
Master Thesis

- Implementation of stereo vision engine -

Project Report
Group 1072

Aalborg University
Electronics and IT

Copyright © Aalborg University 2015

Here you can write something about which tools and software you have used for typesetting the document, running simulations and creating figures. If you do not know what to write, either leave this page blank or have a look at the colophon in some of your books.??



Electronics and IT
Aalborg University
<http://www.aau.dk>

AALBORG UNIVERSITY

STUDENT REPORT

Title:

Stereo vision implementation??

Abstract:

Here is the abstract

Theme:

Master Thesis??

Project Period:

Spring Semester 2016

Project Group:

1072

Participant(s):

Tomas Brandt Trillingsgaard

Supervisor(s):

Peter Koch

Copies: 4

Page Numbers: 43

Date of Completion:

September 7, 2016

The content of this report is freely available, but publication (with reference) may only be pursued due to agreement with the author.

Contents

Preface	ix
1 Introduction	1
1.1 Stereo vision introduction	1
1.2 Motivation	2
1.3 Problem Introduction	2
1.4 Delimitation	2
1.5 Report Structure and Design Process	2
2 Application Analysis	5
2.1 basic principal of stereo vision	5
2.2 Rectification of stereo pairs	7
2.3 Color space and gray scale	8
2.4 Resolution and disparity precision	8
2.5 Occlusion filling	8
2.5.1 Neighbor's Disparity Assignment : NDA	8
2.5.2 Diffusion in Intensity Space : DIS	9
2.5.3 Weighted Least Squares : WLS	9
2.5.4 Segmentation-based Least Squares : SLS	11
2.6 Mini-conclusion	12
3 Requirements	13
3.1 Requirement specification	13
3.2 Test specification	13
4 Algorithm design	15
4.1 Efficient Edge Preserving Stereo Matching:	15
4.2 Fast Cost-Volume Matching:	16
4.2.1 Guided image filter	16
4.3 Simulation and comparison	17
4.4 Choosing an algorithm	17

5	Platform Analysis	19
6	Design methodology	21
7	Architecture design	25
7.1	Parallelism Analysis	25
7.2	Allocation and Scheduling	26
7.2.1	RC scheduling	27
7.2.2	TC schedule	29
7.3	FSMD design	29
7.4	VHDL + Simulation	31
8	Acceptance test	35
9	Conclusion	37
A	Allocation test	39
A.1	General procedure	39
A.2	Adder	40
A.3	Subtract	40
A.4	Adder/Subtract	40
A.5	Multiplier - LUT	40
A.6	Multiplier - DSP48	41
A.7	Divider	41
B	Extra Figures	43

Todo list

ikke færdig	5
ret denne formel til på et tidspunkt. 2.2 er korrekt. Ændre den anden + måske figuren (z går vist fra baseline og ikke bare fra image plane) . . .	7
skriv noget om forskellige farve rum og grayscale og deres indflydelse på stereo algorithmen	8
skriv noget om disparitets opløsning i forhold til billede opløsning osv. .	8
skriv noget om metoder til at udfylde occlusions områder	8
find på et bedre navn til denne sektion	12
Få lavet en tabel som indeholder kravene	13
Regner med at lave en test hvor jeg med min egen python simulering trækker dataen ud lige før hvor dataen skal bruges i det jeg har fået lavet et hardware design af. så vil jeg samligne med middlebury test sets	13
skriv hvordan jeg vil teste de forskellige krav	13
beskrive middlebury test sets her? Nej beskriv dem i appendix	13
ROUGH SKETCH not done yet	15
simulation af de 2 algorithmers og samlign resultaterne.	17
Nok en anden titel til denne sektion. Skriv hvilken algoritme jeg går videre med	17
beskriv Zynq platformen. kom ind på hvad den indeholder	19
bedre navn og cite ds190 product specification og måske trm?	19
FPGA constraints ==> $C = f(A, T, P, N)$, Lav en tabel	19
læs om system design methodologies i gajski's Embedded Systems Design - Modeling, Synthesis and Verification og beskriv Platform Methodology	21
NOT DONE! rough sketch. De nedenstående trin er hvad jeg skal igennem: Para. Anal., Alloc., Optimizaiton, FSMD og VHDL + simulering	25
lav om til operation og husk mult	27
find et bedre ord	29
udfør accept test udfra test specifikationen (brug data fra python simulering og giv det til VHDL implementationen)	35

Preface

Here is the preface. You should put your signatures at the end of the preface.

Aalborg University, September 7, 2016

Tomas Brandt Trillingsgaard
<ttrill10@student.aau.dk>

Chapter 1

Introduction

In this chapter, the project is introduced and motivated. Furthermore, a brief description is presented for stereo vision and the use for it at HSA systems. Lastly, this chapter also describes a delimitation of the project and report.

1.1 Stereo vision introduction

In 280 A.D the greek mathematician Euclid noticed that the perception of depth is caused by each eye receiving a dissimilar image of the same object. Throughout history different people have been working on this this concept. In 1933 Sir Charles Wheatstone began mimic and forced the perception of depth by developing the stereoscope. The stereoscope use images taken by two cameras placed next to each other and the stereoscope then isolated the images so each eye only see one image. This mimics the depth perception and the viewer would be able to see depth in the images. [lit:historyofstereophoto]

Beginning around 1970 computer vision began appearing and a big part of this area is stereo vision, the measurement of depth mimicing the human vision using two cameras [Szeliski2010]. The ability to measure depths enables a computer to distinguish between objects and hence better interact and react on the world.

HSA systems wish to follow packages going through their system. A strategically placed stereo vision camera will enable them to know how many and where these packages are in the system. In case of errors and the like the printer system is the able to notice when the conveyor belt is empty and ready to reset.

In future uses HSA system would like the stereo vision system to have very precise depth measurements of 2 mm.

1.2 Motivation

The area of stereo vision have be researched thoroughly and precise algorithms have been found but most algorithms are very heavy computational wise. HSA system wish for a stereo vision system which can run real-time (10 fps) and have a high precision of 2 mm.

1.3 Problem Introduction

HSA systems wish to follow packages going through their system. A strategically placed stereo vision camera will enable them to know how many and where these objects are in the system. The primary objectives of this is to:

- Analyze obstacles within stereo vision
- Analyze different stereo algorithms
- Design and optimize an architecture for executing stereo vision

1.4 Delimitation

This project is mainly concerned with the design and implementation of a hardware design for a FPGA. This project will not focus on developing a new stereo vision algorithm. Obstacles and issue with stereo algorithms will be analyzed but simpler solution will be used for most obstacles.

1.5 Report Structure and Design Process

The A³ model is a basic design model used internally at AAU and is illustrated on figure 1.1 on the facing page. The model consist of tree design domains which it will explore and it will help structure this report. These domains are Application, Algorithm and Architecture.

The search for a solution starts in the application domain where the problem and application are being explored. Chapter 2 on page 5: Application Analysis will explore the Application domain and the resulting specification for the application are presented in chapter 3 on page 13. In this project not a lot of applications are explored since the problem and the wanted application have been given by HSA system.

