Mälardalen University Press Licentiate Thesis
No.142

Resource Optimized Stereo Matching in Reconfigurable Hardware for Autonomous Systems

Fredrik Ekstrand

September 2011

School of Innovation, Design and Engineering
Mälardalen University
Västerås, Sweden

ISSN 1651-9256
ISBN 978-91-7485-035-2
Printed by Mälardalen University, Västerås, Sweden

Abstract

There is a need for compact, high-speed, and low-power vision systems for enabling real-time mobile autonomous applications. One approach to achieve this is to implement the low- to intermediate-level applications in hardware. Reconfigurable hardware have all these qualities without the limitation of fixed functionality that accompanies application-specific circuits. Resource constraints in reconfigurable hardware calls for resource optimized implementations with maintained performance.

The research group in Robotics at Mälardalens University is moving toward the completion of a reconfigurable hardware-platform for stereo vision, coupled with a compact embedded computer. This system will incorporate hardware-based preprocessing components enabling visual perception for autonomous machines. This thesis covers the reconfigurable hardware section of the vision system concerning the realization of scene depth extraction. It shows the advantages of image preprocessing in hardware and propose a resource optimized approach to stereo matching. The work quantifies the impact of reduced resource utilization and a desire for increased accuracy in disparity estimation. The implemented stereo matching approach performs on par with recent similar implementations in terms of accuracy, but excels in terms of resource utilization and resource sharing, as the external memory requirement is removed for larger images.

Future work aims to further include processes for navigation, and structure and object recognition. Furthermore, the system will be adapted to real world scenarios, both indoors and outdoors.

Acknowledgements

This thesis could not have been done without the support of my supervisors Lars Asplund, Mikael Ekström and Giacomo Spampinato. Thank you for believing in me. Your supervision has brought me a long way toward the realization of an independent researcher.

Many thanks to Carl Ahlberg, Jörgen Lidholm, and Batu Akan for all the deep discussions, provoking ideas, crazy stunts and laughs. A special thank you to Carl and Jörgen for never backing away from assisting and helping, as co-authors and as friends.

A warm thank you to Hüseyin Aysan for all the laughs, all the fika, and for always giving a helping hand and being a great friend.

MDH is a place of work and study, but even more than that, it is a place of interaction with and learning from a body of diverse knowledge and experience, manifested in many joyous conversations. Thank you Nikola, Martin, Marcus, Jimmie, Andreas, Mikael Å, Farhang, Moris, Mirko, Jenny, Carola, Susanne, Malin, Ingrid and all the rest for always answering with a smile. An additional thanks to Martin, Nikola, Mikael, Marcus and Giacomo for never hesitating to assist and help. And a big thanks to Adnan without whom this thesis would have never materialized until this day.

Last, but certainly not least, a huge thank you to my family! My precious sons William and Edward who have supplied me with drawings to enhance my office, questions to occupy my mind, and unconditional love. A huge thank you to my fabulous wife Sara for your support, dedication, devotion and love! I could not have done it without you.

Fredrik Ekstrand
Västerås, September 26, 2011

List of Publications

Papers Included in the Licentiate Thesis[1]

Paper A *Two Camera System for Robot Applications; Navigation*, Jörgen Lidholm, Fredrik Ekstrand and Lars Asplund, In proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Hamburg, Germany, September 2008

Paper B *Resource Limited Hardware-based Stereo Matching for High-Speed Vision System*, Fredrik Ekstrand, Carl Ahlberg, Mikael Ekström, Lars Asplund and Giacomo Spampinato, In proceedings of the 5th International Conference on Automation Robotics and Applications (ICARA), Wellington, New Zealand, December 2011 (*to appear*)

Paper C *Utilization and Performance Considerations in Resource Optimized Stereo Matching for Real-Time Reconfigurable Hardware*, Fredrik Ekstrand, Carl Ahlberg, Mikael Ekström, Lars Asplund and Giacomo Spampinato, Technical Report

[1] The included articles are reformatted to comply with the licentiate thesis specifications

Other relevant publications

Conferences

- *Towards Binocular Realtime Object Recognition - A Work in Progress*, Fredrik Ekstrand, Jörgen Lidholm, Giacomo Spampinato and Lars Asplund, Presented at the International Conference on Machine Vision, Image Processing, and Pattern Analysis (MVIPPA), Bangkok, Thailand, December, 2009

