann C, Tankovich V, Kowdle A, Orts Escolano S, Kim D & Izadi S (2016) Hyperdepth: Learning depth from structured light without matching. Proc. 2016 IEEE Conference on Computer Vision and Pattern Recognition. 5441–5450. DOI: https://doi.org/10.1109/CVPR.2016.587

[19] Geng J (2011) Structured-light 3d surface imaging: a tutorial. Advances in Optics and Photonics 3(2): 128–160. DOI: https://doi.org/10.1364/AOP.3.000128

[20] Büttgen B, Oggier T, Lehmann M, Kaufmann R & Lustenberger F (2005) CCD/CMOS lock-in pixel for range imaging: Challenges, limitations and state-of-the-art. Proc. 1st range imaging research day: 21–32.

[21] Zhang Z, Deriche R, Faugeras O & Luong Q (1995) A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. Artificial intelligence 78(1–2): 87–119. DOI: https://doi.org/10.1016/0004-3702(95)00022-4

[22] Faugeras O (1993) Three-dimensional computer vision: a geometric viewpoint. London, The MIT Press.

[23] Villa-Uriol M, Chaudhary G, Kuester F, Hutchinson T & Bagherzadeh N (2004) Extracting 3d from 2d: selection basis for camera calibration. Proc. Computer Graphics and Imaging: 315–321.

[24] Heikkilä J (1997) Accurate camera calibration and feature based 3-D reconstruction from monocular image sequences. Oulun yliopisto, Department of Electrical Engineering.

[25] Maimone M & Shafer S (1995) Modeling foreshortening in stereo vision using local spatial frequency. Proc. 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. 519–524. DOI: https://doi.org/10.1109/IROS.1995.525846.

[26] Fusiello A, Trucco E & Verri A (2000) A compact algorithm for rectification of stereo pairs. Machine Vision and Applications 12(1): 16–22. DOI: https://doi.org/10.1007/s001380050120.

[27] Scharstein D & Szeliski R (2002) A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. International journal of computer vision 47(1): 7–42. DOI: https://doi.org/10.1023/A:1014573219977.

[28] Lowe D (2004) Distinctive image features from scale-invariant keypoints. International journal of computer vision 60(2): 91–110. DOI: https://doi.org/10.1023/B:VISI.0000029664.99615.94.

[29] Bay H, Ess A, Tuytelaars T & Van Gool L (2008) Speeded-up robust features (SURF). Computer vision and image understanding 110(3): 346–359. DOI: https://doi.org/10.1016/j.cviu.2007.09.014.

[30] Rublee E, Rabaud V, Konolige K & Bradski G (2011) ORB: An efficient alternative to SIFT or SURF. Proc. 2011 International Conference on Computer Vision. 2564–2571. DOI: https://doi.org/10.1109/ICCV.2011.6126544.

[31] Karami E, Prasad S & Shehata M (2015) Image matching using SIFT, SURF, BRIEF and ORB: performance comparison for distorted images. Proc. 2015 Newfoundland Electrical and Computer Engineering Conference.

[32] Tareen S & Saleem Z (2018) A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK. Proc. 2018 International Conference on Computing, Mathematics and Engineering Technologies. 1–10. DOI: https://doi.org/10.1109/ICOMET.2018.8346440

[33] Hauser W, Neveu B, Jourdain J, Viard C & Guichard F (2018) Image quality benchmark of computational bokeh. Electronic Imaging 2018(340): 1–10. DOI: https://doi.org/10.2352/ISSN.2470-1173.2018.12.IQSP-340

[34] Nasse H (2010) Depth of field and bokeh. Carl Zeiss camera lens division report.

[35] Kosloff T & Barsky B (2010) Three techniques for rendering generalized depth of field effects. Proc. 2009 SIAM Conference on Mathematics for Industry: Challenges and Frontiers. 42–48. DOI: https://doi.org/10.1137/1.9781611973303.6.

[36] McIntosh L, Riecke B & DiPaola S (2012) Efficiently simulating the bokeh of polygonal apertures in a post-process depth of field shader. Computer Graphics Forum 31(6): 1810–1822. DOI: https://doi.org/10.1111/j.1467-8659.2012.02097.x.