When moving to a new domain it can result in landing in different areas of the domain as seen on figure 1.1 on the next page. The solid arrows goes from a single

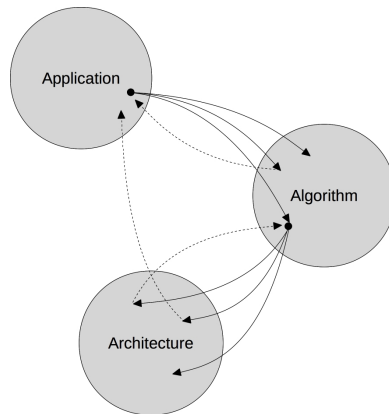


Figure 1.1: A³ model

point in the preceding domain and spreads out to different points in the current domain and these different points can, in the case of the algorithm domain, represent different algorithms. Chapter 4 on page 15 will explore the algorithm domain.

Chapter 6 on page 21 will describe a methodology to go from an algorithm in the algorithm domain to some hardware architecture in the architecture domain.

Notice the dashed arrows in figure 1.1 on page 3. These arrows show that when you are exploring some domains new information might be given which requires you to go back to a former domain and change the found specification etc. and then start moving to a new domain from new.

Chapter 8 on page 35 will test whether the found solution complies with the requirements in 3 on page 13 and chapter 9 on page 37 will conclude the project and its findings.

Chapter 2

Application Analysis

This chapter starts by describes the basic principles of stereo vision then different aspects such as color versus gray scale etc are analyzed.

ikke færdig

2.1 basic principal of stereo vision

A stereo vision setup normally consists of two cameras placed horizontally at a specified distance from each other (the baseline). An example of this is on figure 2.1

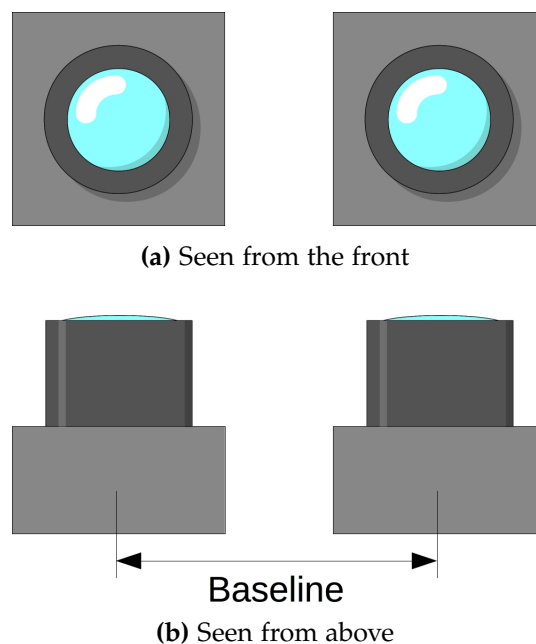


Figure 2.1: Illustration of stereo setup

Figure 2.2 shows how a scene is seen by the camera, is inverted in the optical center and projected onto the image sensor in the camera. The original image plane is located at the position of the image sensor but it is inverted compared to scene captured. To simplified comparisons to the real world an image plane can be placed opposite of the optical center at the same distance from the center and this image plane will not be inverted.

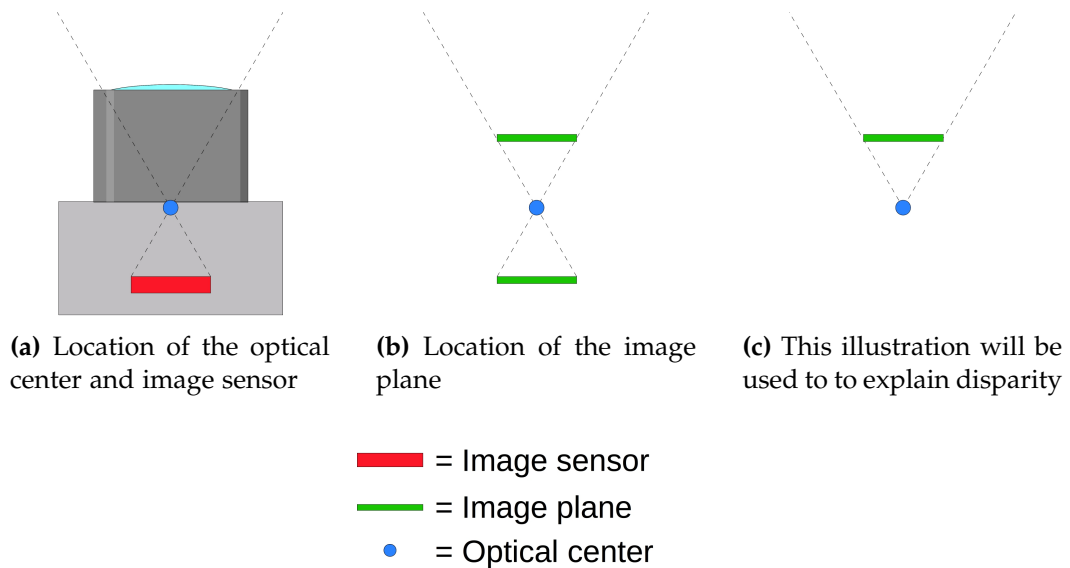


Figure 2.2: Illustration of going from camera to image plane

Figure 2.3a shows that a single camera is not able to differentiate between two points at the same angle from the optical center but a different distance. Figure 2.3b shows that adding the second camera shows that you then are able to differentiate between the two points.

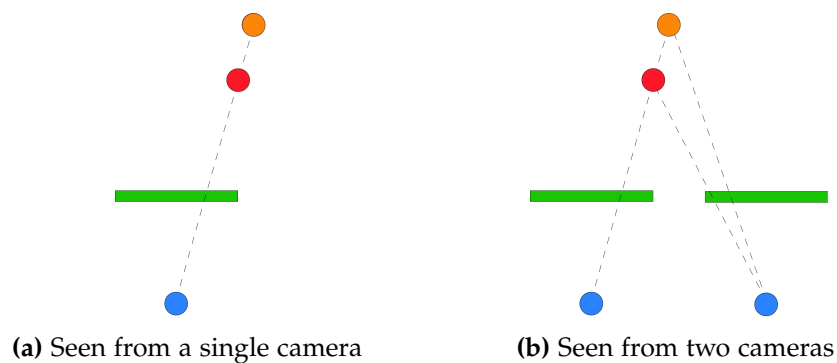


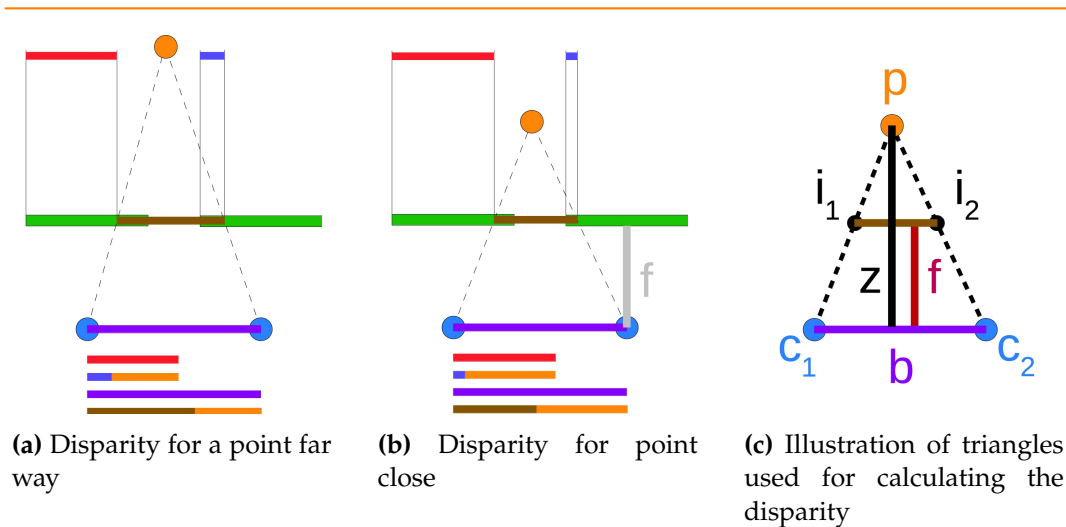
Figure 2.3: Example of two points in a scene at different depths

