Collision detection and avoidance system based on computer vision

Andreas Kalvå

Master of Science in Cybernetics and Robotics
Submission date: February 2014
Supervisor: Amund Skavhaug, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

REPORT

Collision detection system using computer vision on low power devices

by

Andreas Kalvå

February 21, 2014

Project description

There is a need for computer vision and especially stereo vision for perceiving depth with regard to collision avoidance in a wide range of scenarios. This need is amplified by the current trend of wanting more autonomy in new robot systems.

At the Department of Engineering Cybernetics at the Norwegian University of Science and Technology, we wish to develop a module for this, based on open source libraries, where hardware is interchangeable and with a certain emphasis on the limitations present both in computational- and energy resources.

The objectives are:

1. Familiarize with the necessary literature.
2. Familiarize with previous work done at the institute (and if practically possible other places as well.)
3. Evaluate existing methods and libraries.
4. Evaluate possible hardware platforms.
5. If possible suggest a complete and generic architecture.
6. If time allows implement said architecture for a given hardware platform, usage area etc.
7. Evaluate the viability of the implementation.

Submission date: February 2014

Supervisor: Amund Skavhaug
Department of Engineering Cybernetics.

Summary

A wide selection of stereo matching algorithms have been evaluated for the purpose of creating a collision avoidance module. Varying greatly in the accuracy, a few of the algorithms were fast enough for further use.

Two computer vision libraries, OpenCV and MRF, were evaluated for their implementations of various stereo matching algorithms. In addition OpenCV provides a wide variety of functions for creating sophisticated computer vision programs and were evaluated on this basis as well.

A stereo camera were constructed using low cost, of-the-shelf web cameras.

Two low-power platforms, The Pandaboard and the Beaglebone Black, were evaluated as viable platforms for developing a computer vision module on top. In addition they were compared to an Intel platform as a reference.

Based on the results gathered, a fast, but simple, collision detector could be made using the simple block matching algorithm found in OpenCV. A more advanced detector could be built using semi-global stereo matching. These were the only implementations that were fast enough. The other energy minimization algorithms (Graph cuts and belief propagation) did produce good disparity maps, but were too slow for any realistic collision detector.

In order for the low-power platforms to be fast enough, a combination of improvements must be used. OpenCV should be compiled with aggressive optimization options enabled with support for hardware accelerated floating point math. Choice of low-power platform matters, but it is possible to work around this by reducing the workload.

The most effective speedup that enables the low-power platforms were reducing the resolution of the images to be matched. When reducing the size of the sub-problems enough to align with cache size, considerable speedups were found with little penalty in the corresponding disparity map.

Sammendrag

Et større utvalg av algoritmer for stereosyn har blitt vurdert med det formål å lage en kollisjons- deteksjon og unngåelsesmodul. Med varierende grad av nøyaktighet har et fåtall av algoritmene som var raske nok blitt valgt for videre bruk.

To programmeringsbibliotek for datasyn, OpenCV og MRF, har blitt vurdert for sine stereosynsimplementasjoner. OpenCV har i tillegg blitt vurdert ut i fra hvilke funksjoner og datastrukturer den tilbyr med tanke på å konstruere denne kollisjons- deteksjon og unngåelsesmodulen.

En stereokamera ble bygget ved hjelp av et par billige webkameraer.

To laveffekts plattformer, The Pandaboard og BeagleBone Black, ble vurdert etter hvilken evne de har som en underliggende platform for datasyn. Som referanse ble en Intel-basert arbeidsstasjon brukt som referanse.

Ut i fra resultatene som ble funnet, kan en rask, men i overkant enkel, kollisjonsdetektor kan lages ved å bruke OpenCVs «Block Match» -algoritme. En mer avansert kollisjonsdetektor kan lages ved å bruke OpenCVs «Semi-Global Block Match» -algoritme.

Det var kun de her to algoritmene som kunne kjøre raskt nok på laveffekts platformene. Andre algoritmer som ble testet som «Graph Cut» og «Belief Propagation» ga nøyaktige resultat, men de brukte for lang tid til at det var realistisk å bruke dem til kollisjonsdeteksjonsformål.

For at laveffektsplattformene skulle være raske nok, ble en kombinasjon av forbedringer tatt i bruk. OpenCV ble kompilert med maskinvareakselerert flyttalls-støtte og automatisk vektorisering skrudd på. Det anbefales å velge en kjapp laveffektsplatform, men det lar seg gjøre å kompensere ved å redusere arbeidsmengden.

Den mest effektive hastighetsøkningen som i det hele tatt gjorde det mulig å bruke laveffektsplattformene til dette formålet, var å redusere oppløsningen på bildene som skulle sammenlignes. Ved å redusere størrelsen, oppnådde man en stiuasjon der delproblemer av stereosyns-algoritmene samsvarte tilstrekkelig bra med prosessorenes «cache» -størrelse uten at det ble redusert noe særlig i kvaliteten på dybdekartene.

Preface

This master thesis were done during the fall of 2013 to mid winter in 2014 at the Department of Engineering Cybernetics at the Norwegian University of Science and Technology.

The purpose of this thesis was to gain a better insight into the field of computer vision and to create a module that there has been requested a need for.

The project work was supervised by associate professor Amund Skavhaug.

I would like to that my supervisor for being a source of inspiration during this thesis.

February 2014, Trondheim Andreas Kalvå

Contents

List of Figures

Unless otherwise stated, each figure is the work of the author.

List of Tables

1 Introduction

Robots are on the rise. We create robots to replace an ever increasing amount of labor, but they are still very dependent on our programming to act in this world. Simple procedure is not enough if we are to bring robots further into our world.

There is a need for improved autonomy in the robots we create. We want them to be able to operate safely in an environment with other agents. Both other robots, but also humans eventually. To do so an understanding of the world around is needed.

We humans have largely solved this problem through evolutionary means. We have eyesight that let us interpret our surrounding environment and act within it. If we could mimic our visual system in our robots it would be a great step towards full autonomy.

So far we are nowhere close to such a technical implementation, but the field of computer vision has made some progress.

At the same time, low power computers with significant calculation power has become prevalent. Therefore we want to extend upon that and evaluate how future technology can become ubiquitous.

This thesis extend upon the work done previously in computer vision at the Norwegian University of Science and Technology [End10] and [Rø13].

This thesis is divided into four parts. In the first part some theory and methods for creating computer vision. Then in the second part libraries and platforms will be introduced. In the third part the results will be reviewed and the last part will be discussion and conclusion about the work done.

2 Theory

The theory section is divided into three. The first section covers computer vision. The second ection covers image sources and the last section covers feature extraction.

2.1 Computer Vision

Computer vision is the area of research which covers all parts of enabling a computer too see. In this thesis there is a need to sense the external world so computer vision methods will be evaluated.

2.1.1 Stereo Vision

The central problem that we are trying to solve in this thesis is enabling a robot to perceive the world and increase it's autonomy. Although there are many proposed systems for perceiving the world, we choose to look at nature for strategies to emulate. A set of two eyes are often used to form a visual system that perceives the world in a certain direction and objects are correlated by the visual cortex to form a perceived depth to the world. Clearly the success of this evolutionary marvel suggests it is a fitting solution to our problem. Stereo vision is currently a very active area of research.

2.1.2 Epipolar Geometry

Correlating points in two images is difficult as potentially every pixel in one image could be the corresponding pixel to some point in the other image. A 2D search for every pixel would be very resource consuming and highly impractical. Instead we use epipolar geometry. Two image sources of the same scene from distinct positions, like in figure[1] 2, will have a number of geometric relations between the 3D-points and their 2D-projections that lead to constraints

[1]Created by Arne Nordmann, CC-BY-SA-3.0

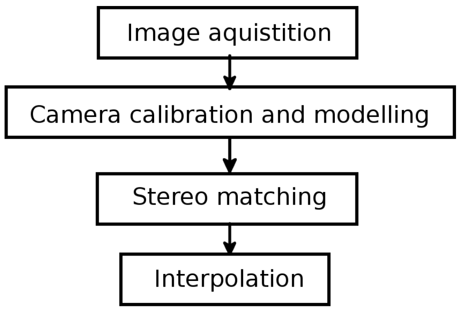


Figure 1: Pipeline

between the image points.

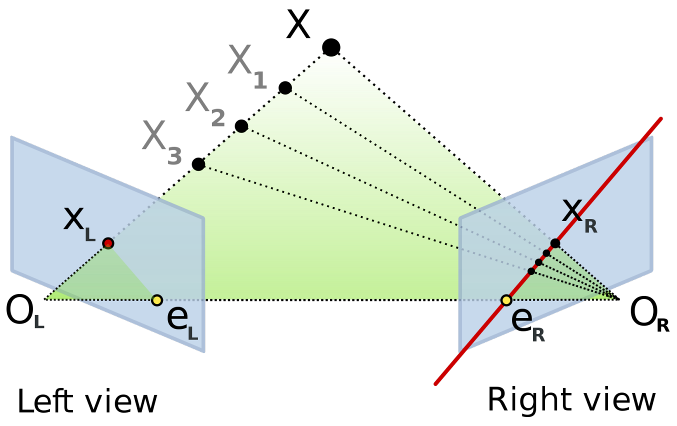


Figure 2: Epipolar views

Epipole

Since the projections are distinct, each projection will project it's center intro the other projection. We denote these projections e_l and e_r.

Epipolar line

The line O_l-X is seen by the left camera as a point while in the right camera it is a line. This is called an epipolar line. Likewise O_r-X is seen as a point in the right camera.

Epipolar constraints

If the relative translation and rotation of the cameras are known, then two important observations can be made. First that each point in either camera must lie on an epipolar line in the other camera. Thus reducing the search down to 1 dimension. Second that if two points are found to correspond, the distance can be found by triangulation.

Special case

If the two camera image planes coincide, then the search is simplified further because the epipolar lines will be horizontal. If the cameras are not placed in such a way, then the projections can be transformed to form a common image plane using image rectification.

2.1.3 Occlusion

Problem 1 - Order of regions may be different in each camera.

Why simply matching edges detected by a canny filter fails. Feature extraction using Canny filter (edge) and some color segmentation can be used to identify regions which in turn can be matched using a region based strategy when generating the disparity map.

Problem 2 - There may be regions in camera 1 that has no match in camera 2.

Because we are trying to extract depth information from a 2 dimensional projection of a 3 dimensional world...

The essential part of the second problem is that near objects will occlude different regions in the background in each camera. While the near regions are then easier to identify, there are situations where there are regions in one camera that has no match in the other camera.

In the more complex situation where there are more than two regions bordering to each other. This easily becomes guesswork since there does not exist enough information reliably match...

The fast, but naive block matching algorithms completely fails in this regard as it is dependent on using SAD-windows of a certain size and simply assigns the matched block to the same depth.

2.1.4 Disparity map

The disparity map is a picture produced by correlating two pictures of the same scene. Each point in the picture represents the distance between a point in picture A and the corresponding point in Picture B.

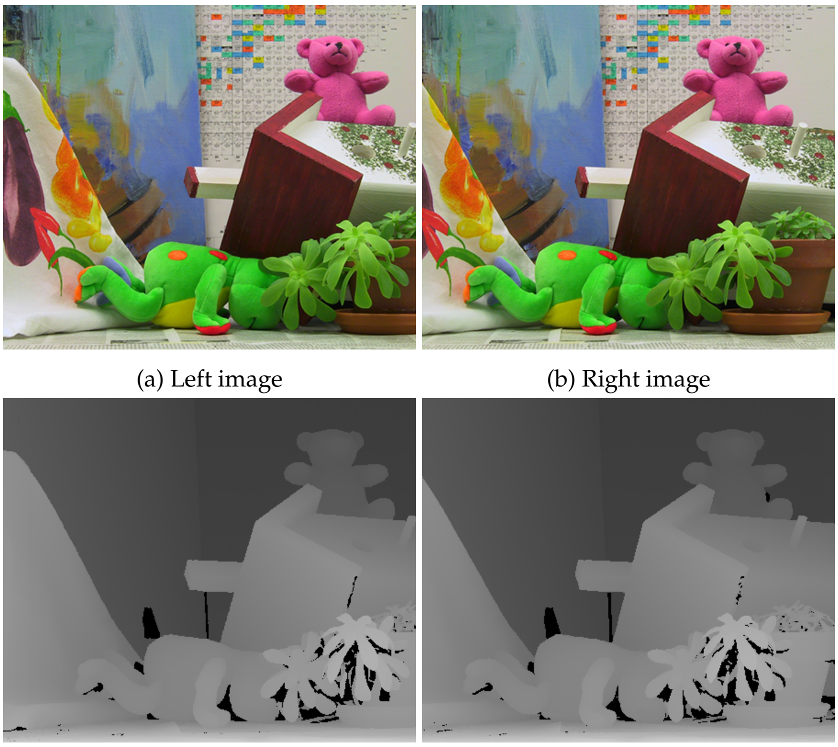
This is best explained with an example. Depicted in figure 3 is the left, fig. 3a, and right, fig. 3b, from the Teddy data set[SS03] made available on the Middlebury Vision web page. For both the left and the right image there is an accompanying disparity map for that picture, figure 3c and 3d.

The disparity map is intuitive in that close objects are brighter than objects further back. Corresponding point that have no distance between then are said to be placed at infinity and is given the disparity level of 0. If the distance is 1 apart then one way of creating the disparity map is to give the point a disparity level of 1 and so on. This creates a continuous disparity map.

Some implementations use fewer disparity levels and scale the resulting disparity map instead. For example with 16 discrete levels, the values assigned is 0 to 15. By scaling with a factor of 17, the maximum level becomes $15 \times 17 = 255 = 2^8 - 1$ which is the maximum brightness that can be represented in an 8-bit gray level image. By scaling this way it becomes easier to visually inspect the disparity map, but some information of scaling and what the disparity levels represent is needed.

In this thesis methods for producing a disparity map will be evaluated. These methods are referred to as stereo matching algorithms.

Another method that is commonly used is structured light[FARW99]. Structured light is a procedure where a known pattern of light is projected onto a scene and from the way the pattern is distorted by the scene, the disparity level



(a) Left image (b) Right image

(c) Left disparity map (d) Right disparity map

Figure 3: Teddy data set. Image pair and disparity maps.

of a known point can be found by measuring the distance to the known point in two cameras.

This is the method Kinect uses. It projects many infra red points onto a scene and matches these point when seen from two cameras.

This method is known to produce good result, but because it is projecting light is does not work very well outside in sunny weather. The disparity maps from the Teddy set in figure 3 is constructed using structured light.

2.1.5 Converting from disparity to distance

Finding the distance using a disparity map requires some information about the distance between the cameras and the cameras angle of view.

In figure 4 there are two cameras that point at the point S. Since there is some distance between camera 1 and camera 2, an angle s is formed. By setting the point S at infinity, the angle efficiently becomes 0 and can be disregarded.

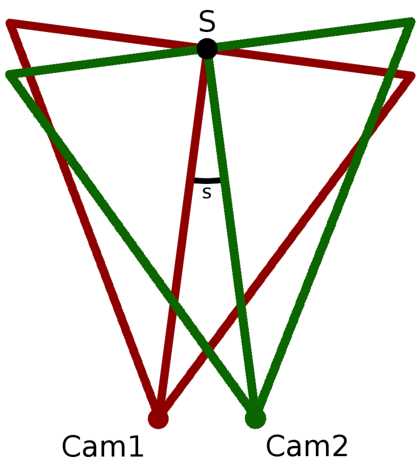


Figure 4: Middle point S at infinity.

With both cameras pointed at the point S, the distance to the point X can be found using the disparity d. In figure 5, the point X and the two cameras for a triangle ABC.

The distance b from camera 1 to the point X is found using the sine rule.

$$\frac{a}{sinA} = \frac{b}{sinB} = \frac{c}{sinC} \tag{1}$$

Where:

$$\angle C = \pi - \angle A - \angle B \tag{2}$$

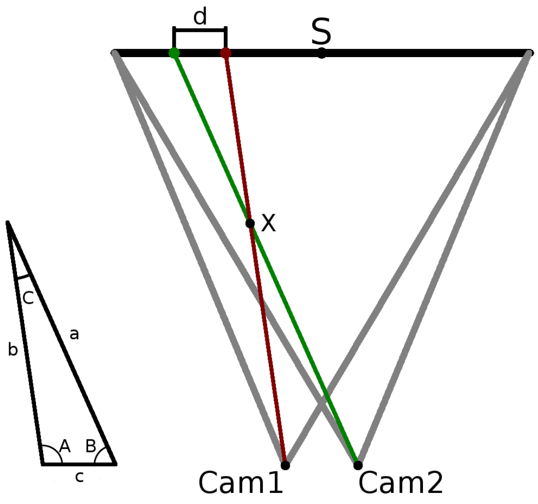


Figure 5: The point X forms a triangle.