- *Robotics for SMEs - 3D Vision in Real-Time for Navigation and Object Recognition*, Fredrik Ekstrand, Jörgen Lidholm and Lars Asplund, Presented at the 39th International Symposium on Robotics (ISR), Seoul, South Korea, October, 2008

Contents

I

Thesis

Chapter 1

Introduction

Self-parking cars, pedestrian-sensitive self-braking trucks, driver-less mining-machines, and museum guiding-robots are all examples of intelligent and autonomous agents. Autonomous agents are entities that are assigned a task and execute it without further guidance or interference from the task originator. Such an agent senses its environment, adopts an approach accordingly, and executes an action toward the fulfillment of the task. Those are the same fundamentals which form the definition of a robot: sense, plan, and act.

Spatial awareness is elementary in any autonomous mobile machine. There are two fundamentals in the concept of spatial awareness - a knowledge of the environment, and one's own relation to that environment. (It can be argued that an autonomous agent is really not in an environment, but part of the environment.) Knowledge of the environment requires sensors, and the degree of perception is determined by the properties of the sensors. Regardless of sensor type, the resolution, accuracy and speed of the sensor limit the awareness. There are several types of sensors for sensing the surrounding space, but the two predominant types used in robotics and industry are rangefinders and vision.

Rangefinders (such as radar, sonar and laser) are active systems that emit waves (such as electromagnetic or light) and then measure the reflections of the waves off an object. Vision sensors, or cameras, are passive sensors that measure whatever light that falls onto the sensor, whether direct or reflected. Common for rangefinders is that they are not as fast as passive visual systems as they rely on returned waves, whereas cameras can measure at much shorter intervals as the light flow is one-directional and constant. These passive and

general properties make cameras versatile, but also limited in the application of range finding as they lack the built-in ranging property of rangefinders. Without knowledge of the temporal origin of the measured light, cameras cannot use the time-of-flight or accumulation methods used by the range-finders, but need to correlate the sensed data over a spatial difference. This is commonly performed by triangulation of views from different angles of the same scene, either by a movement of a single camera or by the use of multiple cameras, referred to as stereo vision.

All types of rangefinders are well suited for map generation and obstacle detection, but they are not optimal for object recognition, or tracking, as they only convey the structure of the surroundings, and nothing about its colors or patterns. Stereo-vision systems are an approximation of human eyes and can enable machines to match or relate to our perception of the world. All information about the environment exist in the data generated by the cameras. It simply needs to be extracted. This simple part has occupied the large computer vision research community for many years, and still do.

In this thesis, we present a stereo vision system for embedded mobile robotics. The end goal with this research platform is to fit a real-time autonomous system for navigation and object recognition in a compact and power efficient hardware system. In order to fit all system parts, each component must be made as compact and efficient as possible. This thesis focuses on reducing the task of extracting depth information of a scene through matching of view-separated images.

1.1 Background

Computer vision involves digital processing of images. Images are captured with a sensor measuring the light falling onto the sensor surface. The amount of light is transformed into a digital representation which is communicated off the sensor. The quality of an image is contingent on the sensor architecture, the lens, the converter electronics, the circuit board design, and many more factors. A great deal of research is dedicated to improving the performance of image sensors. Our research is focused on the application of the image sensor, and the process of extracting the information embedded in the sensor data. Many applications and algorithms exist for image processing, and those concerned with using the images to enable machines to see are labeled as belonging to

machine, or computer, vision.

Computer vision algorithms can generally be characterized by complex and repetitive operations, and large amounts of data, as detailed by Ratha and Jain [1]. Moreover, vision algorithms can be classified as belonging to either low-level, intermediate-level (segmentation), or high-level (higher order structure and matching). Regardless of level, vision algorithms are all preprocessing steps for a main algorithm, such as navigation or object recognition, but the separation is far from distinct. A complete vision system needs to integrate solutions for all levels in order to complete the main application. In this thesis, we are concerned with low-level algorithms.

By definition, the performance of a system is contingent on the performance of its parts. Being the initial node in the chain, the sensors set the performance limit. Image sensors can provide high frame rates, but require the receivers of their pixel stream to match their speed. If a receiver is to receive and process images continuously, it needs to be able to both receive and execute operations on every pixel in time before the next pixel arrives. This implies an operating frequency several times higher than the pixel frequency of the image sensor. Real-time image processing requires reading and operating on millions of pixels per second, putting a hard requirement on the throughput ability of the processing system. The concept of real-time will vary with the topic, and by real-time we mean the execution time of an action or reaction that is adequate to mimic the human counterpart. Concerning cameras, a frame rate of around 30 Hz is sufficient to not appear jerky to the human eye at moderate transitions in the scene. For completely smooth motions an update frequency of above 60 frames per second is required. We use 30 frames per second as the frame rate definition of real-time.