Figure 2.4 shows how the depth to a point can be calculated from the difference in

x-positions on the image plane (the disparity). Figure 2.4a and 2.4b shows how the disparity change depending on the distance to the point. Two similar triangles can be created. One between the point, and the two optical centers. The other triangle is created between the point in the scene and the points where the dashed lines cross the image plane. Figure 2.4c shows these two triangles and their heights. The small triangle have a height of z and the bottom (the brown line) is equal to the baseline (purple line) minus the disparity and the large triangle have a height of $f + z$ and the bottom width is equal to the baseline. The ratio between the height and the bottom width is the same in the two triangles and hence the following equation can be formed:

$$\frac{z + f}{b} = \frac{z}{b - d} \quad (2.1)$$

$$z = \frac{b \cdot f}{d} \quad (2.2)$$



ret denne formel til på et tidspunkt. 2.2 er korrekt. Ændre den anden + måske figuren (z går vist fra baseline og ikke bare fra image plane)

Figure 2.4: Illustration of how to calculate depth from disparity

This explains the basic of stereo vision. The rest of this chapter will venture into other areals of stereo and describe the difficulties and solutions for each area.

2.2 Rectification of stereo pairs

The examples in section 2.1 uses ideal images to explain how stereo vision functions but in the real world the disparity is the difference along the epipolar lines which will have a different slope of each line and image. Different things can be done to simplify the search along epipolar lines. One thing is to rectify the stereo

image pair. This will make all the epipolar lines horizontal and hence you only need to search along the x-axis.

The issue with rectification is that it is difficult to get a perfect match with the stereo setup since the every camera and its objective will be unique and require and all new manual adjustment. HSA system theorizes that a system can be developed which instead of rectifying the image it will find the epipolar lines and feed this information to the stereo camera. In this project stereo image pairs from Middlebury Vision Test set [da] will be used which are rectified hence the rectification of image will not be researched further in this project.

2.3 Color space and gray scale

skriv noget om forskellige farve rum og grayscale og deres indflydelse på stereo algoritmen

The article **Color correlation-based matching** takes the subject of difference in result when using color and which color space is used and grayscale when performing stereo matching. It performs different methods / algorithms using 9 different colorspace including grayscale. The result from the article is that color gives a better result with a few percentage of more correct estimations but the run time is much higher (ranging from 1.9 to 3.7 higher run time than grayscale on the teddy test set). From this it is decided to not use color in case of Normalized Cross Correlation

2.4 Resolution and disparity precision

skriv noget om disparities opløsning i forhold til billede opløsning osv.

2.5 Occlusion filling

skriv noget om metoder til at udfylde occlusions områder

This section will describe methods for filling the occluded areas. All these methods comes from the article: *Occlusion filling in stereo: Theory and experiments* by Shafik Hyq, Andreas Koschan and Mongi Abidi. All these methods assume that the stereo matching is going from left image to right image i.e. templates are taken from the left image matched onto the right image.

2.5.1 Neighbor's Disparity Assignment : NDA

This is the simplest method to fill occlusions. It functions by selecting an occluded point, p_L , then find then nearest non-occluded point, q_L , to the left when filling non-border occlusion. With border occlusion the nearest point to the right is found instead. It is assumed that this non-occluded point is part of same surface as the occluded point (this can be seen on figure ??) and the disparity value from q_L can

be assigned to p_L . This method have some issues. In cases of total occlusions (see figure ??) then a wrong disparity value is given to the total occluded object since it isn't a part of the nearest surface with non-occluded points to the left. In cases with self occlusions the occluded area should have disparity values close to the disparity values of the non-occluded points to the right (This will be the area of the surface which is in view of both cameras) but using NDA will give the occluded area disparity values corresponding to the background.

2.5.2 Diffusion in Intensity Space : DIS

This method is inspired by diffusion. Diffusion is the movement of molecules or atoms from a high concentration region to a low concentration region.

After detecting occluded regions with cross-checking during template matching, the diffusion energy for the region is approximated. This method is depended on the stereo matching algorithm because it use the energy from the last iteration to determine initial diffusion energy for the area.

A change to the method can be made to make it independent from the stereo matching. The initial energy will be 0. Then the diffusion energy for non-border occlusion is found by:

$$E(p_L) = \min_{l_{p_L}=\{0,\dots,l_{max}\}} \left(\frac{1}{2|q_L \in \mathcal{N}(p_L) \wedge l_{q_L}=l_{p_L}|} \sum_{q_L \in \mathcal{N}(p_L) \wedge l_{q_L}=l_{p_L}} (|\bar{I}(p_L) - \bar{I}(q_L)| + E(q_L)) \right) \quad (2.3)$$

And the diffusion energy for border occlusions are found by by:

$$E(p_L) = \min_{l_{p_L}=\{0,\dots,l_{p_{Lf}}-2\}} \left(\frac{1}{2|q_L \in \mathcal{N}(p_L) \wedge l_{q_L}=l_{p_L}|} \sum_{q_L \in \mathcal{N}(p_L) \wedge l_{q_L}=l_{p_L}} (|\bar{I}(p_L) - \bar{I}(q_L)| + E(q_L)) \right) \quad (2.4)$$

The diffusion energy will be calculated for each occluded point and for each point the disparity which corresponds the minimum $E(p_L)$ is set as the disparity l_{p_L} for the occluded point.

2.5.3 Weighted Least Squares : WLS

In this approach, WLS, all the non-occluded and filled occluded neighbors in a neighborhood around the occluded point is considered valid points and is used as control points in interpolation.

Since the neighborhood contains both foreground points and background points and the occluded point is expected to be a part of the background then the background points should have more influence than foreground points. It is assumed that the color intensity between objects is significantly different and this property

can be used to distinguish between foreground points and background points. Each error term in the aggregated residual should be weighted so the foreground don't have much influence. With this the aggregated residual is defined as:

$$\Delta = \sum_{q_L \in \mathcal{N}(p_L)} w_{q_L} (\hat{l}_{p_L}(p_L) - l_{p_L}(q_L))^2 \quad (2.5)$$

where $w_{q_L} = e^{-\mu_L |\bar{I}(p_L) - I(q_L)|}$ (the weight) is the likelihood of p_L with q_L under the assumption of an exponential distribution model of $|\bar{I}(p_L) - I(q_L)|$. $\bar{I}(p_L)$ is the mean intensity of p_L and μ_L is the decay rate. $\hat{l}_{p_L}(p_L)$ is the estimated disparity of p_L (will be estimated during interpolation) and $l_{p_L}(q_L)$ is the disparity of q_L .