To find the angles A and B, the resolution of the camera and the angle of view. As an example the resolution and the angle of view will be the same as the cameras used later in this thesis. The resolution is 320x240 and the angle of view is 0.873 rad from side to side. The change in angle per pixel is found by $0.873/320 = 0.002727$

Using geometric distance as in figure 6:

The angle A is then found:

$$\angle A = \frac{0.873}{320}\sqrt{a_y^2 + a_x^2} \tag{3}$$

And the angle B:

$$\angle B = \frac{0.873}{320}\sqrt{b_y^2 + b_x^2} \tag{4}$$

Because of the epipolar constraint:

$$b_y = a_y \tag{5}$$

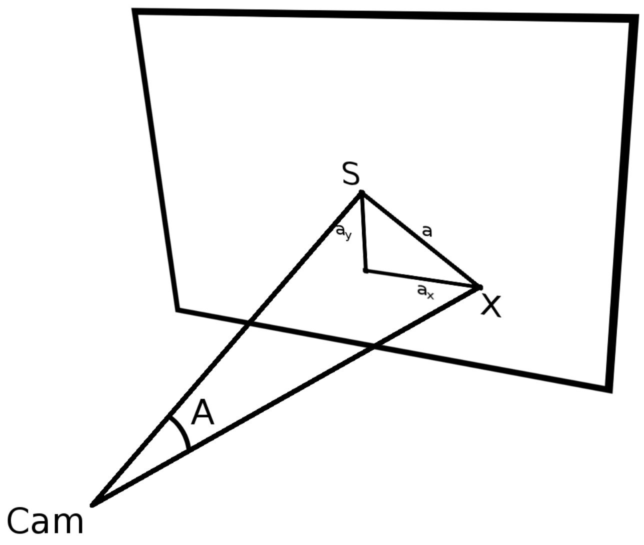


Figure 6: Finding the angle A

While the disparity is used to shift a_x:

$$b_x = a_x - d_{a_x} \tag{6}$$

In figure 7 the distance to the center of camera 1 is plotted for all possible disparity values:

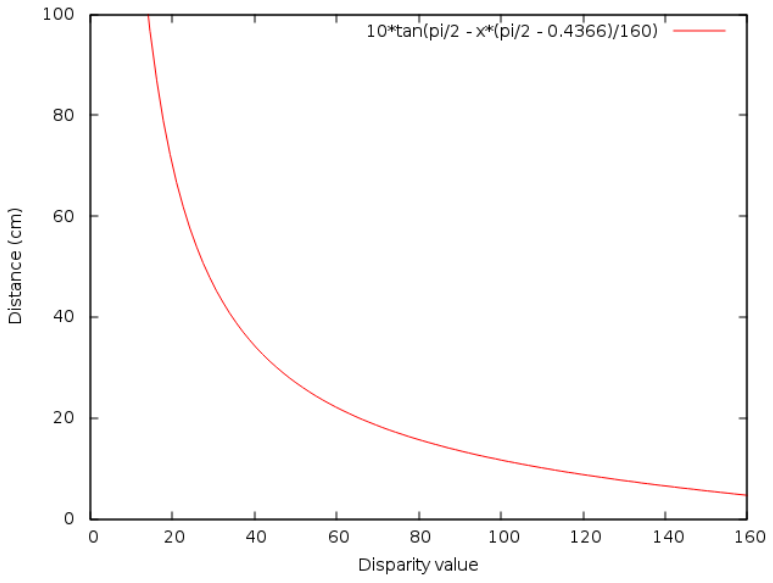


Figure 7: Distance to the center of the camera from disparity value

2.2 Image Source

2.2.1 Non-linear defects

Theoretically, it is possible to construct a camera with no distortions. The problem is that to do so very a very strict quality control and production process would be necessary and lenses would be far more expensive. For most applications such perfection is unnecessary. Instead lenses will have some imperfections. Among the most common forms of distortions, and the one that matter in this regard, is the imperfections in the parabolic structure of the lens. These distortions are called radial distortions because of the symmetrical nature of the lens. Radial distortions are characterized by that pixels in the center of the distortion appear at a different distance than the pixels further out. The two simplest form of radial distortion are called barrel distortion where the edges appear further away and pincushion distortion where the edges appear nearer than they actually are.

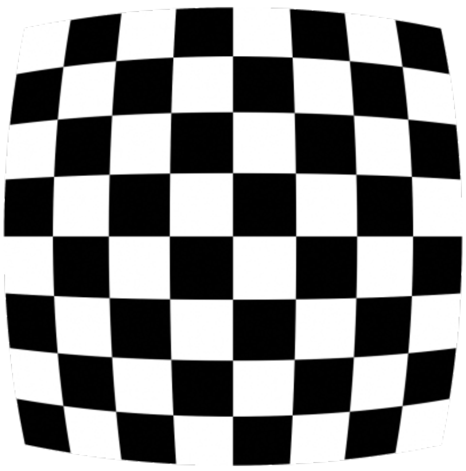


Figure 8: Barrel distortion

A combination of these imperfections are also possible. For example the mustache distortion:

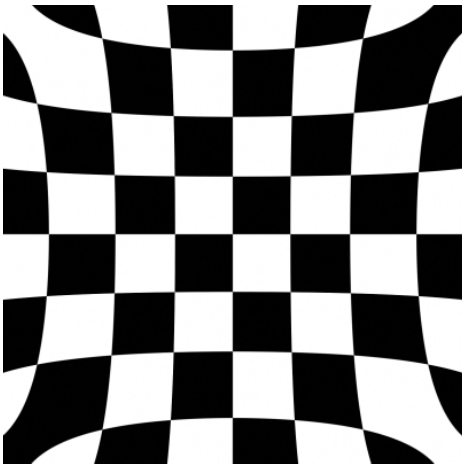


Figure 9: Pincushion distortion

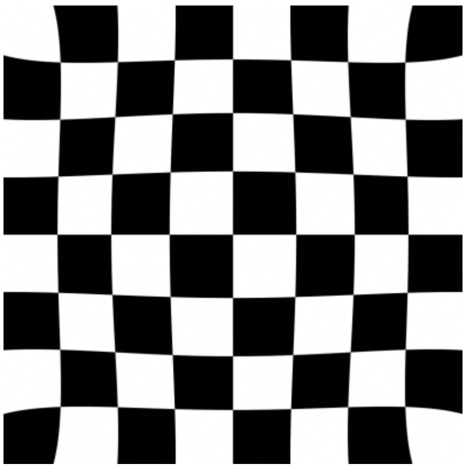


Figure 10: Mustache distortion pattern

Mathematically barrel and pincushion distortions are quadratic meaning that they increase as the square of the distance from the center. Mustache is quartic (4th order). Usually these distortions are found in zoom lenses, but can also be found in prime lenses. The lens′ characteristics should be explored to see if it is necessary to correct for radial distortion.

Tangential distortion appears as skewed image and is caused by the physical elements in the lens not being perfectly aligned.

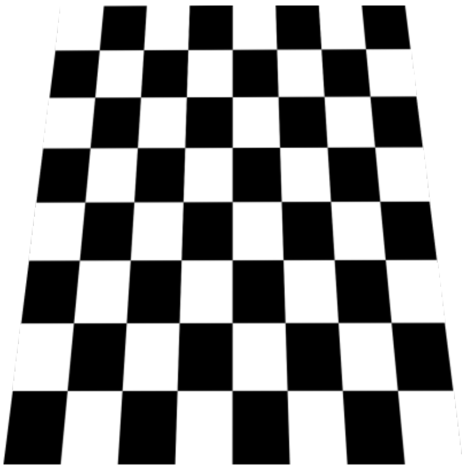


Figure 11: Tangential distortion

Software correction. Use brown′s distortion model to reverse distortion. Needs to know the lens′ characteristic for this. The following describes an alternative method.

2.2.2 Undistort/rectification of camera

Use Brown distortion model for undistortion.

Radial distortions:

$$x_u = (x_d - x_c)(1 + K_1 r^2 + K_2 r^4 + \dots) \tag{7}$$

$$y_u = (y_d - y_c)(1 + K_1 r^2 + K_2 r^4 + \dots) \tag{8}$$

Tangential distortions:

$$x_u = (P_1(r^2 + 2(x_d - x_c)^2) + 2P_2(x_d - x_c)(y_d - y_c))(1 + P_3r^2 + P_4r^4 + \ldots) \quad (9)$$

$$y_u = (P_1(r^2 + 2(y_d - y_c)^2) + 2P_2(x_d - x_c)(y_d - y_c))(1 + P_3r^2 + P_4r^4 + \ldots) \quad (10)$$

2.3 Feature based computer vision

2.3.1 Edge detection

An edge in computer vision is a transition between two surfaces meaning a shift in intensity and/or color. Edge detection is the process of identifying where these edges are and often marking them as an edge. Although there exists methods that finds edges and gives a result on how strong the edge is, it is more common to use some threshold to mark a pixel either as an edge or not. Edge detection is usually used as an intermediate step in feature-based stereo vision.

One of the most popular edge detectors is the Canny edge detector[Can86]. Likely so because of of its efficiency and good results. Canny's aim was to design an optimal edge detector according to a few criteria:

- Low error rate - Low amounts of false edges.
- Good localization - Edges should be close to the actual edge.
- Minimal response - There should only be one edge response.

Edge detection filters are often used as an intermediate in more complex algorithms where edges provides some information needed for the success of said algorithm.

The steps of the Canny edge detector is as follows:

The Canny edge detector is susceptible to noise so the first thing we do is to smooth the surface using Gaussian smoothing. For example a Gaussian filter of 5x5 such as (11) gives a good result.

$$K = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \tag{11}$$

Then we find the intensity gradient of the surface by applying convolution masks in x and y direction (12):

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{12a}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{12b}$$

And gradient strength and direction are found using:

$$G = \sqrt{G_x^2 + G_y^2} \tag{13}$$

$$\theta = \arctan \frac{G_y}{G_x} \tag{14}$$

The direction is rounded off to either 0, 45, 90 or 135 degrees.

Since we only want a single edge response we use non-maximum suppression to remove pixels that we don't consider an edge and we are left with thin lines.

Finally we compare each pixel to two thresholds, upper and lower. Ia a pixel is above the upper threshold it is accepted as an edge. If it is under the lower threshold it is discarded. If it is between the upper and lower threshold it is accepted only if it is connected to a pixel above the upper threshold.

Canny recommends an upper:lower threshold between 2:1 and 3:1.

3 Stereo Matching

An introduction to the various algorithms used in this thesis.

3.1 Region Based

3.1.1 Block Matching

Matching pixel for pixel usually gives more than one match and therefore does not give a good result. Block matching can roughly be explained as matching blocks of pixels in order to reduce the amount of matches. This usually works better, but regions of pixels that are larger than the block where there is little contrast will typically give more than one match.

Methods used for determining the similarity between two blocks of pixels:

SAD - Sum of Absolute Differences

$$SAD = |(A_1 - B_1)| + |(A_2 - B_2)| + \ldots + |(A_n - B_n)| \tag{15}$$

Modern processors

MSE - Mean Squared Error

$$E = \frac{(A_1 - B_1) + (A_2 - B_2) + \ldots + (A_n - B_n)}{n} \tag{16}$$

$$MSE = E^2 \tag{17}$$

Out of these two, SAD is preferable since it is an easier test.

3.1.2 Semi-Global Block Match

Introduced in [Hir05] by Hirschmuller and is an extension upon the simple block matching

Semi-global algorithms are used to locate transitions in a picture when very little a priori knowlegde i had bout the picture.

3.2 Energy minimization algorithms

Energy minimization is the process of defining an energy function for a problem and the minimizing this thus giving a minimum. Depending on the algorithm it may be either a local minimum or a global minimum. A collection of such algorithms are covered in [SZS$^+$08] and will be part of the basis for the algorithms tested.

3.2.1 Iterated conditional modes, (ICM)

Iterated conditional mode [HD92] is a deterministic, greedy algorithm. The way it works is that for each iteration it will choose the disparity that gives the greatest reduction in energy until it converges. Because it always chooses a lower value, it will converge, and usually quite fast. The problem is that it usually converges to a local minimum and not a global minimum.

Since it converges to a local minimum, it is very dependent on the initial estimate.

3.2.2 Graph cuts

There are two graph cut algorithms that will be evaluated. They are **Swap-move** and **Expansion-move**. Both introduced in [BVZ01] and improved upon in [KZ02]. an implementation is described in [BK04] using max-flow.

Compared to ICM, these graph cuts work my solving a global minimization problem instead of a global one.

Swap-move works by taking two subsets of two disparity levels, α, β, and swap their disparity levels and recalculate the energy function. If it produces a lower minimum, the swap-move is kept.

Expansion-move works similarity, but for a set of pixels assigned the disparity level α, it will add more pixels and recalculate the energy function. Keeping it if it produces a lower energy.

3.2.3 Belief propagation

Belief propagation is a message passing algorithm for performing inference on graphical models. It calculates the marginal distribution for each unobserved node, conditional on any observed nodes.

A thorough description is given in [FH04] the implementation used later is described in [TF03].

Note that belief propagation is not guaranteed to converge and may get stuck in an infinite loop repeatedly reassign disparity levels.

3.2.4 Tree-reweighted message passing, TRW

Tree-reweighted message passing was introduced in[WJW05] and an implementation is described in [Kol06].

TRW works similarity to the way belief propagation, but the message passing is different. See implementation for further information.

3.2.5 Variational stereo matching

Variational stereo matching usually is solved the following way. The disparity is embedded in a functional. Then the function is converted to solve a Euler-Lagrange function and a numerical method is then used to solve the Euler-Lagrange function.

The implementation evaluated later is described in [ZLDH11].

4 Platform

In this section an overview of the various hardware and software used for evaluating stereo matching algorithms and building a computer vision module.

4.1 Software

In the following section the various software used in this thesis is presented.

4.1.1 OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision library. It provides a foundation for making computer vision applications by providing accelerated functions for many of the graphics operations needed. OpenCV is available under a BSD-license and the version used i this thesis is 2.4.8.

In this project OpenCV has been used for:

- Calibrating the cameras
- Grabbing frames from the cameras
- Rectifying the frames
- Useful data formats for images of varying characteristics, matrices etc
- Displaying the images.
- Normalizing the disparity maps for visual inspection.
- Provides optimized Block Match, Semi-Global Block Match and Variational stereo matching implementations.
- Gaussian blur.
- Conversions between color spaces.
- Set camera properties.

4.1.2 MRF

MRF 2.2, Markov random fields, is a library offered from the Middlebury Vision lab. It provides a set of stereo matching algorithms that will be evaluated in this thesis.

4.1.3 Operating System

Arch Linux
Arch Linux is rolling release type Linux distribution which means that instead of releasing a new version periodically, Arch Linux is continuously updated with the latest software available.

Arch Linux only supports 32-bit and 64-bit x86-computers but the similarly named project Arch Linux Arm offers optimized packages for ARMv5te, ARMv6 and ARMv7 and root filesystem archives for both the Pandaboard and the BeagleBone Black. Both with hardware accelerated floating point math enabled.

Angstrom Linux
Angstrom Linux is the default Linux installed on the BeagleBone Black. By default hardware accelerated floating point math is not enabled which explains some results found later. The author does not have any personal experience with this Linux variant besides that gained during this thesis which is minimal.

4.1.4 Compilers

GCC
The default C-compiler found in most Linux distributions is the Gnu Compiler Collection. The version used in this thesis is the 4.8 series.

GCC were used to compile auto-vectorized versions of OpenCV on both x86_64 and ARMv7. The results are covered in their own sections.

LLVM and Clang
As a supplement to GCC, LLVM with Clang were used to evaluate the auto-vectorization capabilities of both compilers in order to choose the best one. LLVM did not provide any added benefit and further use were discontinued.

ARMCC
ARM does offer a compiler for its own platform and it would be interesting to see if, and if so, how much better the code produced would perform. Some time were spent on getting a working environment going but due to license issues and little time available, the effort were abandoned.

ICC
Intel does also offer a compiler collection for their processors. While the binaries produced by GCC were good enough, some evaluation of the vectorization-capabilities offered by ICC could be interesting.

4.1.5 Other

Various other software were also used and a short description is offered here.

Git
Git is a revision control system used to keep track of code, results and thesis writing.

ImageMagick
ImageMagick is a collection of image manipulation tools used to resize images, change brightness, compare disparity maps and stitching together images.

LaTeX
LaTeX is a typesetting language used to produce this thesis.