[37] Javornik A, Rogers Y, Gander D & Moutinho A (2017) Magicface: Stepping into character through an augmented reality mirror. Proc. 2017 CHI Conference on Human Factors in Computing Systems. 4838–4849. DOI: https://doi.org/10.1145/3025453.3025722.

[38] Chatzopoulos D, Bermejo C, Huang Z & Hui P (2017) Mobile augmented reality survey: From where we are to where we go. Ieee Access 5, 6917–6950. DOI: https://doi.org/10.1109/ACCESS.2017.2698164.

[39] Atoum Y, Liu Y, Jourabloo A & Liu X (2017) Face anti-spoofing using patch and depth-based CNNs. Proc. 2017 IEEE International Joint Conference on Biometrics. 319–328. DOI: https://doi.org/10.1109/BTAS.2017.8272713.

[40] Narasimhan S & Nayar S (2002) Vision and the atmosphere. International journal of computer vision 48(3): 233–254. DOI: https://doi.org/10.1023/A:1016328200723.

[41] Li Y, You S, Brown M & Tan R (2017) Haze visibility enhancement: A survey and quantitative benchmarking. Computer Vision and Image Understanding 165: 1–16. DOI: https://doi.org/10.1016/j.cviu.2017.09.003.

[42] Fattal R (2008) Single image dehazing. ACM Transactions on graphics 27(3): 72:1–72:9. DOI: https://doi.org/10.1145/1399504.1360671.

[43] Levin A, Lischinski D & Weiss Y (2004) Colorization using optimization. ACM transactions on graphics 23(3): 689–694. DOI: https://doi.org/10.1145/1186562.1015780.

[44] Barron J & Poole B (2016) The fast bilateral solver. Proc. 2015 European Conference on Computer Vision. 617–632. DOI: https://doi.org/10.1007/978-3-319-46487-9_38.

[45] Ramasamy S, Sabatini R, Gardi A & Liu J (2016) Lidar obstacle warning and avoidance system for unmanned aerial vehicle sense-and-avoid. Aerospace Science and Technology 55: 344–358. DOI: https://doi.org/10.1016/j.ast.2016.05.020.

[46] Giusti A, Guzzi J, Cireşan D, He F, Rodríguez J, Fontana F, Faessler M, Forster C, Schmidhuber J & Di Caro G (2016) A machine learning approach to visual perception of forest trails for mobile robots. IEEE Robotics and Automation Letters 1(2): 661–667. DOI: https://doi.org/10.1109/LRA.2015.2509024.

[47] Barry A, Florence P & Tedrake R (2018) High-speed autonomous obstacle avoidance with pushbroom stereo. Journal of Field Robotics 35(1): 52–68. DOI: https://doi.org/10.1002/rob.21741.

[48] Arias-Castro E & Donoho D (2009) Does median filtering truly preserve edges better than linear filtering? The Annals of Statistics 37(3): 1172–1206. https://doi.org/10.1214/08-AOS604.

[49] Smith S (1997) The scientist and engineer's guide to digital signal processing. San Diego, California Technical Pub.

[50] He K, Sun J & Tang X (2012) Guided image filtering. IEEE transactions on pattern analysis and machine intelligence 35(6): 1397–1409. DOI: https://doi.org/10.1109/TPAMI.2012.213.

[51] He K. & Sun J. (2015) Fast guided filter. arXiv preprint arXiv:1505.00996 .

[52] Petschnigg G, Szeliski R, Agrawala M, Cohen M, Hoppe H & Toyama K (2004) Digital photography with flash and no-flash image pairs. ACM transactions on graphics 23(3): 664–672. DOI: https://doi.org/10.1145/1015706.1015777.

[53] Eisemann E & Durand F (2004) Flash photography enhancement via intrinsic relighting. ACM transactions on graphics 23(3): 673–678. DOI: https://doi.org/10.1145/1015706.1015778.

[54] Min D, Choi S, Lu J, Ham B, Sohn K & Do M (2014) Fast global image smoothing based on weighted least squares. IEEE Transactions on Image Processing 23(12): 5638–5653. DOI: https://doi.org/10.1109/TIP.2014.2366600.