How to estimate $\bar{I}(p_L)$ and μ_L :

$\bar{I}(p_L)$ is the mean intensity of p_L which can be obtained using mean shift algorithm in a window around p_L . To estimate this value the initialize the algorithm with $\bar{I}(p_L)$ equal to the intensity of p_L then the mean shift algorithm repeatedly picks those neighbors inside the window that satisfy $|\bar{I}(p_L) - I(q_L)| \geq 3\mu^{-1}$ and then assign the average of intensities of the selected neighbors to $\bar{I}(p_L)$ until $\bar{I}(p_L)$ converges to a fixed average. $|\bar{I}(p_L) - I(q_L)|$ has decay rate μ_L which is related to the decay rate μ of the variable $|I(p_L) - I(q_L)|$ by $\mu_L^2 = \mu$.

A matrix containing all the coordinates:

$$F = \begin{bmatrix} x_1 & y_1 & 1 \\ \vdots & \ddots & \vdots \\ x_n & y_n & 1 \end{bmatrix} \quad (2.6)$$

Vector with the corresponding labels for the coordinates in F :

$$L = [l_1 \cdots l_N] \quad (2.7)$$

Linear model:

$$l_{p_L} = a + bx(p_L) + cy(p_L) \quad (2.8)$$

Where $(x(p_L), y(p_L))$ is the coordinates of p_L and a, b and c are the model parameters.

The weights for the control points can be express in a vector as:

$$w = [w_{q_{L1}} \ w_{q_{L2}} \ \cdots \ w_{q_{LN}}]' \quad (2.9)$$

Then we compute two new matrices, F_w and L_w :

$$F_w = \text{diag}(w)F \quad (2.10)$$

$$L_w = \text{diga}(w)L \quad (2.11)$$

The model parameter vector:

$$P = [a \ b \ c]' \quad (2.12)$$

By combining the equations above then the following equation is given:

$$P = (F_w^T F_w)^{-1} F_w^T L_w \quad (2.13)$$

With these equation the disparity of the occluded point can be estimated:

$$\hat{l}_{p_L} = [1 \ x(p_L) \ y(p_L)]P \quad (2.14)$$

2.5.4 Segmentation-based Least Squares : SLS

Biggest difference between WLS and SLS is that SLS only uses non-occluded points as control points. The control points is a subset of the non-occluded neighboring points. The control points are segmented from the neighborhood by applying different constraints: visibility constraint, disparity gradient constraint and color similarity cues.

Sequence of operations:

- Select an occluded point
- Select control points from the neighborhood around the occluded point
- Interpolate the disparity of the occluded point from the segmented control points

$\mathcal{N}(p_L)$ is a set of non-occluded, neighboring points which will be use for control points in the interpolation. For points to be added to \mathcal{N} then it needs to fulfill some constraints.

Disparity gradient constraint: In most cases the horizontal closest non-occluded point to the right, p_{Lf} , will be part of the foreground and the occluded should be a part of the background. In this cases every non-occluded point with a lower disparity than p_{Lf} will be added to \mathcal{N} hence the condition for added the point, q_L , will be $l_{q_L} < l_{p_{Lf}}$. If the foreground object is narrow then all the non-occluded neighboring points might be from the background and have the same disparity. Due to this a second condition have to be added to the constraint. The horizontal closest non-occluded point to the left will be called p_{Lb} and a second condition is created: $|l_{p_{Lb}} - l_{q_L}| \leq 1$. When these conditions are combined the constraint can be defined as:

$$|l_{p_{Lb}} - l_{q_L}| \leq 1 \vee l_{q_L} < l_{p_{Lf}} \quad (2.15)$$

surface constraint: It is assumed that $\mathcal{N}(p_L)$ will contain points from maximum 2 different surfaces (due to the small neighborhood). Some cases might contain a third surface but this is expected to occur very seldom and therefore it is disregarded. The point with the lowest disparity, l_{min} , is assumed to belong to one of the surfaces and the point with the highest disparity, l_{max} , is assumed to belong to the other surfaces. If $l_{max} - l_{min} \leq 1$ then it is assumed the all the points in

\mathcal{N} belongs to a single surfaces otherwise the points have to be segmented into 2 groups. The first group will contain all points which satisfies $|l - max - l_{q_L}| \leq 1$ and the other group will contain all the points which satisfies $|l - min - l_{q_L}| \leq 1$.

Color constraint: The average truncated color distance from the occluded point, p_L , to each of the two groups to determine which group the point belongs to. The average truncated color distance is found by:

$$D(p_L, \mathcal{N}_i(p_L)) = \frac{1}{|\mathcal{N}_i(p_L)|} \sum_{q_L \in \mathcal{N}(p_L)} \psi(p_L, q_L) \quad (2.16)$$

2.6 Mini-conclusion

find på et bedre navn til denne sektion

To conclude this chapter the findings for each area in stereo vision will be examined and it will be specified which solution will be used from this point.

•

Chapter 3

Requirements

3.1 Requirement specification

Få lavet en tabel som indeholder kravene

No.	Parameter	Value	Unit	Additional Information	Source
1	Something something	0 to 48	Mhz	• Something	1
2	Something something	0 to 48	Mhz	• Something	1
General requirements					
• Something something something					

3.2 Test specification

Regner med at lave en test hvor jeg med min egen python simulering trækker dataen ud lige før hvor dataen skal bruges i det jeg har fået lavet et hardware design af. så vil jeg samligne med middlebury test sets

skriv hvordan jeg vil teste de forskellige krav

beskrive middlebury test sets her? Nej beskriv dem i appendix

Chapter 4

Algorithm design

ROUGH SKETCH not done yet

In this chapter the two stereo vision algorithms, Efficient Edge Preserving Stereo Matching (EPPSM) and Fast Cost-Volume Matching (FCV), is described. Lastly, a simulation of each algorithm is created and the results of these simulations are compared and from this, an algorithm is chosen.

4.1 Efficient Edge Preserving Stereo Matching:

This algorithm works in three steps. The first step is calculating a cost for each pixel and disparity. This cost is a combination of the sum of absolute differences and hamming distance of the census transform around each pixel.

$$C_d^{SAD}(x, y) = \sum_{i=1}^3 |I_{left}(x, y, i) - I_{right}(x + d, y, i)| \quad (4.1)$$

$$C_d^{CENSUS}(x, y) = Ham(CT_{left}(x, y), CT_{right}(x + d, y)) \quad (4.2)$$

$$C_d(x, y) = \alpha \cdot C_d^{SAD}(x, y) + (1 - \alpha) \cdot C_d^{CENSUS}(x, y) \quad (4.3)$$

where d is the disparity estimate, I_{left} is the left image, I_{right} is the right image, i is the color (rgb), $Ham(x_1, x_2)$ is the hamming distance between x_1 and x_2 , and CT_{left} and CT_{right} is the census transform around the specified pixel

then a permeability weight is calculated. Permeability is known from biomedicine and describes the ability to transfer through a membrane. The permeability weight is inspired by this and describes how well the color transfers from one pixel to another pixel.