Libreoffice
Libreoffice is a freely available office suite used in this thesis for spreadsheets and constructing graphs.

4.2 Programming languages used

A short summary of the various programming languages used in this thesis.

4.2.1 C and C++

Programming started in C until it was realized that the C-API offered by OpenCV was broken. After some investigation into why seemingly correct programs refused to compile, language used switched to C++. Some demonstrations are offered in C as they worked as expected.

4.2.2 Bourne Again SHell, bash

Most of the benchmarking were done using bash-script to systematically test various aspects of the stereo matching algorithms.

4.3 Hardware

Each piece of hardware used in evaluating the algorithms.

During this thesis, three different hardware platforms were used to evaluate the stereo matching software. They are each given a section of their own where relevant information regarding that platform is listed.

In addition a summary of relevant specifications for the cameras used to construct a stereo camera is given at the end of the hardware section.

4.3.1 Intel platform

The main development and testing were done on a Dell Optiplex 9010 workstation, hereby referred to as the Intel platform. A summary of relevant specifications is given in table 1

CPU:		Ivy Bridge i5-3470 (3.2 GHz, 4 cores)
Cache:	L1:	128 KiB (instruction) / 128 KiB (data)
	L2:	1024 KiB
	L3:	6144 KiB
RAM:		8 GiB (DDR3), 1600MHz
OS:		Arch Linux 64-bit
Year:		2012

Table 1: Listing of the Intel platform specifications.

4.3.2 Pandaboard

The Pandaboard were chosen to represent a low power ARM-platform and the main ARM-testing were done on this platform.

A summary of relevant specifications is given in table 2

CPU:		OMAP4430, ARMv7 Cortex A9 (1.0 GHz, 2 cores)
Cache:	L1:	32 KiB (instruction) / 32 KiB (data)
	L2:	1024 KiB
RAM:		1 GiB (DDR2), 400MHz
OS:		Arch Linux ARM
Year:		2009

Table 2: Listing of the Pandaboard specifications.

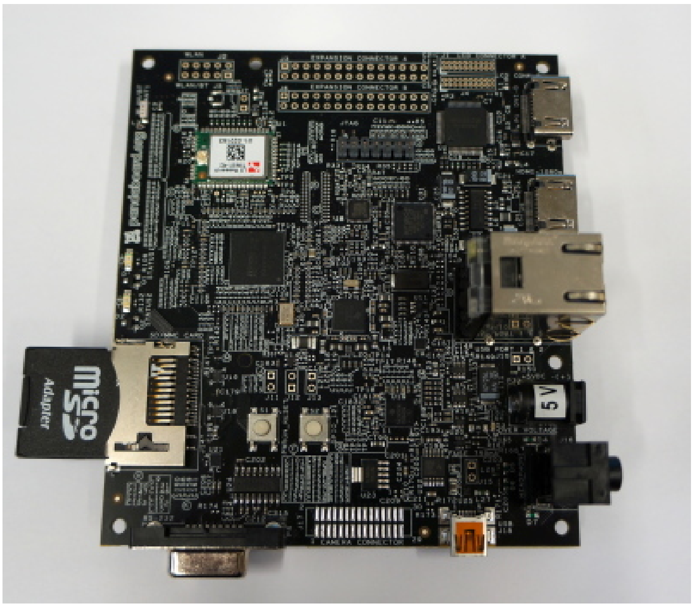


Figure 12: The Pandaboard

4.3.3 BeagleBone Black

A BeagleBone Black were borrowed from a different project to supplement the evaluation of the ARMv7 platform.

A summary of relevant specifications is given in table 3

Unlike the other platforms, the default Linux distribution on the BeagleBone Black is Angstrom Linux which does not enable the hardware accelerated floating point capabilities by default. Since the platform was only temporarily available, the operating system were not replaced by Arch Linux ARM.

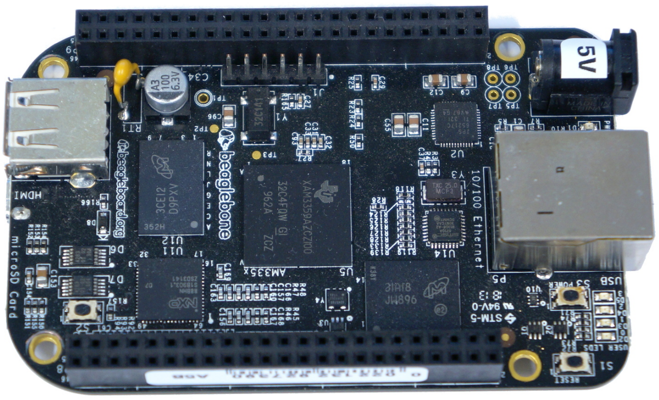


Figure 13: The BeagleBone Black

CPU:		AM3359, ARMv7 Cortex A8 (1.0 GHz, 2 cores)
Cache:	L1:	32 KiB (instruction) / 32 KiB (data)
	L2:	256 KiB
RAM:		512 MiB (DDR3), 800MHz
OS:		Angstrom Linux
Year:		2013

Table 3: Listing of the BeagleBone Black specifications.

4.3.4 Logitech C250 web cameras

In order to evaluate the stereo matching algorithms in a real-world scenario, a stereo camera were constructed using a pair of low cost Logitech C250 web cameras. The stereo camera were constructed by removing the plastic casing and mounting on a Plexiglas frame 10cm apart. This distance were chosen simply because 10cm is an easy number to work with. The stereo camera is depicted in figure 14

A summary of relevant specifications is given in table 4

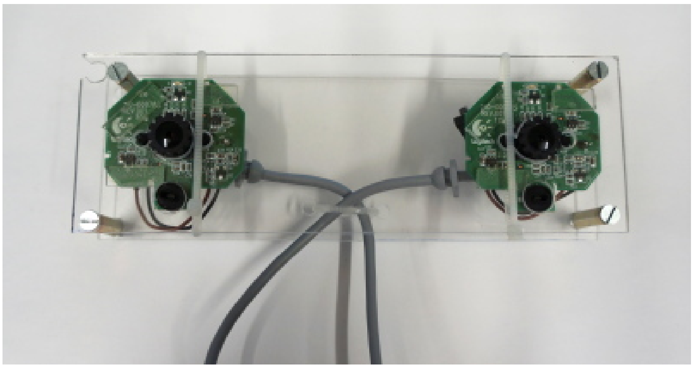


Figure 14: The stereo camera constructed

Connector	USB1
Max resolution	640x480
Resolution used	320x240

Table 4: Listing of the Logitech C250 specifications.

Because the USB-controller does not have enough bandwidth available to run both cameras on the native resolution of 640x480, the next available resolution, 320x240, were chosen. While regrettable at first, this reduction in resolution turns out to be a nonissue for reasons that are further explored in a later section.

The cameras used in this thesis are on the low-end. Better cameras are available without the bandwidth problems and may have features like timestamping which could prove to be useful when estimating the state of the surrounding environment.

4.4 Choice of stereo matching algorithm

The overall goal is to design a computer vision module for collision detection and avoidance. Specifically we want to emulate how humans observe their surroundings using stereo vision. In this section stereo matching algorithms along with the libraries that provide the implementations will be evaluated.

The algorithms will be evaluated according to a clear set of criteria and recommendations for which algorithms and libraries to further use will be given.

4.4.1 Libraries evaluated

The work required to implement the algorithms given earlier is substantial. Even more so to get fast implementations. Since there already exists available implementations of these algorithms, any effort to implement them would be wasted effort. Instead the implementations available are the ones that will be evaluated.

The first library to be evaluated is the OpenCV library that offers implementations of the following algorithms:

- Block Matching.
- Semi-Global Block Matching.
- Variational Stereo Matching.
- Belief Propagation (GPU-only)

Because neither the Pandaboard nor the Beaglebone Black has OpenCL drivers available, The GPU-implementations will not be evaluated.

The Semi-Global Block Matching has option to reduce the search directions evaluated. Both options will be evaluated to conclude if the full search is necessary.

The secondary library to be evaluated is Middlebury's Markov Random Field library, MRF for short. MRF offers implementations for quite a few graph cuts and belief propagation algorithms. The ones offered are:

- Iterated Conditional Modes.
- Expansion-move graph cut.
- Swap-move graph cut.
- Convergent Tree-reweighted Message passing. (TRWS).
- Synchronous Belief Propagation.
- Max Product Belief Propagation.

It should be noted that this library limits the possible disparity levels to 16. The results will be less smooth, but a good result may yet be had.

4.4.2 Criteria for evaluation

In order to fairly judge the algorithm-implementations, the following set of criteria is proposed:

1. Accuracy.
2. Efficiency.

Evaluating accuracy will be done by using a set of pictures, left and right, of a scene to compute a disparity map. Then this disparity map will be compared to the accompanying ground truth for that scene.

Efficiency will be evaluated by timing the algorithm-implementations on the available platforms.

In addition to the two criteria above, the algorithms that are selected as possible candidates will also be used in a set of special cases where they are expected to produce bad results. This to serve as a reminder that computer vision still is early in its infancy.

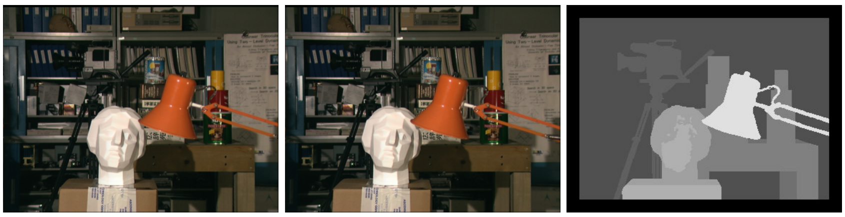
4.4.3 Middlebury stereo data sets

The Middlebury stereo data sets [SSZ01][SS03][SP07][HS07] refer to a set of data sets with 9 pictures each, 0-indexed, of a scene accompanied by ground truth for image 2 and 6. Image 2 and 6 will be used as left and right picture of the scene and the results will be compared to the ground truth belonging to the left image.

These data sets are the standard for which stereo matching algorithms are compared and evaluated.

There is a total of 38 data sets, so only 4 of them will be used. The Tsukuba set and the Venus set from the 2001 publication [SSZ01] and the Cones set and the Teddy set from the 2003 publication [SS03].

Tsukuba set
In figure 15 we have the left image, fig 15a, and the right, fig 15b, image of the Tsukuba scene accompanied by the ground truth, fig 15c, seen from the left image's point of view.

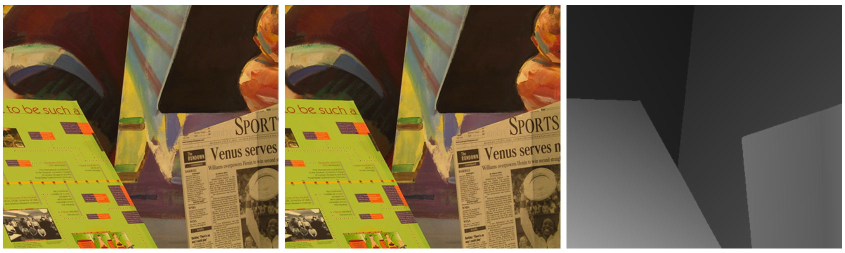


(a) Left image (b) Right image (c) Ground truth

Figure 15: The Tsukuba set

The ground truth were computed using known disparity of specific segments.

Venus set
In figure 16 we have the left image, fig 16a, and the right, fig 16b, image of the Tsukuba scene accompanied by the ground truth, fig 16c, seen from the left image's point of view.

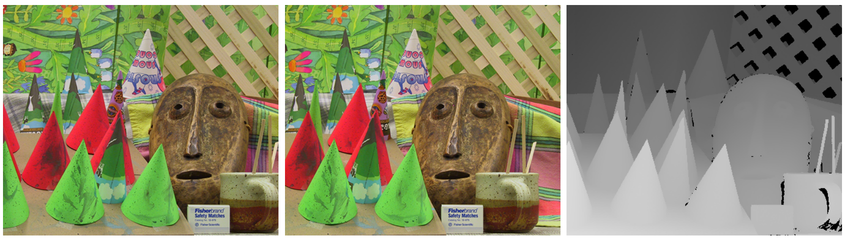


(a) Left image (b) Right image (c) Ground truth

Figure 16: The Venus set

Similarly to the Tsukuba set, the ground truth were computed from know disparity.

Cones set
In figure 17 we have the left image, fig 17a, and the right, fig 17b, image of the Tsukuba scene accompanied by the ground truth, fig 17c, seen from the left image's point of view.



(a) Left image (b) Right image (c) Ground truth

Figure 17: The Cones set

In the 2003 sets, ground truth were found using the structured light technique. The same that is used in the Kinect system.

Teddy set
In figure 18 we have the left image, fig 18a, and the right, fig 18b, image of the Tsukuba scene accompanied by the ground truth, fig 18c, seen from the left image's point of view.

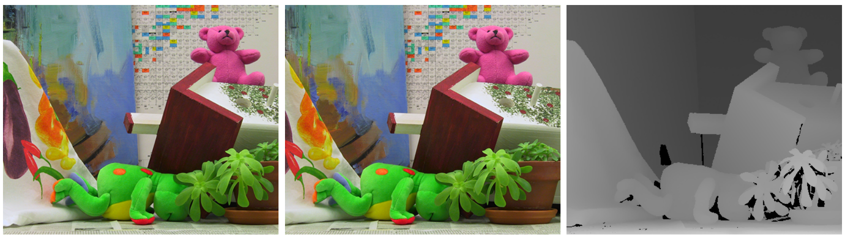

(a) Left image (b) Right image (c) Ground truth

Figure 18: The Teddy set

Like the Cones set, ground truth were found using structured light.

4.4.4 Testing methodology

Each algorithm will produce a disparity map from each of the data sets available. In some cases the produced disparity map will be scaled differently for the ground truth. When comparing the produced disparity map to the ground truth, two correctly identified segments corresponding to the brightest and darkest segments in the ground truth are used to scale the produced disparity map so that these segments have the same intensity. Every level in between should then also be the correct level unless the algorithm has done something wrong.

The scaled disparity map is subtracted from the ground truth and then a threshold of 3% is applied. Every pixel above 3% is then marked as black in a new image hereby referred to as the error map. The black pixels represent the errors made by the algorithm.

Rating the accuracy of the algorithms are done by visual inspection of both the disparity map and the error map.

A good result is one that captures the main features in the ground truth and have few to none black pixels in the error map.

A poor result is one where there is an extensive amount of black pixels in the error map but the main features from the ground truth can still be found in the disparity map.

A bad result is one where there is little to no similarity between the disparity map and the ground truth. The error map will have few to none white pixels left in such a result.

For each algorithm the computed disparity map and the error maps is presented alongside with the left image from the data set used and its corresponding ground truth for easy comparison.

A discussion on the accuracy of the algorithm is presented after the results. Then the efficiency results are presented with a short comment or a minor discussion should there be any results that need further explanation.

After the results from each library is presented, a short summary is given.

4.5 Results from OpenCV

4.5.1 Block matching

In figure 19, from left to right, first column is the left image from the Middlebury 2001 and 2003 data sets (Tsukuba, Venus, Cones and Teddy), second column is the ground truth, third is the computed disparity map using block match and in the last column the erroneous pixels are colored black.

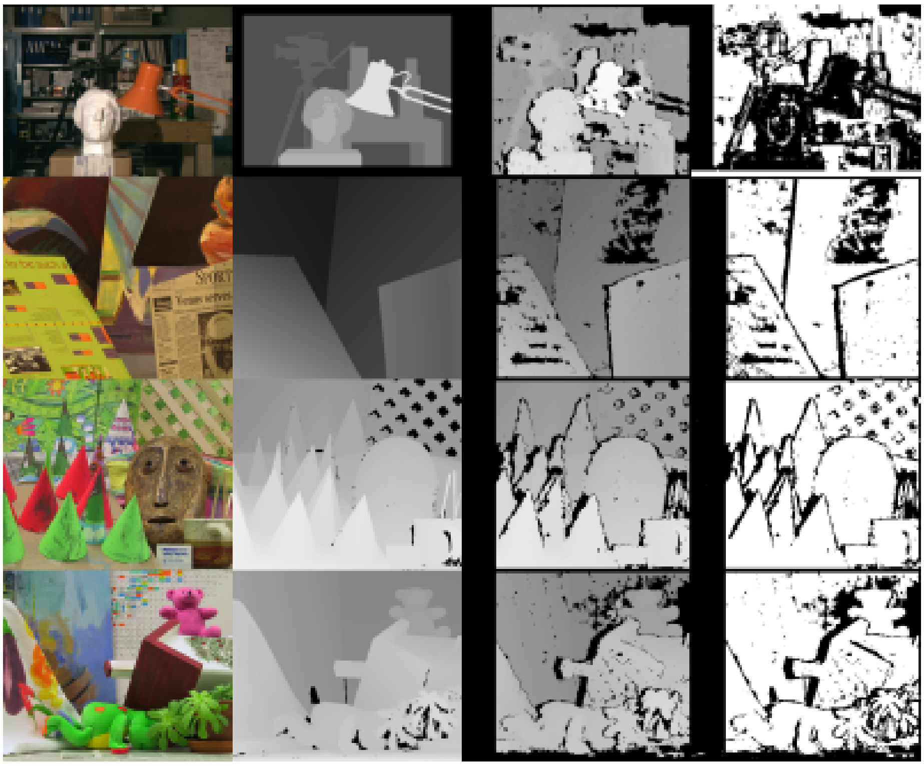


Figure 19: Accuracy results using block matching

The most noticeable feature in the results is the large, black bar on the left side and is caused by the algorithm skipping that amount into the picture before matching. It is possible to reduce this area an increased computational cost.