[55] Lagendijk R, Biemond J & Boekee D (1988) Regularized iterative image restoration with ringing reduction. IEEE Transactions on Acoustics, Speech, and Signal Processing 36(12): 1874–1888. DOI: https://doi.org/10.1109/29.9032.

[56] Farbman Z, Fattal R, Lischinski D & Szeliski R (2008) Edge-preserving decompositions for multi-scale tone and detail manipulation. ACM Transactions on Graphics 27(3): 67. DOI: https://doi.org/10.1145/1399504.1360666.

[57] Liu W, Chen X, Shen C, Liu Z & Yang J (2017) Semi-global weighted least squares in image filtering. 2017 IEEE International Conference on Computer Vision. 5861–5869. DOI: https://doi.org/10.1109/ICCV.2017.624.

[58] Bull D (2014) Chapter 3 - discrete-time analysis for images and video. Communicating Pictures. Oxford, Academic Press, 63 – 98.

[59] Tomasi C & Manduchi R (1998) Bilateral filtering for gray and color images. Sixth International Conference on Computer Vision. 839–846. DOI: https://doi.org/10.1109/ICCV.1998.710815.

[60] Elad M (2002) On the origin of the bilateral filter and ways to improve it. IEEE Transactions on image processing 11(10): 1141–1151. DOI: https://doi.org/10.1109/TIP.2002.801126.

[61] Buades A, Coll B & Morel J (2006) The staircasing effect in neighborhood filters and its solution. IEEE transactions on Image Processing 15(6): 1499–1505. DOI: https://doi.org/10.1109/TIP.2006.871137.

[62] Chen J, Paris S & Durand F (2007) Real-time edge-aware image processing with the bilateral grid. ACM Transactions on Graphics 26(3): 103. DOI: https://doi.org/10.1145/1276377.1276506.

[63] Richardt C, Orr D, Davies I, Criminisi A & Dodgson N (2010) Real-time spatiotemporal stereo matching using the dual-cross-bilateral grid. 2010 European conference on Computer vision. 510–523. DOI: https://doi.org/10.1007/978-3-642-15558-1_37.

[64] Barron J, Adams A, Shih Y & Hernández C (2015) Fast bilateral-space stereo for synthetic defocus. 2015 IEEE Conference on Computer Vision and Pattern Recognition. 4466–4474. DOI: https://doi.org/10.1109/CVPR.2015.7299076.

[65] Adams A, Baek J & Davis M (2010) Fast high-dimensional filtering using the permutohedral lattice. Computer Graphics Forum 29(2): 753–762. DOI: https://doi.org/10.1111/j.1467-8659.2009.01645.x.

[66] Valentin J, Kowdle A, Barron J, Wadhwa N, Dzitsiuk M, Schoenberg M, Verma V, Csaszar A, Turner E, Dryanovski I, Afonso J, Pascoal J, Tsotsos K, Leung M, Schmidt M, Guleryuz O, Khamis S, Tankovitch V, Fanello S, Izadi S & Rhemann C (2018) Depth from motion for smartphone AR. ACM Transactions on Graphics 37(6): 1–19. DOI: https://doi.org/10.1145/3272127.3275041.

[67] Anderson R, Gallup D, Barron J, Kontkanen J, Snavely N, Hernández C, Agarwal S & Seitz S (2016) Jump: virtual reality video. ACM Transactions on Graphics 35(6): 1–13. DOI: https://doi.org/10.1145/2980179.2980257.

[68] MATLAB version 9.5.0.1049112 (R2018b) (2018) The Mathworks, Inc, Natick, Massachusetts.

[69] Jain P & Tyagi V (2016) A survey of edge-preserving image denoising methods. Information Systems Frontiers 18(1): 159–170. DOI: https://doi.org/10.1007/s10796-014-9527-0.

[70] Goyal A, Bijalwan A & Chowdhury M (2012) A comprehensive review of image smoothing techniques. International Journal of Advanced Research in Computer Engineering & Technology 1(4): 315–319.

[71] Pal C, Chakrabarti A & Ghosh R (2015) A brief survey of recent edge-preserving smoothing algorithms on digital images. arXiv preprint arXiv:1503.07297.