$$\mu(x, y) = \min(e^{\frac{-\Delta R}{\sigma}}, e^{\frac{-\Delta G}{\sigma}}, e^{\frac{-\Delta B}{\sigma}}) \quad (4.4)$$

$$\mu_{tb}(x, y) = \min(e^{\frac{-(R(x, y) - R(x, y-1))}{\sigma}}, e^{\frac{-(G(x, y) - G(x, y-1))}{\sigma}}, e^{\frac{-(B(x, y) - B(x, y-1))}{\sigma}}) \quad (4.5)$$

lastly, the cost is aggregated resulting in a combined cost for each pixel at each disparity. The cost from equation ?? is first aggregated horizontally using permeability weights from equation ?. Then the result from horizontal aggregation is aggregated vertically also using the permeability weight.

$$C_d^{lr}(x, y) = C_d(x, y) + \mu_{lr}(x, y) \cdot C_d(x - 1, y) \quad (4.6)$$

$$C_d^{lr}(x, y) = C_d(x, y) + \sum_{i=1}^{x-1} \left(C_d(x - i, y) \cdot \prod_{j=i}^x \mu_{lr}(x - j, y) \right) \quad (4.7)$$

With a cost at each pixel at each disparity estimate, the disparity map can be generated by minimization along the disparity estimates.

4.2 Fast Cost-Volume Matching:

This algorithm starts by calculating a cost for each pixel at each disparity estimate. This cost consists of the sum of absolute differences and differences in the gradient.

$$C_d^{SAD}(x, y) = \sum_{i=1}^3 |I_{left}(x, y, i) - I_{right}(x + d, y, i)| \quad (4.8)$$

$$C_d^{Grad}(x, y) = \nabla_x I_{left}^g(x, y) - \nabla_x I_{right}^g(x, y) \quad (4.9)$$

$$C_d(x, y) = \alpha \cdot C_d^{SAD}(x, y) + (1 - \alpha) \cdot C_d^{Grad}(x, y) \quad (4.10)$$

These cost values are then filtered using a Guided Image Filter. The guided image filter is a filter uses a reference image to generate the weights. The guided image filter is described further in section 4.2.1

$$C'_d(x, y) = \sum_j W_{i,j}(I) C_d(x, y) \quad (4.11)$$

The correct disparity for each pixel can then be found by minimizing along the disparity estimates as seen in equation 4.12.

$$f(x, y) = \arg \min_{d \in [0, d_{max}]} C'_d(x, y) \quad (4.12)$$

4.2.1 Guided image filter

The guided image filter uses a image as a reference for weighting the input. The output from the filter is seen in equation

$$q_i = \sum_j W_{i,j}(I) p_j \quad (4.13)$$

$$q_i = a_k I_i + b_k, \forall i \in \omega_k \quad (4.14)$$

where:

$$a_k = \frac{\frac{1}{|\omega|} \sum_{i \in \omega_k} I_i p_i - \mu_k \bar{p}_k}{\sigma_k^2 + \epsilon} \quad (4.15)$$

$$b_k = \bar{p}_k - a_k \mu_k \quad (4.16)$$

algorithm

Input:

filtering input image: p

guidance image: I

radius: r

epsilon: ϵ

Output:

filtering output: q

Steps:

1. $\mu_I = f_{mean}(I)$
 $\mu_p = f_{mean}(p)$
 $\rho_{II} = f_{mean}(I \cdot I)$
 $\rho_{Ip} = f_{mean}(I \cdot p)$
2. $\sigma_I = \rho_{II} - \mu_I \cdot \mu_I$
 $cov_{Ip} = \rho_{Ip} - \mu_I \cdot \mu_p$
3. $a = cov_{Ip} / (\sigma_I + epsilon)$
 $b = \mu_p - a \cdot \mu_I$
4. $\mu_a = f_{mean}(a)$
 $\mu_b = f_{mean}(b)$
5. $q = \mu_a \cdot I + \mu_b$

4.3 Simulation and comparison

simulation af de 2 algoritmer og samlign resultaterne.

4.4 Choosing an algorithm

Nok en anden titel til denne sektion. Skriv hvilken algoritme jeg går videre med

Chapter 5

Platform Analysis

5.1

beskriv Zynq platformen.
kom ind på hvad den
indeholder

bedre navn og cite ds190
product specification og
måske trm?

FPGA constraints ==> C =
f(A, T, P, N), Lav en tabel

Part	Quantity
Programmable logic cells	85,000
Look-Up Tables	53,200
Flip-flops	106,400
Extensible Block Ram	560 kB
Programmable DSP slices	220

Table 5.1: Parts on zynq 7020

Chapter 6

Design methodology

In [gajski2009] the Gajski-Kuhn Y-chart is described and it is illustrated on figure 6.1a. This chart can help describe a way to design a system. It moves between the three domains: Behaviour, Structure and Physical.

læs om system design methodologies i gajski's **Embedded Systems Design - Modeling, Synthesis and Verification** og beskriv Platform Methodology

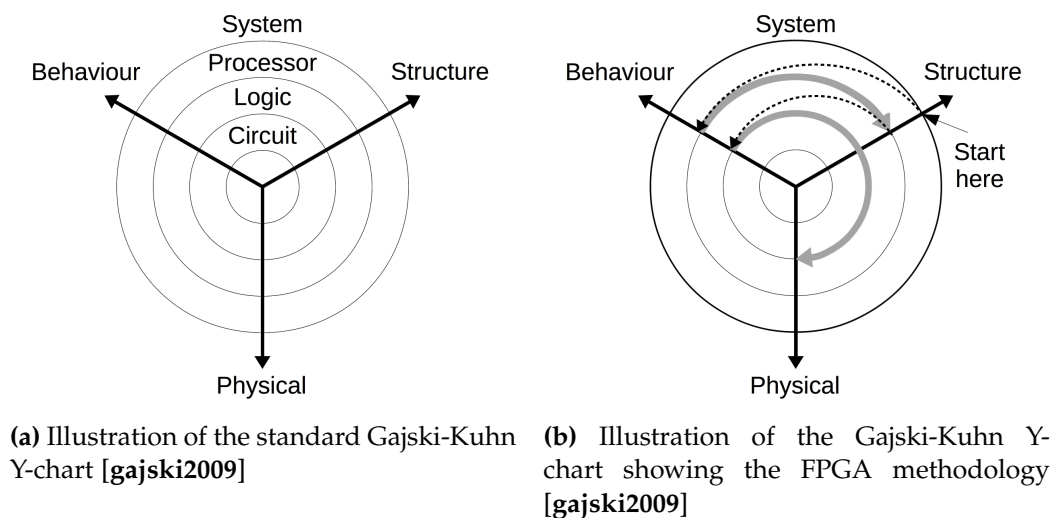


Figure 6.1: Illustrations of Gajski-Kuhn Y-chart

noter til mig selv

processor synthesis:

side 32 i gajski pdf

(a) allocation of components and connections

(b) cycle-accurate scheduling

(c) Binding of variables, operations and transfers

(d) Synthesis of controller

(e) Model refinement

(a-c) kan udføres sammen eller i hvilken som helst rækkefølge men de er afhængige af hinanden. Hvis de udføres samtidigt bliver synthesisen kompleks og utilregnelig. en strategi er at udføre det i følgende rækkefølge: a c b

Alle trinene ovenover kan udføres automatisk eller manuelt. Hvis det hele bliver udført automatisk kaldes det processor-level synthesis eller high-level synthesis (HLS). Hvis a-d bliver udført manuelt og e automatisk kaldes det model refinement.