Except from a few problematic segments in the Venus set, the black areas in the computed disparity map for a contour to the individual objects in the scene. It is noticeably stronger on the left side, especially in the Teddy set, and is caused by occlusion and a left-centric approach. The disparity map is computed for the left image's point of view and occluded areas in the right image is ignored.

The few segments not accounted for is caused bye there being segments with little detail which gives multiple false matches. This is seen to the top right area in the Tsukuba set, top right in the Venus set and around the teddy in the Teddy set.

Overall there are quite the few erroneous pixels, but the at the same time we see that main features in the ground truth is found using block matching.

The time used to run block matching on the data sets is given in table 5

Block Matching	Tsukuba	Venus	Cones	Teddy
Intel platform	7.3 ms	7.0 ms	7.0 ms	7.3 ms
Pandaboard	313 ms	600 ms	611 ms	607 ms
BeagleBone Black	196 ms	427 ms	425 ms	425 ms

Table 5: Block matching

The difference is staggering between the Pandaboard and the Intel platform. For this specific algorithm the Pandaboard uses about 80 times as much time as the Intel platform. Even if it is this slow, it can be used in an application that need a fast response to changes in the surroundings.

4.5.2 Semi-global block matching

In figure 20, from left to right, first column is the left image from the Middlebury 2001 and 2003 data sets (Tsukuba, Venus, Cones and Teddy), second column is the ground truth, third is the computed disparity map using semi-global block match and in the last column the erroneous pixels are colored black.

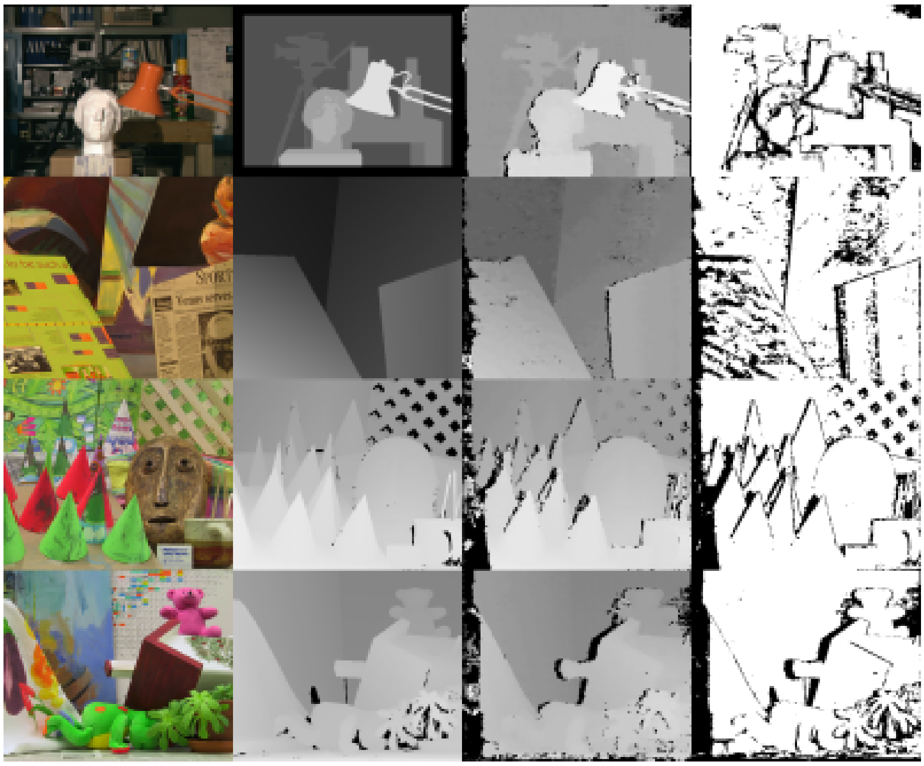


Figure 20: Accuracy results using semi-global block matching

Compared to simple block matching, semi-global block matching produces far lest black spots. There is still a black area to the left, but much smaller. In Tsukuba, to the top right, the black spot is smaller. The problematic areas in

Venus are gone and the only prominent feature is caused by occlusion in Teddy.

Some errors are produced by the disparity level being slightly wrong. The computed segments are not as smooth as the corresponding areas found in the ground truth.

Besides the errors produced by being left-centric, the results are very good.

The time used to run semi-global block matching on the data sets is given in table 6

SG Block Matching	Tsukuba	Venus	Cones	Teddy
Intel platform	32.9 ms	59.7 ms	62.6 ms	62.6 ms
Pandaboard	2571 ms	4986 ms	5040 ms	5072 ms
BeagleBone Black	1700 ms	3362 ms	3527 ms	3580 ms

Table 6: Semi-global block matching

Already here it shows that by itself the ARM based platforms can not be used without considerable speedup. 1 frame per 5 seconds is too slow to avoid any collision.

The Intel platform still performs very good with about 17 frames per second.

4.5.3 Semi-global block matching, all directions

Essentially the same as the previous algorithm but with all directions evaluated.

In figure 21, from left to right, first column is the left image from the Middlebury 2001 and 2003 data sets (Tsukuba, Venus, Cones and Teddy), second column is the ground truth, third is the computed disparity map using semi-global block match with all directions evaluated and in the last column the erroneous pixels are colored black.

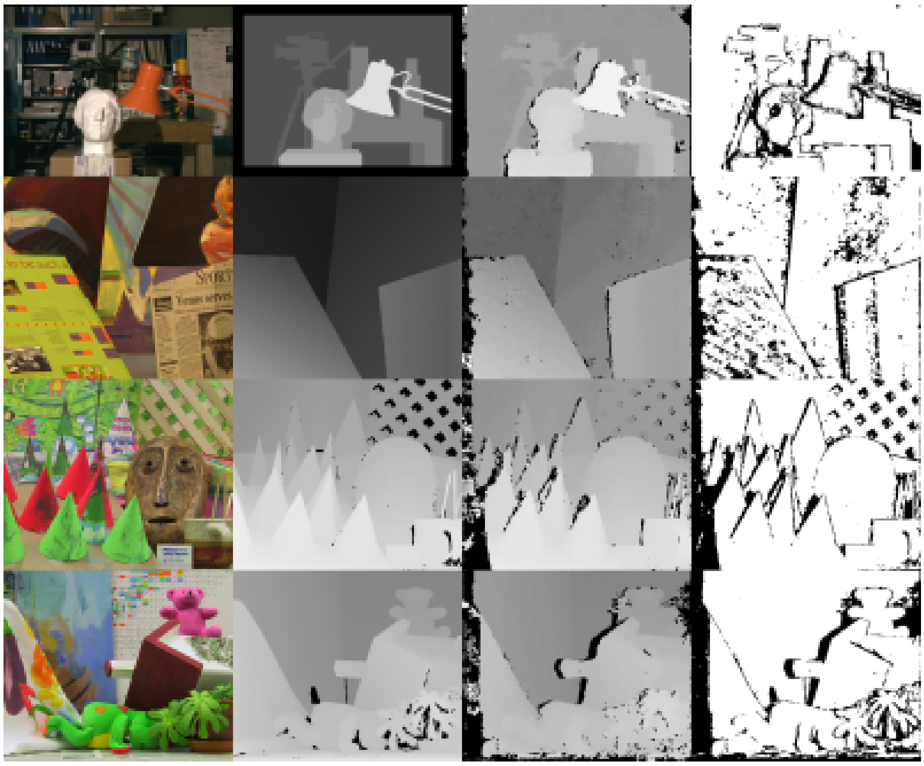


Figure 21: Accuracy results using semi-global block matching with all directions evaluated

The results are essentially the same as those found by SGBM so the conclusion is simple. Evaluating all eight directions is unnecessary as it does not add any benefit.

The time used to run semi-global block matching (all directions) on the data sets is given in table 7

SGBM-all	Tsukuba	Venus	Cones	Teddy
Intel platform	47.0 ms	88.9 ms	91.4 ms	91.6 ms
Pandaboard	3373 ms	6608 ms	6656 ms	6701 ms
BeagleBone Black	2338 ms	4460 ms	4641 ms	4700 ms

Table 7: Semi-global block matching, all directions

Evaluating all directions increases the computational cost and that is no good for the ARM-platforms. On the Intel platform the frame rate is down to about 11 frames per second which is still very good and enough to detect a collision within reason.

4.5.4 Variational stereo matching with consistency constraint left/right

In figure 22, from left to right, first column is the left image from the Middlebury 2001 and 2003 data sets (Tsukuba, Venus, Cones and Teddy), second column is the ground truth, third is the computed disparity map using variational stereo matching and in the last column the erroneous pixels are colored black.

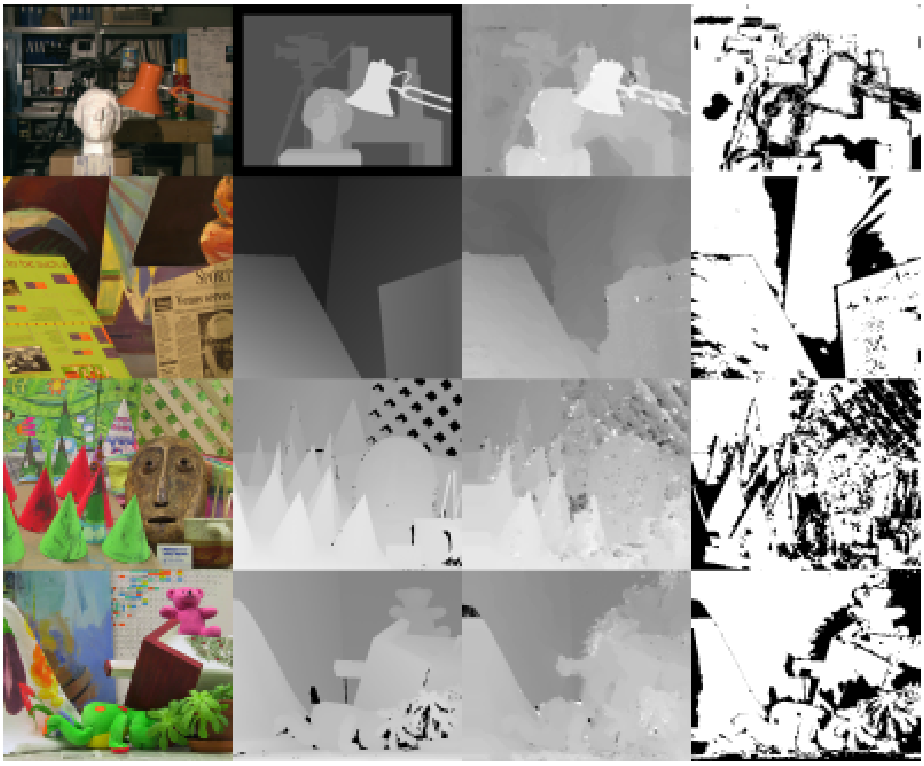


Figure 22: Accuracy results using Variational stereo matching

With a slight exception in Teddy, there are no black spots meaning that occluded spots are assigned a disparity level. This would have been an improvement over SGBM and BM had it only been accurate as well. Comparing the result with the ground truth reveals that OpenCV's implementation produces a poorer result compared to SGBM.

It would be unfair to conclude that the results are poor as they do in each one get the main features found in the ground truth. The problem however it that some errors are not just a little off, but quite a lot. The disparity map produced using the Cones set is very problematic as there are several bright spots that are clearly quite wrong.

It should be noted that the Cones set produces the worst result, but the other sets are not that far behind. In each result, where there should be clear, crisp edges, there is instead patchy or diffuse edges.

Despite the lack of black areas produced by occlusion, the results produced using variational stereo matching are not better than those produced by SGBM.

The time used to run variational stereo matching on the data sets is given in table 8

Variational	Tsukuba	Venus	Cones	Teddy
Intel platform	141 ms	404 ms	409 ms	409 ms
Pandaboard	4570 ms	13269 ms	13310 ms	13380 ms
BeagleBone Black	8782 ms	20386 ms	21410 ms	20773 ms

Table 8: Variational

Variational stereo matching is the most costly of the algorithms provided by OpenCV. The Pandaboard uses 13 seconds on one frame while the BeagleBone Black uses as much as 21 seconds. Clearly something is different here since the BeagleBone is slower. This issue is will be explored further in the next section.

The Intel platform is also slowing down to just over 2 frames per second. Considering the performance of the other algorithms offered, there is little reason to use variational stereo matching in any collision detection module.

4.6 MRF library

The second library evaluated is the algorithms implemented in the MRF library found on the Middlebury Vision website. Details in given in [SZS$^+$08]

It should be noted that unlike OpenCV's algorithms, all the ones found in MRF specifies a discrete disparity levels. For each algorithm, 16 discrete disparity levels where used.

4.6.1 Iterated conditional mode

In figure 23, from left to right, first column is the left image from the Middlebury 2001 and 2003 data sets (Tsukuba, Venus, Cones and Teddy), second column is the ground truth, third is the computed disparity map using iterated conditional mode, ICM, and in the last column the erroneous pixels are colored black.

The disparity maps computed using iterated conditional mode are somewhat different from those found in OpenCV.

The disparity maps computed using the Tsukuba set and the Venus set are the least problematic. A visual inspection of the results will tell that several features like the lamp, the head and the newspaper is close to being correct. The rest of the disparity map is filled with errors.

The disparity maps produced using the Cones set and the Teddy set are very problematic. Any observer would have great difficulty telling what the scene is supposed to be from the results.

In conclusion the results produced using iterated conditional mode are too poor for any further use.

The time used to run iterated conditional mode on the data sets is given in table 9

Out of the algorithms offered by MRF, ICM is the fastest. Yet only comparative to variational stereo matching.

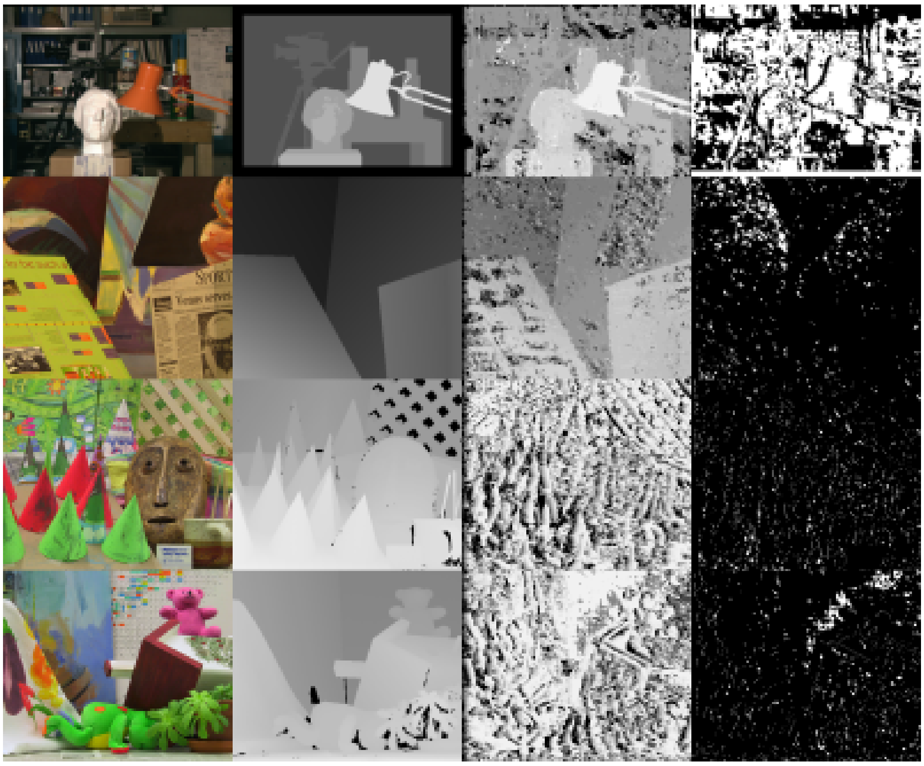


Figure 23: Accuracy results using ICM

ICM	Tsukuba	Venus	Cones	Teddy
Intel platform	0.4 s	0.4 s	0.5 s	0.4 s

Table 9: Iterated conditional mode

4.6.2 Expansion-move graph cut

In figure 24, from left to right, first column is the left image from the Middlebury 2001 and 2003 data sets (Tsukuba, Venus, Cones and Teddy), second column is the ground truth, third is the computed disparity map using expansion-move method and in the last column the erroneous pixels are colored black.