1.1.1 Reconfigurable Hardware

As opposed to standard sequential computer systems, which require a processing frequency several orders of magnitude greater than the pixel frequency, reconfigurable hardware enables pixel-wise image processing at a frequency matching that of the pixels. Reconfigurable hardware, such as FPGAs, is a hardware component where the functionality is loaded at startup. The central processing unit of a typical PC fetches its instructions from memory, executes them, and then stores the result back into memory. In the FPGA, the physical configuration itself is the instructions and there is no setup delay [1]. It is a standalone component that executes like a fixed state machine without an operating system or external components. The big advantage of FPGAs is that

they enable concurrent processing of multiple data by parallelization. This removes the need for the processing unit to operate at a frequency above that of the vision sensor.

An important parameter of reconfigurable hardware is the limited available resources. Being a component of fixed size where the functionality is determined by the physical interconnect of its logic elements, only a certain amount of instructions can be concurrently realized. Moreover, shifting of instruction sets is not possible as resources cannot be reallocated during run-time. In other words, FPGAs can get full [1].

The type of algorithms appropriate for FPGAs are also limited due to the types of operations possible with the internal circuitry. Any type of operation can be realized in theory, but the cost in doing so might render it impractical. Registers, comparators, adders, multipliers, and internal memory are all in finite numbers and realizing complex algorithms might require more than available. Implementations of algorithms thus have to fit both in type and size. The functionality of an FPGA is described with code written in a Hardware Description Language (HDL) such as VHDL or Verilog. FPGAs are easily reconfigured using tools ranging from low-level programming languages, such as HDL, to more general languages, such as variants of C and Python.

1.1.2 Feature detectors

In certain applications, such as navigation and object recognition, limited parts of an image is often of more interest than the rest. These parts are features of an object or a scene, and can be used as descriptors for that object or scene. Algorithms identifying and extracting these defining parts of an image is referred to as Feature Detectors. Different feature detectors are good for different applications, but their common task is to identify salient areas (areas with low similarity in the surrounding area), such as edges, corners, blobs, etc. Their primary function is to reduce the amount of data associated with an object or scene, without sacrificing the important information. One of the most important properties of a feature detector is its repeatability: the ability to repeatedly identify the same feature on any two separate occasions. This ability is crucial when locating features between multiple images, as in matching for tracking, depth, shape, etc. Another important parameter is the information content of a feature detector, a measure of the distinctiveness of a salient point. The more spread out the features are over an object, the higher the information content, and the higher the likelihood of a successful match [2].

A multitude of feature detectors exists, and in Paper A the Stephen and Har-

ris Combined Corner and Edge Detector [3] is used. It has been widely used in computer vision applications for a long time, due to its high repeatability and information content [4]. The Stephen and Harris detector, also common in many other feature detectors, looks at the intensity of each pixel and how it relates to that of its neighbors. A pixel is evaluated based on how well it matches the defined feature types - sharp discontinuity in one direction equals an edge, and in two or more directions equals a corner. The better the match, the higher the absolute cornerness value (positive for corners, negative for edges). The algorithm produces only this definition of a feature, which makes a feature-to-feature correlation challenging. Although similar in fashion, the edges and corners have one small difference: corners are by definition isolated objects not linked to other corners, whereas edges have a stronger relation to other edges and can be formed into lines or curves possible to use for matching [5].

1.1.3 Feature Matching

Matching of individual pixels based solely on their intensity is an almost impossible task. Performing the same operation on corners or edges from the Stephen and Harris detector can be less difficult, but is very scene and parameter dependent as the amount of features impact the matching confidence. Finding a single point from one image in another image of thousands, or even only hundreds, of points with only a single value to compare, is not trivial. An approach is to look at several features and their individual relations, and match them as a point cloud [6]. Such operations are highly iterative, and not suitable for a resource constrained real-time system.