System-level behavioral model / system synthesis: side 35

(a) Profiling and estimation.

(b) Component and connection allocation

(c) Process and channel binding

(d) Process scheduling

(e) IF component insertion

(f) Model refinement

1.3 System Design methodology: side 39

model algebra side 42

algebra:<objects, operations>

$$a * (b + c) = a * b + a * c$$

dette gør det muligt for system designere at optimere designet vha. arithmetic algebra regler. Udtrykket til venstre for = kræver en multiplifier og en adder hvorimod udtrykket på højre side kræver 2 multipliers og en adder.

modelalgebra : <objects, compositions>

1.4 System-level models: side 44

applications designers, system designers og implementations designers.

2 system design methodologies side 56

Dette kapitel beskriver forskellige system design methodologies.

Bottom-up: start i bunden (circuit level) køber fra behavior til physical og lav et library. brug dette library til næste level.

Top-down: Start i øverste level (system level) køber fra behavior over til structure. gå et level ned og gentag. ved nederste level gå fra behavior til physical.

Platform methodology: side 61

FPGA methodology: side 64

er based på FPGA substrat som består af multitudes af 4-bits ROM celler (Look-up Tables (LUTs)) og disse LUTs kan implementere enhver 4-variable boolsk funktion. I denne design methodology vil hver RTL komponent i biblioteket være opbygget

af disse 4-variable funktioner. Og derefter bliver processor delene synthesiseret af disse RTL componenter.

sagt på en anden måde: denne metodologi bruger en top-down approach på både system og processor levels hvor standard og custom Processing Elements og Communication Elements er opbygget af LUTs. Et system design starter ved at man mapper en applikation ned på en given platform og derefter syntetiserer brugerdefineret komponenter ned til RTL komponenter som er defineret i form af LUTs. Standard processor komponenter i processor biblioteket er allerede defineret i form af LUTs. Når alle vores komponenter er defineret via LUTs så simplificere vi designet ned til LUTs og BRAM og derefter kan vi udføre placering og routing med værktøjer FPGA producenten har udviklet.

Denne form for top-down metode har de samme svagheder som andre top-down metoder som er at det kan være svært at optimere hele designet ved at simplificere designet ned til basiske LUT celler. Derudover ved udvikler heller ikke om hvordan FPGA producentens værktøj placerer og forbinder LUTs og BRAMs.

Chapter 7

Architecture design

This chapter will describe the design process for the hardware architecture. In section ?? a parallelism analysis is performed. Section ?? will describe the allocation and scheduling of the hardware elements.

NOT DONE! rough sketch. De nedenstående trin er hvad jeg skal igennem: Para. Anal., Alloc., Optimizaiton, FSM D og VHDL + simulering

7.1 Parallelism Analysis

The inherent parallelism of the system has been analyzed to find out which improvements can be made. Figure xx shows a diagram of the whole system and on the following pages, data flow graphs (DFG) for each subsystem can be found.

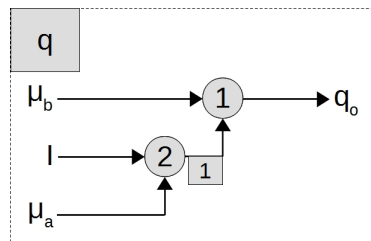


Figure 7.1: Synchronous data flow graph of the q block in the guided image filter

To find the parallelism in the system precedence graphs (PG) have to be created. Since the subsystems are simple the precedence graphs can be created looking at the system from end to start and for each operator find out which signal is needed and have to be calculated before. Another method is to create a synchronous data flow graph (SDFG) and from the SDFG a matrix, $\underline{\Gamma}$, can be created. This matrix expresses the relationship between the in- and outputs from each node in the SDFG and is called the *topology* matrix. An example of an SDFG is illustrated on in figure

7.1. In this example the topology matrix is then:

$$\underline{\underline{\Gamma}} = \begin{matrix} & \text{nodes} \\ \text{arcs} & \begin{bmatrix} -1 & 1 \end{bmatrix} \end{matrix} \quad (7.1)$$

If $\text{rank}(\underline{\underline{\Gamma}}) \leq s - 1$ where s is the number of nodes then a positive integer vector \underline{q} can be found such that $\underline{\underline{\Gamma}} \cdot \underline{q} = \underline{0}$. Then resulting \underline{q} is:

$$\underline{q} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad (7.2)$$

This \underline{q} vector expresses how many times each node have to be executed within one sample period.

With $\underline{\underline{\Gamma}}$ and \underline{q} a periodic admissible sequential sequence (PASS) can be found. This tells a sequential sequence in which the nodes can be executed and with it, a periodic admissible parallel sequence (PAPS) can be generated. To find the PASS generate a randomly ordered list of all the nodes, L . For each

7.2 Allocation and Scheduling

To develop a Finite State Machine (FSM) we need to know which hardware is allocated and when each operation is scheduled. This enables us to define some states for the FSM.

From section 7.1 on page 25 some precedence graphs are found and theses graphs shows how many FUs can run in parallel. We need to know how much hardware is needed for each FU. To calculate some simple system have been generated in Vivado 2016.2 and then synthesized. Then utilization of the hardware can be found. Appendix A on page 39 describes in detail the procedure and table ?? shows the result.

	LUT	FF	BRAM	DSP48
Adder	15	15	-	-
Adder/Subtractor	≈ 16	15	-	-
Subtract	15	15	-	-
Multiplier - LUT	352	36	-	-
Multiplier - DSP48	-	-	-	1
Divider	≈ 177	82	0.5	7

Table 7.1: Number of logic elements used in average for each FU

To create the FSM specification we need to introduce time into the precedence graphs to establish some states. To achieve this scheduling is used. There exists different methods for scheduling and some of these are:

- Resource Constrained (RC)
- Time Constrained (TC)

7.2.1 RC scheduling

For both methods the first step is create as soon as possible (ASAP) and as late as possible (ALAP) schedules. Then for the RC scheduling a ready list is created. This list contains a list of operations which are ready for scheduling and sorted by mobility. The mobility, $M(op)$, expresses the difference between the states in which the operation, op , have been scheduled in ASAP and ALAP schedules, e.i. $S_{ALAP}(op) - S_{ASAP}(op)$.