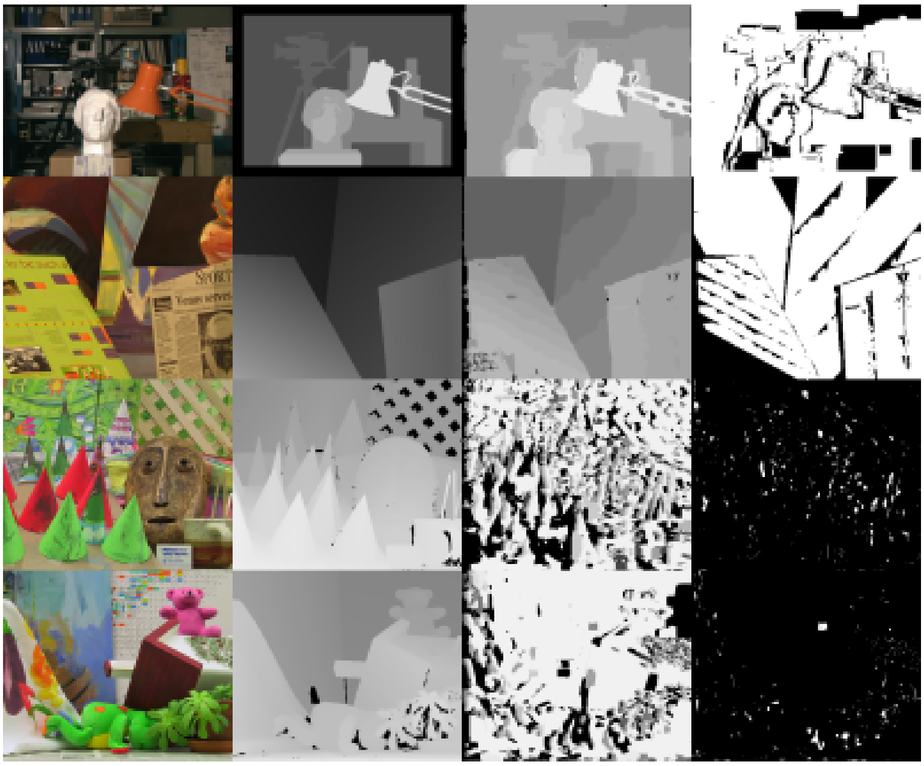


Figure 24: Accuracy results using expansion

The results found by using the Tsukuba set and the Venus set are quite good. There are few black spots. The edges are crisp and the areas that are marked as wrong are for the most part marked as such because of only allowing 16 disparity levels. Some segments are marked wrong, but they are not far off.

The results found by using the Cones set is chaotic and a poor match to the ground truth.

The Teddy set is less chaotic, but most segments are similarly far off.

Disparity maps produced using expansion are also too poor for further use.

The time used to run expansion-move graph cut on the data sets is given in table 10

Exp	Tsukuba	Venus	Cones	Teddy
Intel platform	2.7 s	5.5 s	9.8 s	12.2 s

Table 10: Expansion-move graph cut

Even for the rather small resolution of the Tsukuba set, 2.7 seconds is still far too long. The extreme case of the Teddy set is even worse.

4.6.3 Swap-move graph cut

In figure 25, from left to right, first column is the left image from the Middlebury 2001 and 2003 data sets (Tsukuba, Venus, Cones and Teddy), second column is the ground truth, third is the computed disparity map using Swap method and in the last column the erroneous pixels are colored black.

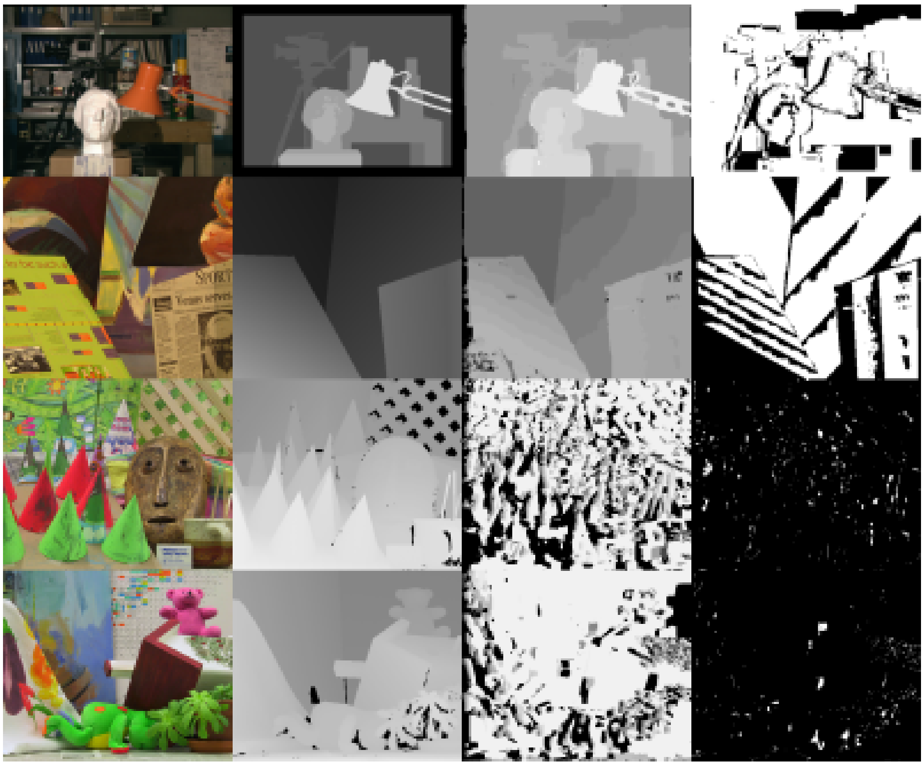


Figure 25: Accuracy results using swap

The results found using the swap method are comparable to those found using expansion. The Tsukuba set and the Venus set produces some errors, but nothing major, while the Cones set and the Teddy set are too chaotic.

As with the other energy minimization algorithms, Swap is too poor for further use.

The time used to run swap-move graph cut on the data sets is given in table 11

Swap	Tsukuba	Venus	Cones	Teddy
Intel platform	2.5 s	4.7 s	5.8 s	9.9 s

Table 11: Swap-move graph cut

The timing results from swap-move is comparative to expansion-move and that is too slow.

4.6.4 Sequential tree-reweighted message passing for energy minimization

In figure 26, from left to right, first column is the left image from the Middlebury 2001 and 2003 data sets (Tsukuba, Venus, Cones and Teddy), second column is the ground truth, third is the computed disparity map using Sequential Tree-re-weighted Message Passing and in the last column the erroneous pixels are colored black.

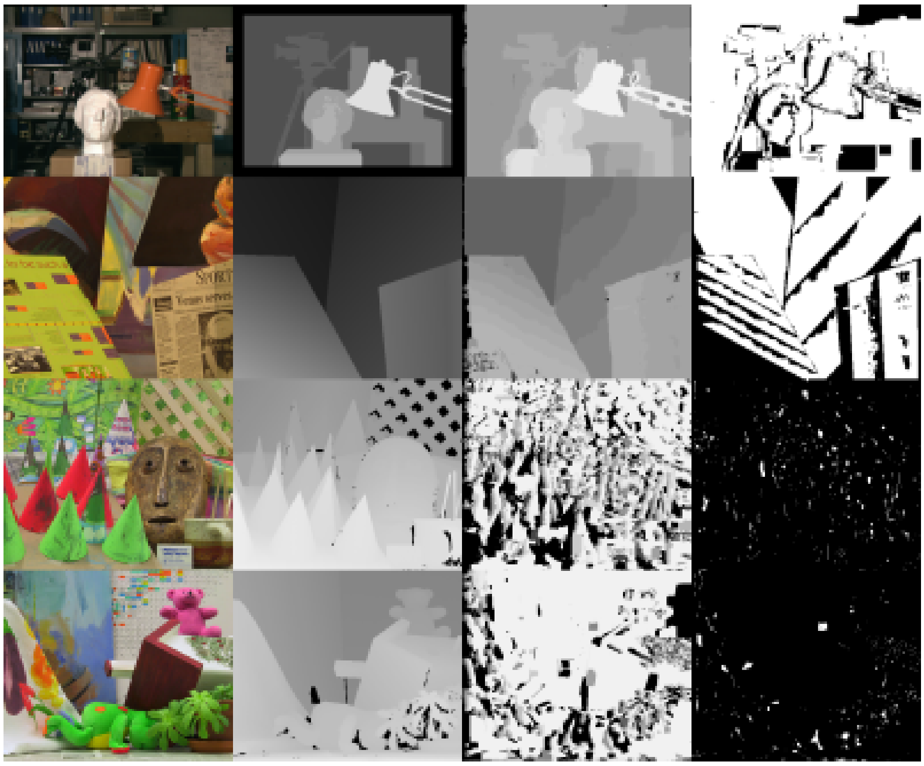


Figure 26: Accuracy results using Sequential Tree-re-weighted Message Passing

Like Swap-move and Expansion-move, TRWS produces comparable results. The Tsukuba set and the Venus set are the ones that produces the best results with minor errors, while the Cones set and the Teddy set produces chaotic results.

Like the previous algorithms, TRWS is also producing too poor results.

The time used to run sequential tree-reweighted message passing on the data sets is given in table 12

TRWS	Tsukuba	Venus	Cones	Teddy
Intel platform	6.1 s	10.1 s	47.7 s	47.7 s

Table 12: Sequential tree-reweighted message passing

Tree-reweighted message passing is even slower. The results from the Teddy set and the Cones set especially stand out. The explanation is that the algorithm is “stuck” and sort of times out.

4.6.5 Synchronous belief propagation

In figure 27, from left to right, first column is the left image from the Middlebury 2001 and 2003 data sets (Tsukuba, Venus, Cones and Teddy), second column is the ground truth, third is the computed disparity map using synchronous belief propagation and in the last column the erroneous pixels are colored black.

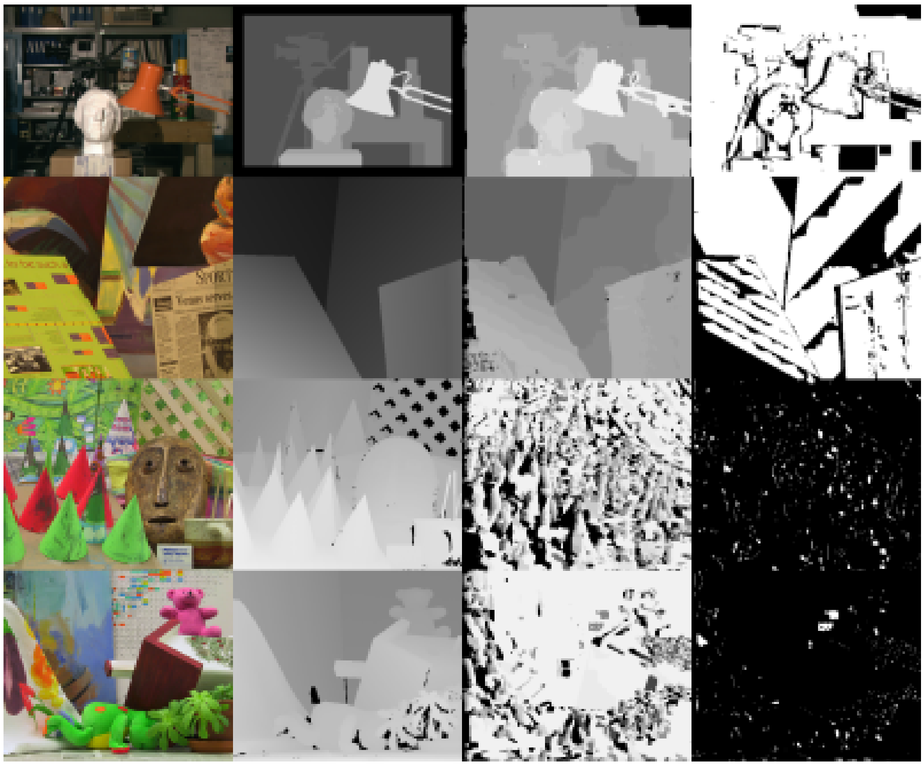


Figure 27: Accuracy results using Synchronous Belief Propagation

Out of the energy minimization techniques, synchronous belief propagation is the one that has produced the most accurate disparity map for the Tsukuba set. There are some segments at the top and at the bottom that are marked as errors, except for those the result is comparable to SGBM.

The disparity map produced using the Venus set is also quite good. Most of the errors can be attributed to the limited amount of disparity levels allowed.

The results produced using the Cones set and the Teddy set are comparable til the previous MRF-algorithms. They are chaotic and of such low accuracy that this algorithm will not be used any further.

The time used to run synchronous belief propagation on the data sets is given in table 13

BP-S	Tsukuba	Venus	Cones	Teddy
Intel platform	1.4 s	4.7 s	34.3 s	34.3 s

Table 13: Synchronous belief propagation

Synchronous belief propagation gives a decent result on the Tsukuba set. And like TRWS is seems “stuck” on the Cones set and the Teddy set.

4.6.6 Max product belief propagation

In figure 28, from left to right, first column is the left image from the Middlebury 2001 and 2003 data sets (Tsukuba, Venus, Cones and Teddy), second column is the ground truth, third is the computed disparity map using Belief Propagation M and in the last column the erroneous pixels are colored black.

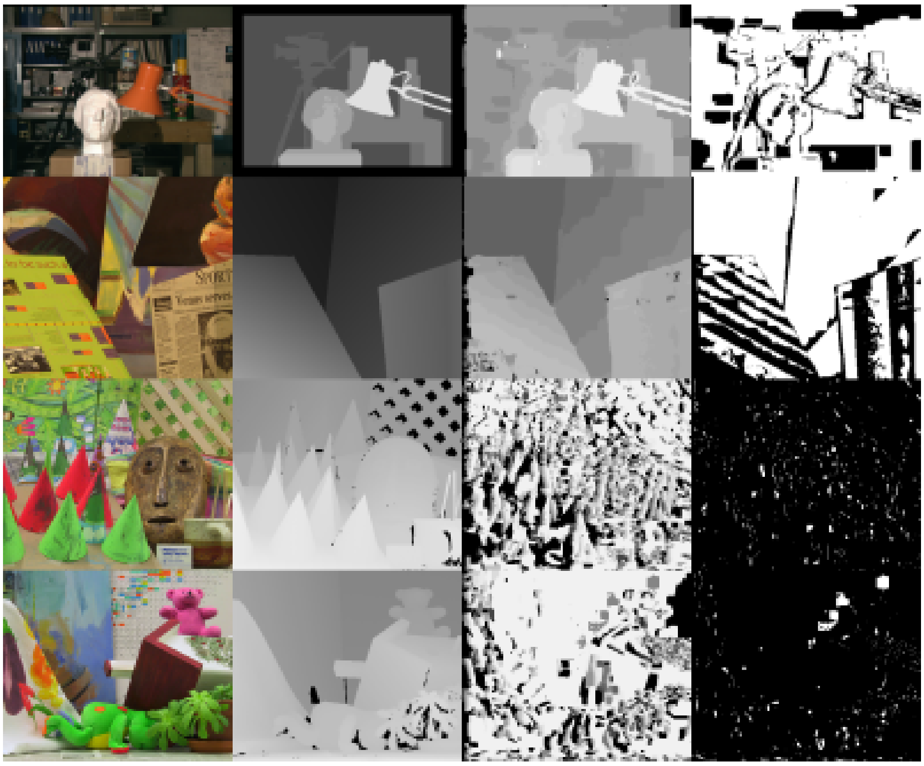


Figure 28: Accuracy results using max product belief propagation

Similar to synchronous belief propagation, but slightly worse. More segments are marked as wrong in the Tsukuba set, the Venus set is almost identical and the two last ones are similarly chaotic. Neither this algorithm will see any further use.