To reduce the challenge of correlation, it is possible to increase the feature uniqueness by including more properties of the feature and its surroundings, such as angle or scale. This property specification adds descriptors to the features, such that it is possible to look at the feature descriptors individually and not simply at their mutual relation. A good example of a feature descriptor is SIFT (Scale Invariant Feature Transform) [7]. However, the added descriptiveness is computationally intense and of an iterative nature, and the matching process can be very time consuming for extensive feature sets [8].

Matching of non-aligned images, irrespectively of whether based on individual points or areas, require a costly 2-dimensional search across the other image for every element. The remedy is to transform the images into the same coordinate system, a process called rectification. Rectification involves identifying the intrinsic and extrinsic parameters of the image capturing device to determine the relation of the projection planes of respective images. This in-

cludes correction of lens distortion, and aligning the images so that image scanlines are parallel and aligned between images. The matching problem is thus reduced to a 1-dimensional search, significantly reducing the complexity, as long as the geometric distortion is at a minimum [9]. The rectification process is computationally heavy, and needs to be performed for every image pair of unknown relation. For fixed stereo camera systems, the calculation of the rectification parameters need only be performed once as the parameters of the capturing devices are static. Rectification is then performed by image transformation through applying a constant set of parameter-based coordinate shifts.

The concept of extensive feature descriptors, such as SIFT, is to include more than just the saliency of the point, and also include additional information on the neighborhood, such as qualities of other salient parts (edges) in the area, the saliency at different scales, etc. The reason is obvious, identification is easier the more information available. This notion can be applied to the underlying pixels directly, without performing an analysis of their properties. Area-based approaches match an area instead of a point, and they are the most used approach to stereo matching in computer vision.

1.1.4 Stereo Matching

The area of computer vision contains many branches, and stereo matching, or stereo correspondence, is one of the widest. It deals with extracting depth information from 2-dimensional images by way of finding corresponding points in two, or more, images. The sole purpose of using two cameras is to capture a scene from two different views at any given time in order to extract 3-dimensional data of the scene.

Any vision approach concerned with depth needs to solve the correspondence problem, that is, which part in one image correlates to which part in another image. In the machine vision community, the majority of approaches can be categorized into either of two groups, global or local [10]. In general terms, the global algorithms are considering the estimation of the separation, or disparity, of the two view-diverging images as an optimization problem. A global cost function incorporates both data (matching) and smoothness terms, which the disparity selection seeks to minimize. Local algorithms, on the other hand, only consider a limited area surrounding the point under evaluation for disparity estimation.

Global methods generally outperform local methods in terms of accuracy, but suffer from a high computational cost. Global methods usually consist of several, often iterative, steps in their refinement of an initial disparity map,

often attained with a local method [10]. As a consequence, they are not optimal for real-time applications.

Local methods can be further divided into area-based or feature-based correlation. Both are sprung out of the same basic notion - a pixel in itself gives poor correlation data with low confidence in matching, thus a larger view is required. Both approaches use the neighbors, to more or less extent, of the current pixel for more defining data. Area-based methods use them to correlate with another same-size area, whereas feature-based methods use them to determine the interest level of the pixel and use that rather than the underlying image data.

Area-based matching techniques usually create a dense map with depth information for every pixel. Feature-based techniques can only create a sparse map as information is removed from the images. However, it is argued that the confidence in the match is higher with feature-based techniques as they are only matching on individual pixels, rather than a set of pixels [9]. Nevertheless, which technique to use should be based on the application.

Feature-based matching techniques are more concerned with finding a relation to the scene or image as a whole than to get a complete 3-dimensional reconstruction of the scene. They can be used, for example, to determine the ego-motion of the agent, or to correctly identify the rotation and translation of an object. Working with feature images also significantly reduces the amount of data in the system, leaving room for additional calculations or an increased frequency. Thus, for applications not in need of depth information in the whole scene but rather high speed, such as certain object recognition [11], the feature-based approach is a good candidate. An additional advantage is that a crystallization of the important information in the lower-level can both reduce the amount of data as well as its rate. The data rate reduction is advantageous for higher level processes, but only if the data is sufficient.

1.1.5 Area-based Matching

Area-based methods correlate the entire pixel neighborhood, element by element, through the use of a support window. The support window is compared with same size support windows in the other image, and is usually in the form of a square. To evaluate the similarities of two windows, a correlation measure is required. Several exist, but one of the simplest and most straightforward to implement, and thus widely used, is the SAD (*Sum of Absolute Differences*). With the SAD, the matching cost for two points residing in two different images is calculated through an aggregation of the element-wise absolute differ-

ences of the support windows for respective point.