[72] Lee V, Kim C, Chhugani J, Deisher M, Kim D, Nguyen A, Satish N, Smelyanskiy M, Chennupaty S, Hammarlund P (2010) Debunking the 100x GPU vs. CPU myth: an evaluation of throughput computing on CPU and GPU. ACM SIGARCH computer architecture news 38(3): 451–460. DOI: https://doi.org/10.1145/1816038.1816021.

[73] Eklund A, Dufort P, Forsberg D & LaConte S (2013) Medical image processing on the GPU – past, present and future. Medical image analysis 17(8): 1073–1094. DOI: https://doi.org/10.1016/j.media.2013.05.008

[74] Kalaiselvi T, Sriramakrishnan P & Somasundaram K (2017) Survey of using GPU CUDA programming model in medical image analysis. Informatics in Medicine Unlocked 9: 133–144. DOI: https://doi.org/10.1016/j.imu.2017.08.001.

[75] Zhou J & Ross K (2002) Implementing database operations using SIMD instructions. 2002 ACM SIGMOD international conference on Management of data. 145–156. DOI: https://doi.org/10.1145/564691.564709.

[76] NEON: Programmer's Guide (2013) ARM Limited, Cambridge.

[77] El Mezeni D & Saranovac L (2016) Fast self-guided filter with decimated box filters. International Symposium INFOTEH–JAHORINA. 633–638.

[78] Mazumdar A, Alaghi A, Barron J, Gallup D, Ceze L, Oskin M & Seitz S (2017) A hardware-friendly bilateral solver for real-time virtual reality video. 2017 High Performance Graphics 13. DOI: https://doi.org/10.1145/3105762.3105772.

[79] Scharstein D, Szeliski R & Hirschmüller H (), Middlebury stereo evaluation version 3. Middlebury Stereo Vision Page. URL: http://vision.middlebury.edu/stereo/eval3/. Accessed 27.6.2019.

[80] Menze M & Geiger A (2015) Object scene flow for autonomous vehicles. 2015 IEEE Conference on Computer Vision and Pattern Recognition. 3061–3070. DOI: https://doi.org/10.1109/CVPR.2015.7298925.

[81] Schops T, Schonberger J, Galliani S, Sattler T, Schindler K, Pollefeys M & Geiger A (2017) A multi-view stereo benchmark with high-resolution images and multi-camera videos. 2017 IEEE Conference on Computer Vision and Pattern Recognition. 3260–3269. DOI: https://doi.org/10.1109/CVPR.2017.272.

[82] Butler D, Wulff J, Stanley G & Black M (2012) A naturalistic open source movie for optical flow evaluation. 2012 European conference on computer vision. 611–625. DOI: https://doi.org/10.1007/978-3-642-33783-3_44.

[83] Hirschmuller H & Scharstein D (2007) Evaluation of cost functions for stereo matching. 2007 IEEE Conference on Computer Vision and Pattern Recognition, 1–8. DOI: https://doi.org/10.1109/CVPR.2007.383248.

[84] Scharstein D & Pal C (2007) Learning conditional random fields for stereo. In: 2007 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, pp. 1–8. DOI: https://doi.org/10.1109/CVPR.2007.383191.

[85] Scharstein D, Hirschmüller H, Kitajima Y, Krathwohl G, Nešić N, Wang X & Westling P (2014) High-resolution stereo datasets with subpixel-accurate ground truth. 2014 German conference on pattern recognition. 31–42. https://doi.org/10.1007/978-3-319-11752-2_3.

[86] Chai T & Draxler R (2014) Root mean square error (RMSE) or mean absolute error (MAE)? – arguments against avoiding RMSE in the literature. Geoscientific model development 7(3): 1247–1250. DOI: https://doi.org/10.5194/gmd-7-1247-2014.

[87] Sheikh H & Bovik A (2006) Image information and visual quality. IEEE Transactions on image processing 15(2): 430–444. DOI: https://doi.org/10.1109/TIP.2005.859378.

[88] Sheikh H, Sabir M & Bovik A (2006) A statistical evaluation of recent full reference image quality assessment algorithms. IEEE Transactions on image processing 15(11): 3440–3451. DOI: https://doi.org/10.1109/TIP.2006.881959.