The time used to run max product belief propagation on the data sets is given in table 14

BP-M	Tsukuba	Venus	Cones	Teddy
Intel platform	9.4 s	10.3 s	36.2 s	33.7 s

Table 14: Max product belief propagation

Max product belief propagation is the slowest of the algorithms. Like the previous algorithm, it also seems stuck on the last two data sets.

It seems likely that the chaotic result and the “stuck” energy minimization is related.

4.7 MRF's accuracy problems revisited

The most unsettling about the accuracy of the results produced so far with MRF is that the results in other publications are very good. Clearly something is wrong and it turns out that MRF was used in a sub-optimal manner.

It is possible to set the amount of disparity levels allowed to a much higher number than the default of 16. The problem is that MRF was ruled out because of the time and not the accuracy, but a fairer test with handpicked disparity levels should be used if proper accuracy were to be tested.

By the time MRF were revisited, there were too little time left to redo the experiments. To mend the situation, this section is provided instead.

The results from the MRF-algorithms is clear. The Tsukuba set will do nicely with 16 disparity levels.

According to the paper published on MRF[SZS$^+$08] suggests that 20 disparity levels should be used on the Venus set and 60 on the Teddy set.

In figure 29 a comparison of accuracy versus the amount of disparity allowed is presented.

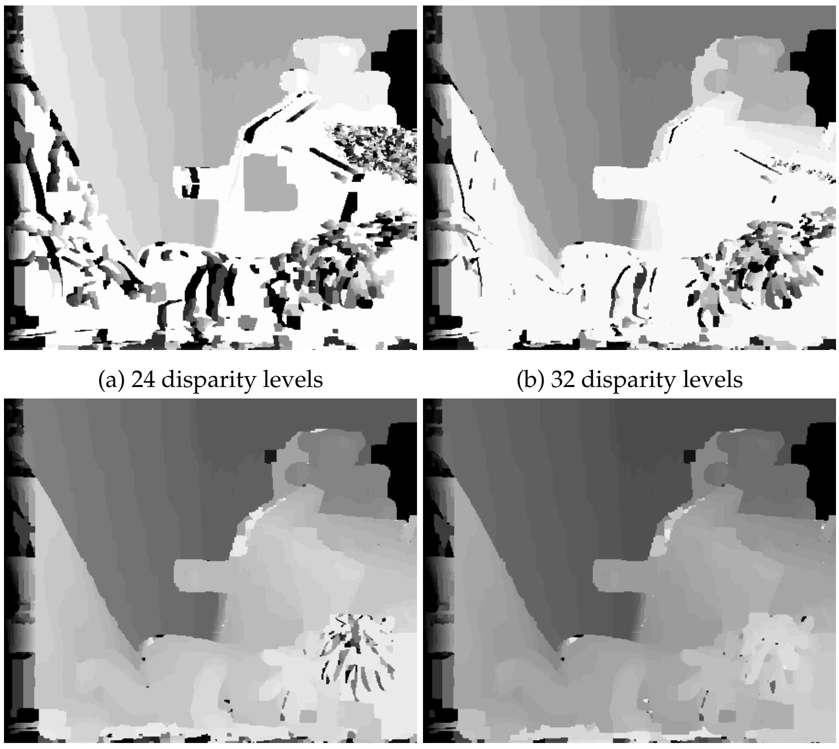
Given enough disparity levels, MRF's expansion-move will give good results but at an increased cost as seen in table 15.

Disparity	16	24	32	40	48	60
Time(s)	12.2 s	19.4 s	19.3 s	28.7 s	34.0 s	44.0 s

Table 15: Increase in time with more disparity levels using expansion-move on Teddy set

The problem should be clear where this is going. Increasing the amount of disparity levels, greatly increases the computational cost. Going from 16 disparity levels up to 60 increases the computational cost by a factor of almost 4.

What then is the value of the results gathered so far? The first thing is that methods like graph cuts and belief propagation needs to be allowed a certain


(a) 24 disparity levels (b) 32 disparity levels

(c) 40 disparity levels (d) 48 disparity levels

Figure 29: Accuracy versus disparity levels allowed in MRF's Expansion-move

amount of disparity to produce good results. If not, then they don't work at all.

Secondly is that even if the method were efficient enough for use in a live system in the first place, the methods would have to be efficient enough to use with a large range of values. This creates a situation were the MRF would either work or not at all unless some guarantee can be made about the maximum depth in the picture.

The conclusion still stands that MRF simply is too slow for any use even if the accuracy-issue is addressed.

4.8 Closer look at performance issues

The results are quite clear that there is a significant gap in performance between the ARM-platforms and the Intel-platform. In this section a closer into what happens under the surface that would explain some gap as well as what steps can be taken to reduce it.

4.8.1 Pandaboard vs BeagleBone Black

Except for variational stereo matching, all test ran faster on the BeagleBone Black than it did on the Pandaboard. The improvement is listed in table 16.

Alg.	Tsukuba	Venus	Cones	Teddy
BM	60 %	41 %	44 %	43 %
SGBM	51 %	48 %	43 %	42 %
SGBM-all	44 %	48 %	43 %	43 %
Var.	-48 %	-35 %	-38 %	-36 %

Table 16: Percentage-wise improvement when running on the BeagleBone Black over the Pandaboard.

The most peculiar result here is that variational stereo matching is slower while all the others are faster. This is because variational stereo matching is very dependent on floating point math which is hardware accelerated by default on Arch Linux ARM. Had it been enabled in Angstrom, then the results would be very different.

The Tsukuba set which is a lower resolution than the other data sets. This is a peculiar result which will be further explored later in this section.

The last thing to note is that the BeagleBone Black is about 40 % to 60 % faster than the Pandaboard which is most likely explained by faster RAM and the difference between ARC Cortex A9 and ARM Cortex A8.

4.8.2 Pandaboard vs Intel

Considering the results found when running the same code on the Intel platform and the Pandaboard, the difference in performance is vast. In some test cases the Pandaboard used as much as 75 times as much time.

This performance gap raises some questions. Why is the performance of the Pandaboard so poor. The ARM processor is clocked slower than the Intel processor. The Pandaboard is equipped with DDR2 ram compared to DDR3 on the Intel platform, yet neither of these things are sufficient to explain the performance gap.

To explain this it is perhaps best to look at it differently. Why does the Intel platform perform so much better? Part of the answer lies in the nature of what OpenCV works, which is a lot of matrix operations. Accelerating such operations will yield a better performance. Which leads to the matter of SIMD-instructions sets like MMX and SSE2.

OpenCV started as an Intel project. That the code were written to take advantage of the capabilities present on modern Intel processors should come as no surprise.

As mentioned earlier, the block matching algorithms compare blocks by either computing the Sum of Absolute Differences, SAD, or by computing the Mean Square Error, MSE. Of these, SAD is preferable because the operation costs less CPU-cycles. But the real improvement comes from special instructions provided by SSE2, **psadbw**, which computes the Sum of Absolute Differences of an x by x block in 3 or 4 cycles.

Disassembling parts of the OpenCV libraries and looking for psadbw should give positive results:

objdump -d libopencv_core.so | grep sad

```
  21e760:        66 0f f6 ca              psadbw  %xmm2,%xmm1
  21e7b3:        66 0f f6 ca              psadbw  %xmm2,%xmm1
```

objdump -d libopencv_imgproc.so | grep sad

```
  19d616:        66 41 0f f6 d6           psadbw  %xmm14,%xmm2
```

ARMv7 (Non-x86 platforms in general) does not have this SSE2-instruction set. Instead ARM has their own SIMD-instruction set called NEON. Like SSE2, NEON provides an instruction for computing the Sum of Absolute Differences, USAD.

Disassembling the ARMv7-versions of OpenCV does not find this NEON-instruction.

In Arch Linux, one can find the default build options in **/etc/makepkg.conf**. These build options can in turn be overridden in each separate package. Inspecting these files suggest that OpenCV was not built with NEON-support but only with VFPv3.

4.8.3 Improving ARM performance using aggressive compiler options

In the previous section it was found that OpenCV was most likely not compiled with NEON-support. Rebuilding OpenCV with NEON-support is a good starting step to increase the performance.

Since OpenCV is not specifically written with NEON-support, an attempt at increasing performance is done by using the compilers auto-vectorization capabilities.

The following compiler options were used:

```
armv7l-unknown-linux-gnueabihf-g++ -O3 stereo_match.cpp
$(pkg-config --libs opencv) -mfpu=neon -ftree-vectorize
-mfloat-abi=hard -funsafe-math-optimizations
```

The results are listed in table 17

The results are impressive given that no changes to OpenCV were necessary.

Block matching had improved by about 11% which is good. Semi-global block matching improved the most both reduced directions and full directions. Since it is the same code that is running, the lower improvement on the full directions suggests that the part of the code responsible for checking directions were not improved much or at all and instead a part of the code that is the same both times. The 8 % improvement of an algorithm that uses much floating point

BM	313 ms	600 ms	611 ms	607 ms
NEON	285 ms	538 ms	549 ms	547 ms
Imp.	10 %	12 %	11 %	11 %
SGBM	2571 ms	4986 ms	5040 ms	5072 ms
NEON	1839 ms	3539 ms	3604 ms	3608 ms
Imp.	40 %	41 %	40 %	41 %
SGBM-all	3373 ms	6608 ms	6656 ms	6701 ms
NEON	2571 ms	4948 ms	5029 ms	5034 ms
Imp.	31 %	34 %	32 %	33 %
Var.	4570 ms	13269 ms	13310 ms	13380 ms
NEON	4234 ms	12238 ms	12306 ms	12341 ms
Imp.	8 %	8 %	8 %	8 %

Table 17: Improvement of auto-vectorized code on the Pandaboard

math is also impressive.

There are clear improvements here, but still far off. Disassembly of the new libraries did not show any sign of the USAD-instruction.

The auto-vectorized binary was only tested on the Pandaboard as the BeagleBone Black required invasive changes to the software stack. Since it was borrowed from a different project it was decided against spending time on this.

4.8.4 Improving Intel performance using aggressive compiler options

Considering the improvement gained by using aggressive compiler flags on the ARMv7 platform, it is natural to consider the possible improvements on the Intel platform as well. As previously noted, OpenCV is already favorably optimized on x86_64.

As noted previously, Arch Linux provides OpenCV-binary libraries built with SSE2 optimizations. It is unlikely that there is much more to gain from further optimizations compared to the standard binary.

The Intel processor supports the following instruction sets in addition to SSE2: PNI(SSE3), SSSE3, SSE4(SSE4.1 and SSE4.2) and AVX.

The following compiler options where used:

```
-march=x86-64 -mtune=native -O3 -mssse3 -msse4.1 -msse4.2
-mavx -pipe -fstack-protector --param=ssp-buffer-size=4
```

Alg.	Tsukuba	Venus	Cones	Teddy
BM	6.7 s	6.3 s	8.9 s	6.7 s
SGBM	32.5 s	58.6 s	61.5 s	61.8 s
SGBM-all	46.6 s	87.6 s	90.7 s	90.4 s
Var.	144 s	413 s	417 s	417 s

Table 18: timings of algorithms running auto-vectorized code

Alg.	Tsukuba	Venus	Cones	Teddy
BM	9 %	11 %	-22 %	9 %
SGBM	1 %	2 %	2 %	1 %
SGBM-all	1 %	1 %	1 %	1 %
Var.	-2 %	-2 %	-2 %	-2 %

Table 19: Percentage-wise improvement of auto-vectorized code

The small difference before and after can easily be attributed to other factors like competing processes and the accompanying scheduling. There is no benefit to further optimization on the Intel platform. The large percentage fluctuations from BM is simply because it is very fast to begin with.

Disassembly of the compiled OpenCV library shows that where there previously were **psadbw**-instructions, there is now found the AVX counterpart **vpsadbw**.

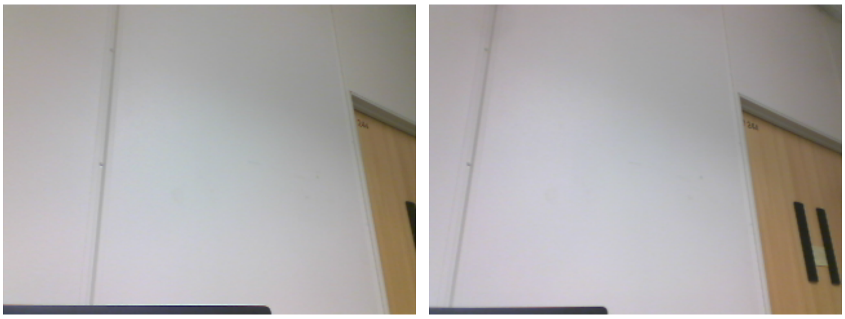
Similarly, if AVX is not used, GCC produces binaries with the SSE4.1 counterpart **mpsadbw**.

In order to gain the benefit of these wider instructions, changes in the form of better alignment is needed in the OpenCV library.

4.9 Special cases where stereo matching does not work

Considering the disparity maps produced when running OpenCV algorithms on the Middlebury data sets, the results seem very optimistic in what they are capable of. In reality the scenes are often far less perfect. Occlusion has already been covered earlier. That is to say segments in a stereo image set that only exists i one of them as some other segment is overlapping where it should have been.

Another case where these algorithms fail to perform is in scenes with ambiguous segments. For example a white wall will have very little information to match against. Take for example the scene in figure 30:



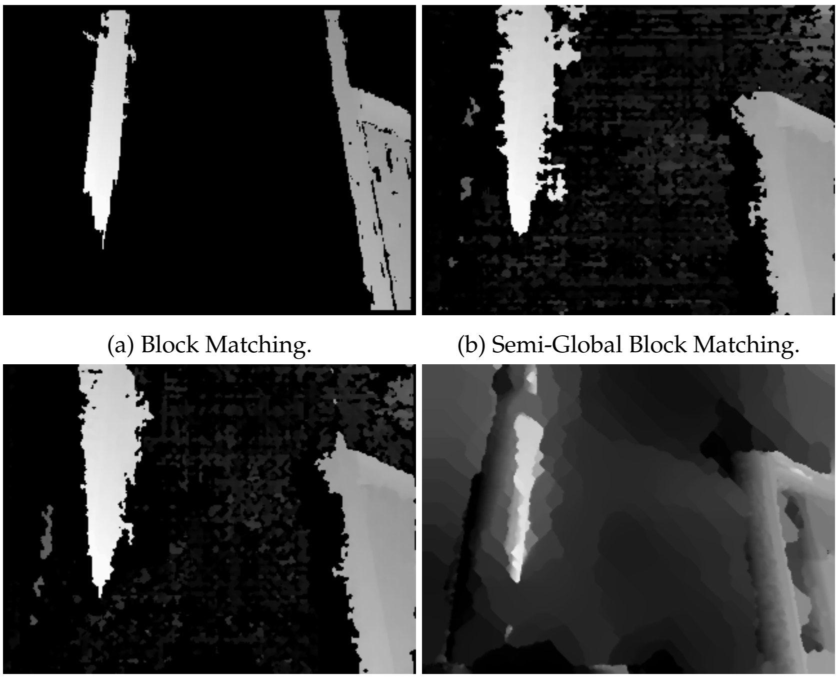
(a) Left (b) Right

Figure 30: Scene with ambiguous areas

Running each of the OpenCV-algorithms with this set produces results that are not very good. The disparity maps produced are presented in figure 31.

The simple block matching only gets the areas that are not ambiguous. Which can be to an advantage. By marking all the black segments as unreliable, a theoretical path planner could take this information into account when deciding where to go.

The semi-global block matching fares similarly, but both of them mark spots in the black segments with a wrong disparity level as the gray should be somewhere between the intensities of the segments it does get right.



(a) Block Matching. (b) Semi-Global Block Matching.

(c) Semi-Global Block Matching, full (d) Variational Stereo Matching

Figure 31: OpenCV-algorithms on ambiguous scene

Variational are perhaps the worst one as the ambiguous segments are wrongly marked as well as the non-ambiguous segments are also wrongly marked. Another reason to not using this algorithm for any proper solution.

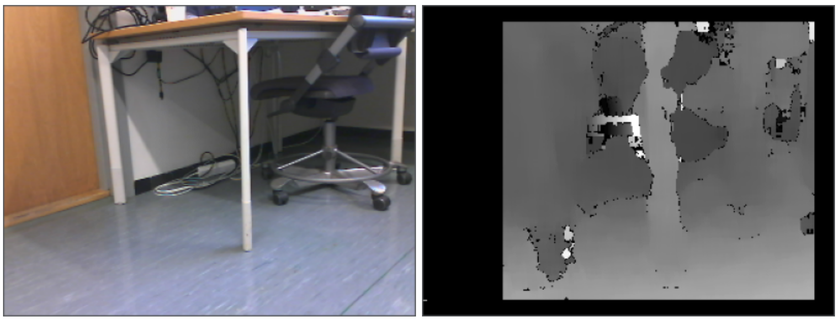

(a) Left image. (b) Wide leg.