One of the fundamental problems of window-matching is the selection of the window-size [12]. A small window achieves higher precision in the disparity estimation, but exhibits more noise. Large windows reduce noise by an increase of the matching data, but reduce the precision, especially at depth discontinuities. Thus, the optimal window size will vary from scene to scene, but also within a scene.

Several approaches have been proposed to solve the size selection problem. Variable-size windows, as proposed by Kanade and Okutomi [12], are adapting to the conditions of the underlying image and have been shown to significantly improve the matching, but lack in terms of speed. This idea have been refined to variable window shapes, as presented by Mei et al. [13] and weighting of the support window, as proposed by Yoon et al. [14], to only consider information on similar data, such as color. All these approaches strive to improve the outcome of the matching algorithm, the generated disparity map.

1.1.6 Disparity Map Creation

The role of the disparity map is to convey the depth in an image represented as the distance of the index of a certain point between two images. The matching algorithm will approximate the real-world depth relation for the entire image, but hard-to-match areas of the image, such as those of low texture or low signal-to-noise ratio, will generate false matches. Additionally, foreground objects occlude background objects, and due to the different perspectives in the two images, the parts that are occluded will differ in the two images. This causes pixels adjacent to object borders, or depth discontinuities, to be estimated at the depth of the foreground object as the edge is a very prominent feature. This causes the disparity maps to extend outside of the foreground object, and is called foreground fattening. The inadequacies of the area-based approaches limit the possible quality of the disparity map, and several approaches have been proposed to deal with this.

Approaches seeking to create dense disparity maps try to remedy the deficiencies, whereas those aiming for a sparse but highly confident disparity map simply remove them. Regardless of the approach, the initial step is to identify the erroneous values, which can be done using as set of assumptions about the underlying image. They act as constraints on the disparity map, and can be used to determine the validity of a match, as explained by Ozanian and Takouhi [15].

The surface continuity constraint states that a scene is made up of solid sur-

faces which vary smoothly. As a consequence, adjacent pixels are most likely at the same depth. The uniqueness constraint states that a point in one image can only have one corresponding point in the other image, which is natural as the images depict physical objects. The ordering constraint states that the order of pixels in one image must be fulfilled in the disparity map. Violations of these constraints occur, for instance, at depth discontinuities, heavily slanted surfaces, and occlusion. However, for the most part they can be used to validate the estimated disparity of a pixel in rectified images.

One of the common ways of finding these violations is to perform a left-right consistency check (LRC) [16]. A regular matching procedure uses one of the images as the base and then tries to find corresponding pixels in the other image. Pixels that have no corresponding mate, as they are not visible in the other image, will generate false matches. The LRC also performs matching with the other image as the base and then checks to see that a pixel indicated as the match in one image is referring back to the indicating pixel in the other image, that is, that they select each other as the best match. This is a very robust method that identifies the majority of false matches due to perspective distortion [17].

After false matches are identified, sparse approaches just discard them and leave the pixels void of disparity. Dense approaches need to assign a value though, and the constraints mentioned earlier can also be utilized for this purpose. Instead of estimating the disparity by correlation, similarity in adjacent pixels, which are assumed to be of same surface according to above constraints, can approximate the disparity. A popular method is to use median-filtering to remove noise and smooth the disparity map. As surfaces are more likely to be smooth than bumpy, this increases the quality of the map. Another approach is to interpolate or propagate values from surrounding pixels to fill in empty areas.

The quality, or correctness, of a disparity map is assessed through comparison with the scene ground truth. A set of stereo image pairs were proposed by Scharstein and Szeliski [10] and they are used as the benchmark of correspondence approaches today, with tools available online [18].

1.2 Motivation

Reducing the workload in a visual perception application can be achieved in two ways: reduce the amount of data by only sending data of interest to the application, or extract necessary information so that the receiver only needs

to consume, not process. The first scenario is realized with a feature detector, and the second with any of several transforms: depth extraction, segmentation, object identification, etc.

The initial project with our stereo camera system was to produce a fast navigation application capable of simultaneous localization and mapping (SLAM) through the use of vision [6],[19]. In short, SLAM is a process where an agent enters an unknown environment, picks out identifying landmarks or geometries so that it can move around and always find its way back to the starting point with the help of the identified visual cues. As the agent traverses the environment, it continuously builds a map of the environment which it later uses for navigation.