Figure 7.2 on page 27 shows an example of an ASAP and an ALAP schedule. As seen on the figure node 1-10, 13, 17, 20 and 22-23 are part of the critical path and therefore mobility is 0 for each of the nodes. Nodes 11-12,14-15,18 and 21 all have a mobility of 2 and nodes 16 and 19 have a mobility of 3.

lav om til operation og
husk mult

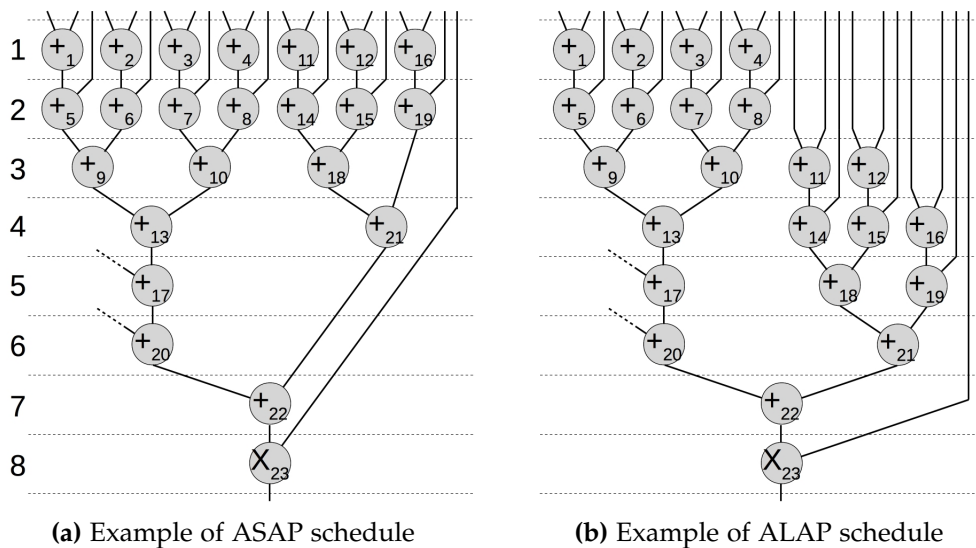


Figure 7.2: Illustration of ASAP and ALAP schedules. The illustration shows a part of the mean filter shown on figure ??

With mobility found for each node/operation then a ready list can be generated. Add every ready node and then sort them by mobility. In the start the list would look like this: 1. node 1 ($M(+_1) = 0$), 2. node 2 ($M(+_2) = 0$), 3. node 3

($M(+_3) = 0$), 4. node 4 ($M(+_4) = 0$), 5. node 11 ($M(+_{11}) = 2$), 6. node 12 ($M(+_{12}) = 2$) and 7. node 16 ($M(+_{16}) = 3$).

Lets say the system have 3 ALU's available then node 1-3 can be scheduled to state 1 since they are higher on the ready list and there is no free ALU for the remaining nodes. The scheduled nodes are removed and the ready list is updated with newly available nodes. The list is still sorted by mobility and will now look like this: 1. node 4 ($M(+_4) = 0$), 2. node 5 ($M(+_5) = 0$), 3. node 6 ($M(+_6) = 0$), 4. node 7 ($M(+_7) = 0$), 5. node 11 ($M(+_{11}) = 2$), 6. node 12 ($M(+_{12}) = 2$) and 7. node 16 ($M(+_{16}) = 3$).

Then nodes 4-6 are scheduled into state 2 since they have lower mobility than the rest of the ready list. The list is updated again and will look like this: 1. node 7 ($M(+_7) = 0$), 2. node 8 ($M(+_8) = 0$), 3. node 9 ($M(+_9) = 0$), 4. node 11 ($M(+_{11}) = 2$), 5. node 12 ($M(+_{12}) = 2$) and 6. node 16 ($M(+_{16}) = 3$).

Nodes 7-9 are scheduled into state 3 and the list is updated again: 1. node 10 ($M(+_{10}) = 0$), 2. node 11 ($M(+_{11}) = 2$), 3. node 12 ($M(+_{12}) = 2$) and 4. node 16 ($M(+_{16}) = 3$).

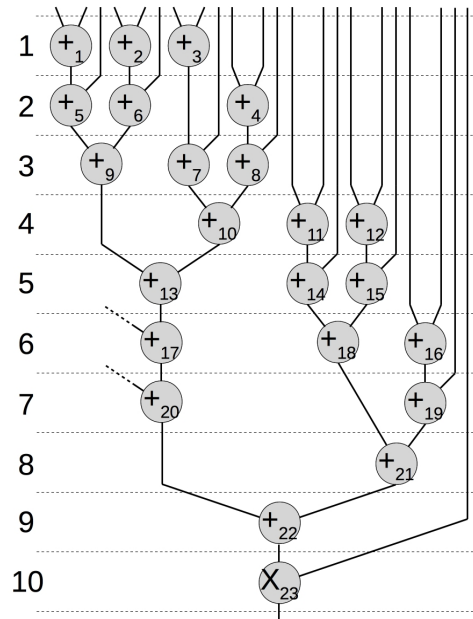


Figure 7.3: RC schedule

This is repeated until all nodes have been scheduled and the result is seen in figure 7.3 on the preceding page. This schedule uses 2 states more than the ASAP

and ALAP but it reduces the cost since it uses 1 less ALU than the ALAP schedule and 5 less than the ASAP schedule.

This information is described in [gajski2009].

7.2.2 TC schedule

The RC schedule improves the cost of the implementation while the TC intends to improve the performance of the implementation. First a maximum number of states is decided and then ASAP and ALAP schedules are generated and mobility ranges are found. Then some probabilities are assigned to each operation. These probabilities express the probability for the specified operation to be scheduled in each state in its mobility range e.g. operation 11 in 7.2 on page 27 have 1/3 probability for being scheduled in each of state 1-3 if maximum number of states is kept at 8. Figure 7.4 on page 29 shows the probabilities for each operation in figure 7.2 on page 27. Fortsæt med at skrive senere

find et bedre ord

									ALU	MULT
1	$+_1$	$+_2$	$+_3$	$+_4$					$1 \times 4^{+1/3} \times 2^{+1/4} = 4^{11/12}$	0
2	$+_5$	$+_6$	$+_7$	$+_8$	$+_{11}$	$+_{12}$	$+_{16}$	$+_{19}$	$1 \times 4^{+1/3} \times 4^{+1/4} \times 2 = 5^{10/12}$	0
3		$+_9$		$+_{10}$		$+_{14}$	$+_{15}$	$+_{21}$	$1 \times 2^{+1/3} \times 5^{+1/4} \times 2 = 4^{2/12}$	0
4			$+_{13}$			$+_{18}$	$+_{20}$		$1^{+1/3} \times 4^{+1/4} \times 2 = 2^{10/12}$	0
5			$+_{17}$				$+_{22}$		$1^{+1/3} \times 2^{+1/4} \times 1 = 1^{11/12}$	0
6			$+_{20}$						$1^{+1/3} = 1^{1/3}$	0
7					$+_{22}$				1	0
8					X_{23}				0	1

Figure 7.4: Probabilities for each operation in figure 7.2 on page 27

7.3 FSM design

noter til mig selv

Controller <-signaler-> Data path

Controller:

Data path: indeholder register, FU, connections, network osv.

skal jeg lave et eksempel med en lille del af mit system? så kan man se hvor voldsomt det kan blive og se processen. Derefter kan jeg vise hvad jeg kommer frem til

noter fra møde med peter:

- lifetime analysis
- synchronous diagram

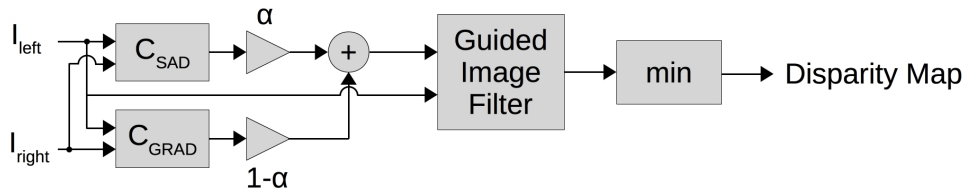
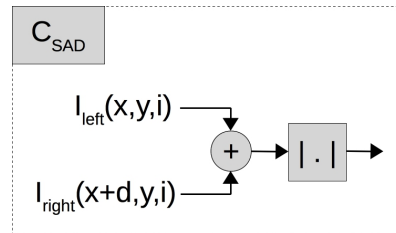
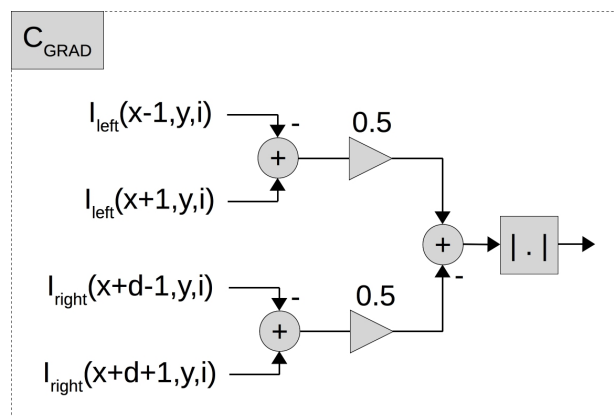


Figure 7.5: Data flow graph of the whole system

Figure 7.6: Data flow graph of the C_{SAD} block in figure 7.5Figure 7.7: C_{GRAD} ikke sikker

noter til mig selv

7.4 VHDL + Simulation

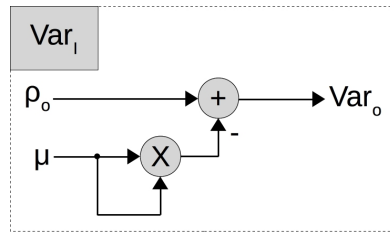


Figure 7.11: Variance

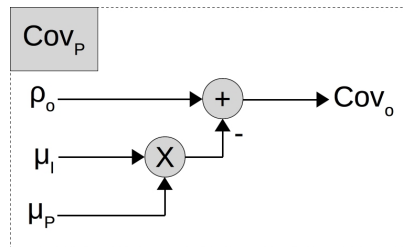


Figure 7.12: Covariance

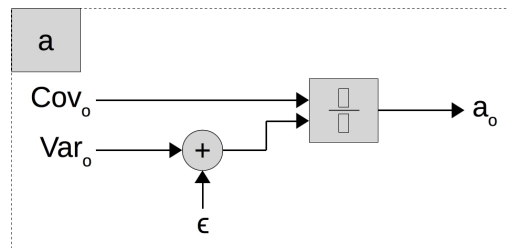


Figure 7.13: A box

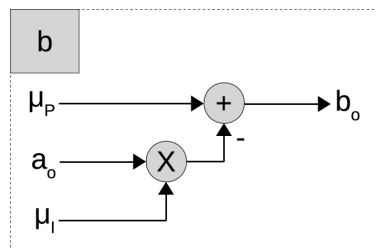


Figure 7.14: B box

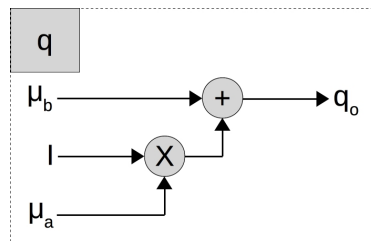


Figure 7.15: Q box

Chapter 8

Acceptance test

udfør accept test ud fra test specifikationen (brug data fra python simulering og giv det til VHDL implementationen)

Chapter 9

Conclusion

This chapter will contain the conclusion

Appendix A

Allocation test

This appendix will explain how the average allocation of logic elements for each functional unit is found.

The result is used in table ?? in chapter 7. The table is repeated here in table A.1 on page 39

	LUT	FF	BRAM	DSP48
Adder	15	15	-	-
Adder/Subtractor	≈ 16	15	-	-
Subtract	15	15	-	-
Multiplier - LUT	352	36	-	-
Multiplier - DSP48	-	-	-	1
Divider	≈ 177	82	0.5	7

Table A.1: Number of logic elements used in average for each FU

A.1 General procedure

This section will describe the general procedure to find the utilization of logic elements for the specified FUs.

First a new project is created in Xilinx Vivado 2016.2. All the settings are set to standard except for the target which is set to ZedBoard Zynq Evaluation and Development Kit. Then a block design is created and a single IP of the wanted type is added. Then for each input and output a port is generated. The inputs are connected to the corresponding ports and between the output of the IP and the corresponding port a Xilinx Slice IP core is inserted which strips every bits except one (MSB) from the output and this IP is connected to the output and the port. This is done to not use to many I/O ports which will make . Using TCL

commands the IP block is copied 99 times and for each copy a new slice and output port is generated and connected to the copy. The inputs are all connected the original corresponding ports except for the divider FU where each copy will have its own input ports. When everything have been connected then Vivado is set to generate top-level HDL wrapper for the block design. Run synthesis and implementation and after completion check the utilization table under the project summary. Note these values, go to the block design and change one of the slices to full output width. Then run synthesis and implementation again and notices if there is a difference in LUT utilization. The LUT utilization is this value minus the LUT value from the former run + 1. These values should be divided by 100 and inserted into table A.1 on page 39.

The following sections will describe the specific settings for each FUs.

A.2 Adder

This FU uses the Xilinx Adder/Subtractor v12.0 LogiCORE IP. The IP is set to implement using *Fabric*. The inputs are set to a width of 15 bits, mode is set to *add* and the output is set to 15 bits and latency is set to 1. All control signals are disabled.

A.3 Subtract

This FU uses the Xilinx Adder/Subtractor v12.0 LogiCORE IP. The IP is set to implement using *Fabric*. The inputs are set to a width of 15 bits, mode is set to *subtract* and the output is set to 15 bits and latency is set to 1. All control signals are disabled.

A.4 Adder/Subtract

This FU uses the Xilinx Adder/Subtractor v12.0 LogiCORE IP. The IP is set to implement using *Fabric*. The inputs are set to a width of 15 bits, mode is set to *add subtract* and the output is set to 15 bits and latency is set to 1. All control signals beside the ADD signal are disabled.

A.5 Multiplier - LUT

This FU uses the Xilinx Multiplier v12.0 LogiCORE IP. The IP is set to construct the multiplier using *LUTs* and is set to optimize for speed. The inputs are set to a width of 18 bits and the output is set to 36 bits and pipeline stages is set to 1. All control signals are disabled.

A.6 Multiplier - DSP48

This FU uses the Xilinx Multiplier v12.0 LogiCORE IP. The IP is set to construct the multiplier using *Mults* and is set to optimize for speed. The inputs are set to a width of 18 bits and the output is set to 36 bits and pipeline stages is set to 1. All control signals are disabled.

A.7 Divider

This FU uses the Xilinx Divider Generator v5.1 LogiCORE IP. The algorithm type is set to *High Radix*. The inputs are set to a width of 8 bits and the output fractional width is set to 8 bits and latency is set to 3. All control signals are disabled. With this only 5 instances of the Divider is used since each instance uses a lot of IO and to ensure the ability to run implementation without error.

Appendix B

Extra Figures

noter til mig selv

This chapter will contain extra which might fill to much in the report. Looking at you precedence graphs. Maybe make it as fold out image as we did with large diagrams in earlier reports.