Figure 32: Large SAD-window gives wide leg.

On a related issue. The effects of using a large SAD-window also gives interesting results on scenes with ambiguity. Figure 32 gives an example of this:

Here the disparity segment corresponding to the table's leg is quite much larger than the actual size which is a result from the large SAD-window used, 25, and the ambiguous area behind it. It would be best to avoid large SAD-windows where it is possible to avoid such results.

4.10 Improving performance by shaping workload

In this section other was of improving performance on the ARM-platforms will be evaluated.

4.10.1 Threading

Since it is more and more common to have multi-core CPUs, one way of speeding up the stereo matching algorithm by slicing up the image pair and then match each new pair instead. A visualization of this is depicted in figure 33 for parallelization using 4 threads.

For parallelization using SGBM, the process is as follows:

1. Retrieve an image pair.
2. Slice each image into n parts where n is the number of threads used.
3. For each slide run SGBM in its own thread to produce part of the disparity map.
4. Reassemble the slices to produce the final disparity map.

The increase in performance is theoretically linear, but there are many factors that can reduce the performance so in practice there is not such an improvement.

There are downsides to this approach as well. The other processes will be starved of resources if all the cores are busy with stereo matching.

4.10.2 Lower resolution

Accelerating the code through the various methods mentioned previously can and do give improvements, but for platforms like the Pandaboard they are not enough. So the last thing then is to make the workload small enough so that the platform becomes a viable one.

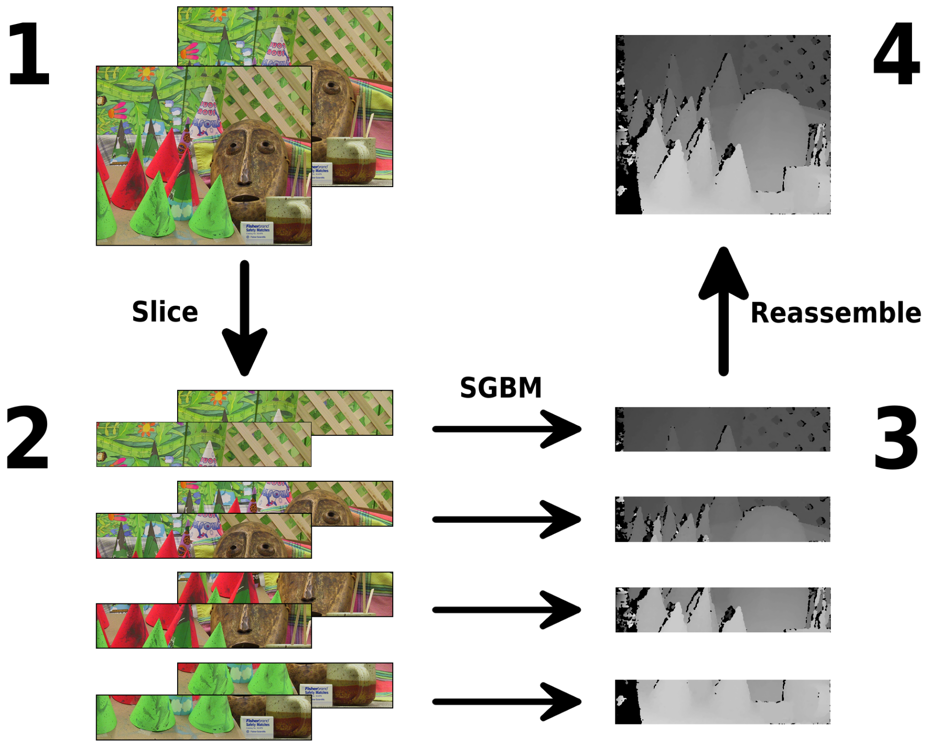


Figure 33: Threading visualized

Using the Cones set, a test were devised to test the possible speedup gained by scaling the data set down. The Cones set used earlier is 450 pixels wide and 375 pixels tall. From this pair, 25 new pairs were created each time reducing the height by 10 pixels while maintaining aspect ratio.

Using the NEON-enabled OpenCV-library to stereo match using the SGBM-method and running each pair 5 times to get an average running time. The speedup is calculated as follows in equation. 18:

$$\text{Improvement factor} = \frac{\text{Rate of reduced image (Hz)}}{\text{Rate of original image (Hz)}} \tag{18}$$

The factor of improvement is plotted against image height in figure 34.

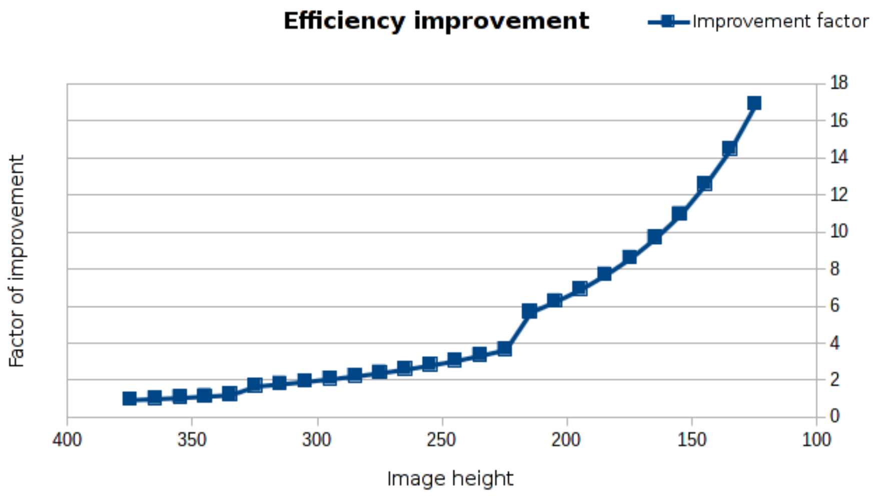


Figure 34: Improvement factor gained by using smaller resolution

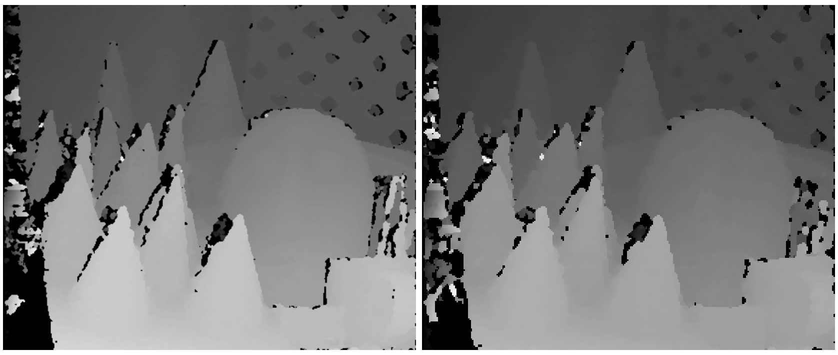
There are two results of special note here. The first is that by reducing the resolution enough, SGBM becomes viable on the Pandaboard and it is not necessary to shirk it all the way down to a third in height either.

The second interesting result here is the abrupt jumps from 335 to 325 pixels and from 225 to 215 pixels in height. Especially the last jump that is almost the double in performance improvement. Most likely these jumps are caused by fewer cache misses.

It is important here to remember that these results are specific to the Pandaboard running a self-compiled version of OpenCV. Other implementations will not necessarily give the same results. What is offered is instead a method for tuning said implementations so that “sweet spots” like the above can be found and taken advantage of.

There is one concern regarding this method and that is what detail is lost? To address this the Cones pair were resized to 288x240 pixels which is the same height as one of the resolutions supported by the Logitech cameras, 320x240 pixels. From the graph above this should be an improvement factor of just below 3.

A disparity map were produced from the 288x240 pixel pair and is presented in figure 35b alongside the disparity map produced using the original resolution in figure 35a.



(a) 450x375 px (b) 288x240 px

Figure 35: Original and resized resolution

Note the difference in brightness. This change reflects the change in distance in the corresponding pixels which is now less. Further note that all the main features of the disparity map is still there while the occluded areas are marginally reduced. As long as there is sufficient detail in the image pair used then sufficiently good disparity maps can still be constructed.

It should be noted that the aspect ratio of the camera resolution here is $3/4$ and not $6/5$. To get to the "sweet spot" a lower resolution can be selected or a region of interest can be used within the image pair.

4.11 Camera calibration

As mentioned earlier in the theory it is near impossible to create cameras without optical distortions. Some distortions are acceptable because there exists mathematical models that can be used to rectify the images taken.

OpenCV offers implementations for both measuring the distortions in a camera for correcting the image using the detected intrinsic and extrinsic properties of the camera.

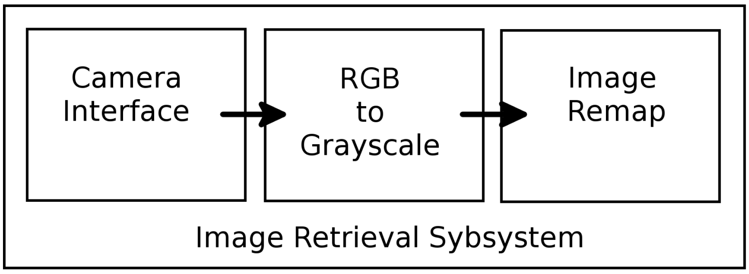


Figure 36: Image retrieval subsystem

4.11.1 Calibrating with OpenCV

The implementation uses a set of pictures with a known pattern. Different patterns are supported but in this thesis a 8x8 chessboard pattern is used. The chessboard in question is depicted in figure 37.

Then the corners are detected using OpenCV:

```
findChessboardCorners()
```

The results are shown here in figure 38:

This generates a set of matrices with the extrinsic and intrinsic properties of the cameras. In turn using this to rectify the image using OpenCV:

```
initUndistortRectifyMap()
```

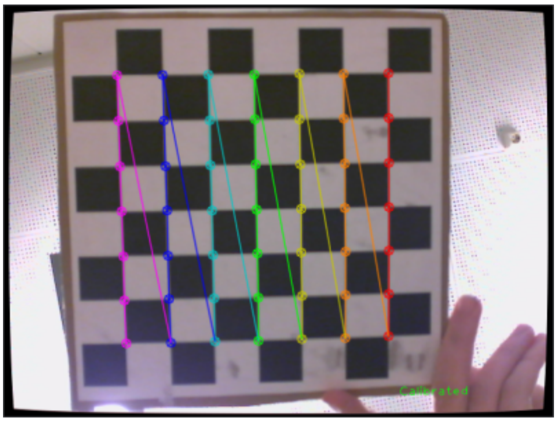


Figure 37: 8x8 chessboard used

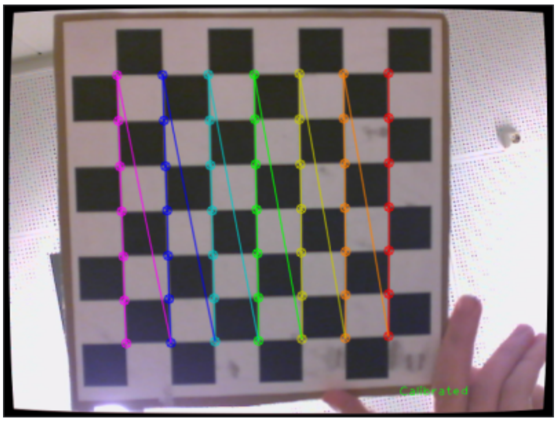


Figure 38: The detected corners marked

An example of the remapped image is found in figure 39

Notice that the spots align.

Although the method is not perfect and may need a few tries. An extreme example is presented in figure 40.

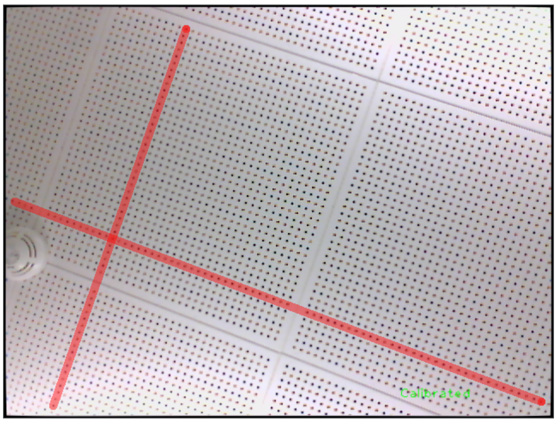


Figure 39: Calibrated spot of roof with repeating pattern

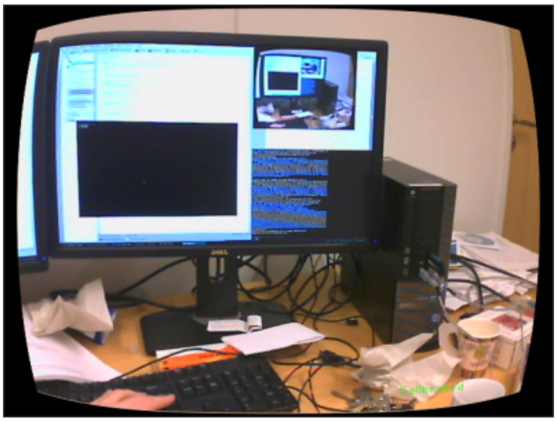


Figure 40: Wrongly calibrated camera

OpenCV does also provide calibration options for stereo cameras that in addition will provide information such as rotation matrix between the cameras coordinate systems and translation vector between the coordinate systems.

4.11.2 Rectification timings

Since rectifying the images is cost computational resources some effort were put into identifying the required cost. Using a resolution of 320x240 pixels and running on the Intel-platform, a running average of the rectification used about **7.5 ms**. This cost were deemed small compared to the stereo matching algorithm and therefore only a minor issue.

4.12 Collision detection and avoidance

Images of a scene accompanied with a disparity map is enough to begin rudimentary 3D-scene reconstruction. Extending this to a continuous stream of images and disparity maps, motion estimation of objects is possible. In turn more complex path planning is possible and estimation of collision course with other agents[2] can be avoided.

Clearly there is room for complex motion planning and this is the reason why the choice of algorithm for stereo matching is important. It is desirable to build upon this to construct a robust computer vision system module. However, advanced motion planning is outside the scope of this thesis as it is only intended to set the foundation for it.

Instead the simplest possible collision detection system is proposed; Is there anything closer than a set threshold? If yes, then stop.

1. Decide on a minimum safe distance.
2. Calculate the corresponding disparity level.
3. Threshold the disparity map at the selected disparity level.
4. Sum the remaining pixels as a filter to exclude the possibility of wrongly matched segments.

It should be clear that the quality of the disparity map is negligible, as long it is capable of detecting objects that are close to the cameras. The results in figure 22, made using variational stereo matching, have such bright spots and would perhaps not be a good candidate at all.

In this instance we can use simple block matching to create the disparity map. And thus enabling the evaluated ARM-based platforms for this specific scenario.

[2]Other entities acting acording their own set of rules

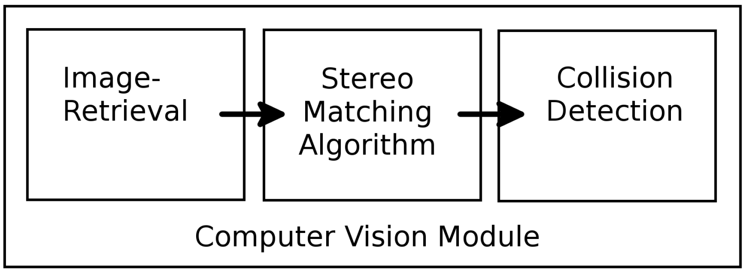


Figure 41: Module for computer vision with collision detection

Extending the collision detection

The described collision detection above is of little value as it does not provide anything beyond what a simple ultrasound based distance sensor would provide. To justify constructing a collision detection module based on computer vision, more is needed.

The most promising aspect of using computer vision is by visually sensing the surroundings like humans and estimating what will happen given the current and previous state. In a multi-agent environment it is necessary to act accordingly. For example a crowded street is filled with individual agent who all are able to maneuver to their destination by observing others and passively communicating their own trajectory simply by walking along an available route with minor modifications as they arise. By emulating such behavior, we would increase a robotss autonomy.

While creating the modules necessary for such autonomy is outside the scope of this thesis, instructions for how one might extend upon the foundation set is given instead.

Each segment in the disparity map represents at least one object or agent as the segments themselves are insufficient to determine the boundaries between possible objects and/or agents. Further steps to clearly identify this is needed and research into object recognition is suggested. By using a spherical coordinate system, the segment's disparity value can be used to find the distance, r, to the object or agent.