Common approaches are to use the SIFT [7] or SURF [20] feature descriptors for landmark matching. The biggest challenge of SLAM is to identify salient areas with high confidence in the estimated depth. The SIFT approaches rely on unique identifiers which is slow and/or large in implementation. Simpler feature detectors can be made faster, but lose in matching confidence. However, a lack of accuracy might be compensated with higher frequency. We thus opted for a fast but less accurate approach in an attempt to reduce the computational complexity.

To improve the accuracy of the initial approach, we then propose a concurrent simple correspondence approach for an increase of the disparity estimation confidence. A stereo matching component running concurrently with the simple feature detector, delivering depth information for the features. This approach needs to be resource optimized to not hinder the application processes.

Disparity map estimation, however, is a non-trivial problem which the community is only now starting to find a complete solution to. However, these solutions either require bulky systems or extended computation time. For mobile autonomous systems, real-time operation is required. Extracting depth from two images of half a million pixels at this rate is no small feat. Additionally, a complete vision system residing in an FPGA requires several processing components just for preprocessing the image data, such as, image rectification, motion artifact compensation, and depth estimation. Furthermore, higher-level applications, such as tracking, object recognition, or navigation, should also fit. Fitting all these parts of an autonomous agent onto a compact and power-constrained embedded mobile system is a real challenge.

It is necessary to adopt an approach that is capable of meeting the requirements for the low-level processing to enable high-level processing, but that can also fit the high-level processes concurrently. Thus, all building blocks need to be reduced. Enabling more computations in the FPGA, by reducing

the components for preprocessing, will improve the capability and flexibility of the system. Furthermore, it is not important to achieve maximum accuracy in the algorithms. With a high-speed system, correction or filtering can be used to compensate. A high sample rate allows for more simple algorithms. Then, rather than trying to develop a new feature detector or correspondence algorithm, our focus is on utilizing "good enough" algorithms by combining and optimizing them for reconfigurable hardware. The end goal is a small and high-speed hardware system working as the eyes and visual cortex of any type of autonomous vehicle or robot.

1.3 Outline of thesis

The continuation of this thesis consists of two main parts. The first part consists of 3 chapters: Chapter 2 presents the related work; Chapter 3 provides an overview of the included research papers; Chapter 4 presents overall contributions and conclusions together with possible future work. The second part of this thesis consists of Chapters 5 through 7 and is a collection of the research publications which form the basis of this thesis.

Chapter 2

Related Work

The concept of using reconfigurable hardware for image processing is not new. Several competent approaches exist, but most have one or more tradeoffs: quality, resource utilization, or limitation in image size. Which is the most important parameter is an application specific question, but for our purpose, resource utilization is important as we seek to fit an entire autonomous agent in our system.

2.1 Visual Navigation

Several SLAM approaches have been presented, such as Barfoot [21], Bertolli et al. [6], and Montemerlo et al. [22]. However, the approaches are not suitable for FPGA implementation. An FPGA implementation of SURF is presented by Svab et al. in [23]. However, they only implement part of the algorithm as the complexity and time-consuming nature of the algorithm makes it difficult to realize on the FPGA. The descriptor generation is handled in software on a Power-PC, and the complete navigation system is residing on a laptop. Hence, another approach is required to fit a complete navigation system in an embedded system.

2.2 Stereo Matching

Performance measurements of correspondence algorithms, such as presented by Hirschmüller and Scharstein [24], mostly focus on the accuracy of the dis-

parity map, whereas real-time implementations rank the throughput, or frame rate, higher.

2.3 Resource Constraint

Since the aim of our work is to achieve an acceptable performance at a low resource usage, we need to specify what low resource usage is. Resource utilization in an FPGA is normally expressed in slices and LUTs (LookUp-Table which realize boolean operations). In our previous work, our system produced an acceptable disparity map at 1221 slices when implemented in a Spartan-3 FPGA. This is just above 4% of the available slices on the chip.

Several stereo matching approaches with low resource usage have been proposed, such as by Arias-Estrada et al. [25]. Their utilization is only 4.2K slices on a Virtex-II, but with a fair disparity map. The implementation presented by Lee et al. [26] comes in at a resource usage below 10K slices. The produced disparity map is moderate showing extensive blurring of edges and noise.

For higher quality disparity maps, the resource usage inevitably go up. Very good results are presented by Zhang et al. [27], but the utilization is 95K slices plus a large amount of ALUTs and DSP blocks, leaving little room for concurrent processing. A collection of proposed FPGA implementations is presented by Lazaros et al. in [28].

2.4 Area Matching

Very accurate results have been presented for area-based approaches [18], but the high quality of these implementations mostly come at the expense of computational power and, hence, processing time.

Recently a number of non-global near real-time implementations have been presented. They are not truly local as they are akin to global methods such as Dynamic Programming [29], but operate on a limited area [30]. The near real-time software implementations tend to utilize special purpose hardware, such as GPUs [31],[13], to accelerate the processing. Although impressive in their performance, they are not really suitable for mobile and embedded systems, considering the cost, size and power requirements. Transferring these approaches to an FPGA is not optimal, as they resort to iterative approaches with computational and memory requirements that are hard to realize for the limited resources of an FPGA [31]. Large memory can be included when constructing

an FPGA-system, but the memory speeds required are above the capacity of standard FPGAs.

2.5 Support Window

There are numerous proposals to overcome the static window issue, as discussed before. Adaptive window approaches suggested by Kanade and Okutomi [12] or the one by Boykov et al. [32] are an ill match for our system, as they exhibit the same problems as we do with noise and sensitivity to low-texture areas. Additionally, they rely on models with empirically derived parameters unique to every scene. This might not be much different from empirical selection of window size for our standard approach, but it is not an improvement either.

Hirschmüller et al. [33] suggests an approach using multiple windows for good depth discontinuity performance. Although based on SAD, it requires a large memory. Another multiple window approach proposed by Chonghun et al. [34] seems promising at first, but their reason for multiple windows is the refinement of an overly-smoothed noise-free first estimation, the inverse from our approach.

Adaptive support-weight approaches, as suggested by Yoon et al. [14] and Gu et al. [35], produce good disparity maps but at a low frame rate.

Yi et al. [36] found that the effect of the shape of the support window has less impact than the number of pixels in the window. This together with the result from Lee in [26] that square matching windows can be reduced to half the height without substantial reduction in quality, leads to a question of to what extent a window height reduction can be compensated with a increased width. Ambrosch showed that for window widths beyond the commonly used sizes (up to 21 pixels) the accuracy actually degrades [37].

The ultimate reduction in window height is the 1-dimensional window. It is not extensively found in literature, possibly because its produced disparity map is noisy. However, a few implementations can be found.

Ambrosch [37] uses a 1x1 SAD, for weighting the comparison of a Census matching approach in advantage of the center pixel. Calin et al. use a 1-dimensional SSD [38] implementation. It runs at 30 fps producing dense disparity maps of 160x120 pixels on an FPGA. The objects of the disparity map are excessively bloated, as to be expected when using a wide correlation window, and the depth resolution is limited, partially due to the small image size. Lefebvre et al. [39] presents an approach for 1-dimensional matching

paired with a confidence estimation. The work produces semi-dense disparity maps with associated match confidence map. However, the matching is made through multiple 1-dimensional windows of different sizes and not in real-time. An interesting conclusion of theirs is that the basic 1-dimension approach yields better results than the 2-dimensional in areas of texture and near depth discontinuities [40]. The difference is actually quite substantial for larger window sizes, with the advantage of the 2-dimensional in other areas being marginal. The matching algorithm is SSD, but any correlation technique may be used to construct the correlation volume from which the estimate the disparity and confidence. They show that 1-dimensional windows contain sufficient information for estimating semi-dense disparity maps with good confidence. The approach is far from real-time with a calculation time of 7 seconds for the Tsukuba image pair.

2.6 Disparity Map Improvements

For completing hollow disparity maps, common approaches are to interpolate or propagate disparity values from nearby matched pixels. Yoon et al. [41] perform a spatial interpolation by the use of median filtering. In propagation, the approach is that a window of estimated disparity values completes the non-valid elements with the least value available in the window to limit the foreground fattening, Fusiello et al. [42]. However, a propagation of background disparity values will thin out and often break thin foreground objects. The propagation window can instead be weighted to include disparity information only from same object neighbors. Sun et al. [30] restrict the selection to pixels of similar color, supported by the color-disparity constraint. Although producing good results, propagation methods rely on a fairly accurate first disparity estimation. Moreover, it is common with streaking artifacts in methods of propagation [30].