Using an edge detector like the Canny edge detector on the segments and fit a box around it. The coordinates of the center of the box is a good candidate for that object or agents coordinates, θ,ϕ.

To reduce the amount of objects this approach would identify, it is suggested to merge segments proximity and similar disparity level. This could potentially merge an agent with the surroundings so it is clearly not this straightforward.

Keep a list of identified agents together with coordinates from the last few seconds. Use this to extrapolate path and determine speed and in turn determine what steps, if any, is required to avoid any collision with any of the detected agents. Avoiding stationary objects should simply be left to the path planner.

While avoiding any situation where a possible collision may happen, it is important at the same time to have some assertive behavior or else risk being unable to perform.

5 Focused CV

While simply reducing the resolution of the image pairs gives considerable gains, a more refined approach is proposed in this section. An important observation:

- A scene usually changes little from frame to frame.

This fact can be taken advantage of in various circumstances. A few ideas are proposed in this section.

5.1 The stationary case

When standing still, each subsequent frame can be subtracted from the previous frame to produce a new frame when the non-zero portions represent a change since the point in time the last frame were taken. Because the observer is not moving, any movement must belong to other agents in the scene.

If it is an agent moving, there is a likelihood that the shape of the agent can be extracted from a disparity map of the scene given that one has been made. Should this be the case, then fitting a box around the old and new position of the agent and the new disparity of the agent can be found as well as any area that were previously occluded.

More complex situations where the temporal difference frame contains segments belonging to overlapping agents. It this case it becomes increasingly difficult to identify an optimal box to investigate. Either a box is encompassed around all the overlapping segments or an attempt at partitioning the problem into smaller problems can be attempted.

The boxes need to be wide enough to include the necessary information from each camera.

By computing the disparity map for these boxes, the old disparity map can be updated as all new information is accounted for.

5.2 The non-stationary case

When actively moving around there will constantly be changes in the robot's view. Simply subtracting one frame from the next will no longer work as there will be introduced changes over the whole angle of view. The only plausible segments that can be ruled out is the background and that is if there is no rotation in the movement.

However, given an disparity map and information about the movement, a estimation of how the disparity map should look like and thus also how the previous frame would look now. Subtracting this frame instead, along with some suppression of the errors introduced, there is a possibility that there is enough hints to successfully identify the other agents and minimize the needed work.

When rotating, there will be a percentage of the screen that cannot be estimated as the new part has been outside of the angle of view up until this moment. If the angle of rotation is known along with the angle of view, then it is possible to calculate how much of the disparity map the must be updated and the rest of the disparity map is shifted likewise.

A different approach would be to identify the background and static objects in order to reduce size of the problem.

5.3 Path-based focus vision with peripheral vision

Yet another suggestion is to keep a list of interesting objects and continuously updating this list. By fitting a box around the proposed path acquired from the path planner, a reduction i the computational cost can be had by only calculating the disparity map of these boxes.

The box should be expanded enough to compensate for the size of the robot to avoid finding non-traversable paths being suggested.

See figure 42 for a visualization.

The green box represent an object that is somehow in the path an therefore interesting to know where it is.

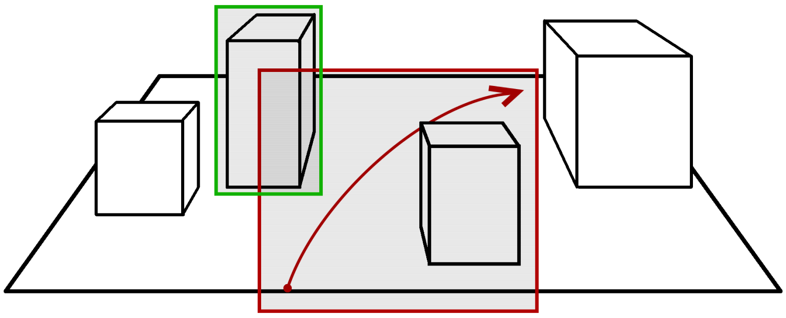


Figure 42: Path-based focus vision

The rest of the view should continuously subtracted from the next to create a frame with potential candidates for further inspection.

5.4 Deadline based feedback for movement control

Depending on the success of the methods described above, there might become too much work to do. By using soft deadlines for each disparity frame, there becomes a possibility of signaling the motor control to slow down or in the worst case to stop in order to the back into a state where enough is known about the scene in the angle of view to continue on without matching the whole view.

For example, sharp turns will reset everything that is known about the view and a new disparity map must be found. In such a situation it would be advisable to stop for a moment before continuing on.

6 Discussion

6.1 Obstacles

USB-camera wants whole USB-bandwidth. Solution is to use USB2/3-cameras So that more bandwidth is available. FireWire went out of style years ago and finding compatible hardware is difficult. In the end this proved to be a non issue as the algorithms could not handle the full resolution satisfactory.

6.2 Algorithmic evaluation

Considering the algorithms evaluated, only block matching and semi-global block matching were fast enough. Variational stereo match were too slow and the results were somewhat disappointing.

The energy minimization algorithms were interesting, but not suitable for the task at hand.

6.3 Software Evaluation

OpenCV

Pros	Cons
• Provides methods for most of the things needed to create a computer vision module.	• Little to no NEON support. This effectively makes ARM platforms without GPU-acceleration nonviable.
• Useful data structures available. Less code needed to create such things and more time for experimentation.	• OpenCV provides APIs for C, C++, Java and Python. Or at least that is what the documentation would have you believe. What is less obvious is that close too all development is being done in C++ and features are not ported back to the other APIs. This is problematic when code that once run, no longer runs. Code that should compile no longer compiles or even compile in the first place. This could easily be avoided by marking the outdated APIs as deprecated, but the current politics wants to keep them.
• Optimized for modern x86-processors thus enabling this sort of work to even be possible to do without specialized hardware.	
• Algorithms give good enough. According to the middlebury research site there have been published "better" algorithms that give less erroneous pixels compared to ground truth. If the MRF-library should be any indication then it is unlikely that any of these algorithms have optimized implementations that would enable them to be used for real-time purposes. As such they are of little interest.	
• Not researched in detail, but rudimentary support for GPU algorithms have been implemented.	

Table 20: Evaluation of the OpenCV computer vision library

Middlebury MRF

Pros	**Cons**
• For the most part the source for advanced stereo vision algorithms.	• Disappointing results. • Impractically slow. • The README suggests that some optimization has been done. However not enough to make the library useful for real time performance. • According to the README the library was updated in 2012 in order to make it compile on 64 bit platforms and reduce warnings produced by GCC 4.6. Otherwise code suggests that most work were done in 2006 and it is unlikely that any meaningful further development will take place. It seems that the library is meant for research purposes and not real time is not a criterion. Which is understandable. • Did not compile on ARMv7. No effort were done to investigate this as it was already concluded that the library was not fit for this task.

Table 21: Evaluation of the Middlebury MRF library

Arch Linux (x86_86 and ARMv7)

Pros	Cons
• Consistent environment on all platforms. • Provides resent packages of programs used on platforms. • Hard floating point ABI is enabled by default on ARMv7. • A large selection of ARM based development boards are supported.	• Obscure and choice of distribution is biased

Table 22: Evaluation of the Arch Linux operating system

Angstrom Linux

Pros	Cons
• Unlike other Linux distributions, Angstrom does not change rapidly.	• Hard floating point ABI is not enabled by default. The performance hit is devastating if the algorithm of choice is dependent on floating point performance.

Table 23: Evaluation of the Angstrom operating system

6.4 Hardware Evaluation

Logitech C250 Web Camera.

Pros	Cons
• Surprisingly little lens distortion. • Low prototyping costs.	• Limited bandwidth on the USB-controller. Could not use full resolution. • MJPEG stream. Lossy compression and CPU-time needed to decode the stream resulting in another performance hit.

Table 24: Evaluation of the cameras used

PandaBoard

Pros	Cons
• Hard floating point. • NEON SIMD instructions.	• Nearing 5 years old. Not representative of current hardware options. • Despite having a GPU on die, no suitable driver is offered by Texas Instruments. • Relatively large and has many peripherals not needed. • No on-board storage options and needs SD-cards. Not a very robust solution although makes for easy prototyping.

Table 25: Evaluation of the Pandaboard

BeagleBone Black

Pros	Cons
• Faster than PandaBoard despite both being ARMv7 based and clocked at 1 GHz. Tests showed 40% increase. • Physically smaller. • On-board storage. No need for SD-cards.	• Very restrictive on peripherals. Only 1 USB-port so a USB-hub is at least needed for the implementation in this thesis. • Comes with Angstrom Linux. Not a very enticing Linux variant.

Table 26: Evaluation of the BeagleBone Black

Compared to each other, BeagleBone Black is the superior choice.

Intel platform
The vast performance difference gives an overly optimistic view of the viability of the evaluated algorithms.

6.5 Future Work

the stereo camera constructed should be viewed as a simple toy. A better one should be made that is tailored to the specific application in mind.

ARM as a platform is barely viable if the steps mentioned earlier is taken. It would be more interesting to see what GPU-acceleration could bring but for now there is little code to test.

OpenCV might get NEON acceleration, but there is no indication of this at the moment.

After a suitable solution for the algorithmic challenges has been found, an environment-aware path-planner with collision avoidance capabilities should be made.

A higher level CSP-style way for communication with other modules would be best. Easier integration. Up to the constructors of this system to decide.

6.6 Recommendations

Software
Besides the obvious flaws, OpenCV is good enough. If the flaws are mended or worked around in a compatible way, (Like using OpenCL, writing NEON support or choosing a better ARM-device), then it is this thesis recommendation that OpenCV is used. OpenCV sort of wins by being the only viable contender.

It is not recommended writing a separate library as an alternative to OpenCV. Such a task is fairly large and complex. Furthermore this has already been done in the form of UncannyCV. For it to be worth it, such a library must outperform OpenCV and at least match UncannyCV.

SGBM(partial and full) gives good enough results. SGBM is at least recommended. Should resources be available for performing the full SGBM then this is recommended as well.

The author is convinced that GPGPU is the only reasonable way forward. Creating disparity maps is a massively parallel problem and CPU-time is better spent on other aspects of robot software like motor control and decision making. This thesis recommends OpenCL over SIMD optimization. (Cuda is too hardware specific.)

Hardware
BeagleBone Black is not an option unless the software situation changes. The same goes for the PandaBoard.

If one should choose an ARM-based solution then a platform should be chosen that has a decent community behind it and proper OpenCL drivers available. On board storage is a must since SD-card are not very rugged.

Low power x86 is an alternative. Intel reports that Haswell-based solutions can be found in as low power as 4.5 W. Both Intel and AMD does a far better job at supporting Linux than any of the ARM-licensees.

No recommendation with regard to choice of cameras. This needs to be decided according to the needs and possibilities in each individual project. That said there exists stereo cameras that are far better than what was constructed for this thesis or the ones chosen in for example eurobot 2010

6.7 Similar work, OpenVX

Is the work done in this thesis the right approach to increasing autonomy in robots? Well, as a case in point, Khronos announced a new standard called OpenVX during this thesis work. It is a standardization for computer vision describing itself as complementary to OpenCV yet describes OpenCV as best used for rapid prototyping, while itself is suited for production deployment. The fact that OpenVX is a standard validates the approach taken with this thesis.

On the negative side there is only a specification and no implementations available yet. This is likely to change.

7 Conclusion

A wide selection of stereo matching algorithms have been evaluated for the purpose of creating a collision avoidance module. Varying greatly in the accuracy, a few of the algorithms were fast enough for further use.

Two computer vision libraries, OpenCV and MRF, were evaluated for their implementations of various stereo matching algorithms. In addition OpenCV provides a wide variety of functions for creating sophisticated computer vision programs and were evaluated on this basis as well.

A stereo camera were constructed using low cost, of-the-shelf web cameras by Logitech.

Two low-power platforms, The Pandaboard and the Beaglebone Black, were evaluated as viable platforms for developing a computer vision module on top. In addition they were compared to an Intel platform as a reference.

Considerable efforts were conducted into elevating the low-power platforms to a state were a collision avoidance module could be deployed.

Based on the results gathered, a fast, but simple, collision detector could be made using the simple block matching algorithm found in OpenCV. A more advanced detector could be built using semi-global stereo matching. These were the only implementations that were fast enough. The other energy minimization algorithms (Graph cuts and belief propagation) did produce good disparity maps, but were too slow for any realistic collision detector.

OpenCV as a library contains a very large set of functionality which were more than enough to construct rudimentary demos to demonstrate the potential for a collision detection module. However there are major drawbacks present. OpenCV is optimized for x86-instruction sets like SSE2 which the low-power platforms cannot take advantage of. OpenCV has APIs for other languages than C++ which it is written in, but all features are not necessarily ported to these APIs which can be a frustrating experience.

In order for the low-power platforms to be fast enough, a combination of improvements must be used. OpenCV should be compiled with aggressive optimization options enabled with support for hardware accelerated floating point math. Choice of low-power platform matters, but compensations can be made elsewhere.

The most effective speedup that enables the low-power platforms were reducing the resolution of the images to be matched. When reducing the size of the sub-problems enough to align with cache size, considerable speedups were found with little penalty in the corresponding disparity map.

There were not enough time to construct a proper collision avoidance module, but the necessary groundwork is laid to be built upon.

References

[BK04] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(9):1124–1137, September 2004.

[BVZ01] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222–1239, November 2001.

[Can86] J Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, June 1986.

[End10] Kai Hugo Hustoft Endresen. Tracking objects in 3D using Stereo Vision. Master's thesis, Norwegian University of Science and Technology, Tronheim, Norway, 2010.

[FARW99] R.B. Fisher, A. P. Ashbrook, C. Robertson, and N. Werghi. A low-cost range finder using a visually located, structured light source. In *3-D Digital Imaging and Modeling, 1999. Proceedings. Second International Conference on*, pages 24–33, 1999.

[FH04] P.F. Felzenszwalb and D.P. Huttenlocher. Efficient belief propagation for early vision. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–261–I–268 Vol.1, June 2004.

[HD92] Y. Hu and T.J. Dennis. Simulated annealing and iterated conditional modes with selective and confidence enhanced update schemes. In *Computer-Based Medical Systems, 1992. Proceedings., Fifth Annual IEEE Symposium on*, pages 257–264, Jun 1992.

[Hir05] Heiko Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2 - Volume 02*, CVPR '05, pages 807–814, Washington, DC, USA, 2005. IEEE Computer Society.

[HS07] H. Hirschmuller and D. Scharstein. Evaluation of cost functions for stereo matching. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, 2007.

[Kol06] Vladimir Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(10):1568–1583, October 2006.

[KZ02] Vladimir Kolmogorov and Ramin Zabih. What energy functions can be minimized via graph cuts? In *Proceedings of the 7th European Conference on Computer Vision-Part III*, ECCV '02, pages 65–81, London, UK, UK, 2002. Springer-Verlag.

[Rø13] John Magne Røde. Continous Calibration of 3D Vision Systems. Master's thesis, Norwegian University of Science and Technology, Tronheim, Norway, 2013.

[SP07] D. Scharstein and Chris Pal. Learning conditional random fields for stereo. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, 2007.

[SS03] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–195–I–202 vol.1, 2003.

[SSZ01] D. Scharstein, R. Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In *Stereo and Multi-Baseline Vision, 2001. (SMBV 2001). Proceedings. IEEE Workshop on*, pages 131–140, 2001.

[SZS$^+$08] Richard Szeliski, Ramin Zabih, Daniel Scharstein, Olga Veksler, Vladimir Kolmogorov, Aseem Agarwala, Marshall Tappen, and Carsten Rother. A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(6):1068–1080, June 2008.

[TF03] Marshall F. Tappen and William T. Freeman. Comparison of graph cuts with belief propagation for stereo, using identical mrf parameters. In *Proceedings of the Ninth IEEE International Conference on*

Computer Vision - Volume 2, ICCV '03, pages 900–, Washington, DC, USA, 2003. IEEE Computer Society.

[WJW05] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. Map estimation via agreement on trees: Message-passing and linear programming. *IEEE Trans. Inf. Theor.*, 51(11):3697–3717, November 2005.

[ZLDH11] Wenqiao Zhu, Dongming Lu, Changyu Diao, and Jingzhou Huang. Variational stereo matching with left right consistency constraint. In *Soft Computing and Pattern Recognition (SoCPaR), 2011 International Conference of*, pages 222–226, Oct 2011.

A Contents of project

Calibration

Image files needed for calibration.

Code

Various code demoes and examples.

Papers

Background litterature.

Results

Where all the results are gathered.