Chapter 3

Research Summary

The research group in Robotics at Mälardalens Högskola is focused on visual pre-processing for robots and autonomous machines. This initial and crucial stage of autonomy deals with information gathering and environment perception - such as navigation based on visual cues, and object recognition. The work presented has been performed within this group, and the focus has been on electronics, hardware, and looking at computer vision from an electronics perspective.

This chapter presents a short overview of the underlying papers of this thesis.

3.1 Paper Overview

3.1.1 Paper A

Two Camera System for Robot Applications; Navigation, Jörgen Lidholm, Fredrik Ekstrand and Lars Asplund, In proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Hamburg, Germany, September 2008

Summary We present a hardware-based stereo vision system for navigation. The objective is to create a system for simultaneous location and mapping through the use of vision on an embedded reconfigurable hardware system.

SLAM is a complex task with a lot of data to process and many parameters to consider. Our approach is to see if it is possible to use only a limited feature

descriptor, instead of SIFT or SURF, at high speed to identify landmarks. Although only concerned with a limited number of reasonably separated features, the confidence in a straight-forward matching technique (SAD on cornerness) is too low as the corner descriptors are too simple for matching on individual basis. This led to the alternative approach suggested here, which is a combination of traditional stereo matching, back-projection [43] and tracking.

We propose to remove the problem of outlier detection and removal by matching of 3D coordinates. The approach is similar to that of area-based matching. For every feature in one image we match with all possible features in the other image, constrained by the rectified image condition limiting the search area to 1-dimensional. There is no selection performed, all the possible matches are stored (similar to the Disparity Space Image in left-right consistency check implementations). Within this set there can be only one valid match.

This landmark set is stored and the robot is moved slightly. By tracking the motion using wheel-based odometry, we have a notion on how the correct features should have moved in 3D space, and by back-projecting this onto the stored landmark set coordinates, we get their expected new coordinates. Correlating these with the newly acquired landmark set, only those representing the correct landmark should match. The confidence of the landmark increases with the uniqueness and stability (number of correlations). Of course, wheel-based odometry is not reliable over longer paths, so as soon as a sufficient set of landmarks with good confidence is generated, it is superseded by visual odometry.

An FPGA implementation of Stephen and Harris combined edge and corner detector is used to reduce the data amount in the main application. A novel approach focused on a high frame rate to reduce the problem of matching and tracking is proposed. The approach, however, was not fully developed and a modified approach was presented in [44] by the use of clustering.

My contribution I am the second author of this paper contributing with electronics design and implementation, co-implementation of VHDL-components, co-developing the idea, and formulating sections of the text.

3.1.2 Paper B

Resource Limited Hardware-based Stereo Matching for High-Speed Vision System, Fredrik Ekstrand, Carl Ahlberg, Mikael Ekström, Lars Asplund and Giacomo Spampinato, In proceedings of the 5th International Conference on Au-

tomation Robotics and Applications (ICARA), Wellington, New Zealand, December 2011 (*to appear*)

Summary The depth assessment in Paper A was not satisfactory. An alternative approach is to work with features with a good initial 3D coordinate guess. A matching component providing valid disparity information in the salient parts of the image only, will allow for depth information without feature matching (by superposition). This concurrent matching component must use only a limited set of resources, in order not to restrict the other processes.

The task is to find a stereo matching approach suitable for resource constrained implementation. An important issue is also the memory requirement of the matching component when handling large images, as the higher level processes may not be blocked from memory access by the correspondence component.

A constrained implementation of two popular correlation approaches specifically suited for hardware implementation, SAD and Census, showed that the basic approach performed best with significant limitation of the matching area. A 1D SAD implementation resulted in a resource optimized disparity component suitable for the task, fulfilling the prerequisites of no limitations in terms of external memory or image size.

My contribution I am the main author of this paper contributing with the idea, literature survey, algorithm and hardware implementation, and verification. The second author provided relevant insights, data for the publication, software-based validation of findings, and paper revision. The other authors have contributed by giving feedback on the theory and actively participating in paper revisions.

3.1.3 Paper C

Utilization and Performance Considerations in Resource Optimized Stereo Matching for Real-Time Reconfigurable Hardware, Fredrik Ekstrand, Carl Ahlberg, Mikael Ekström, Lars Asplund and Giacomo Spampinato, Technical Report

Summary As a direct result of the findings in Paper B, we formulated an extension of the approach into a matching component producing a dense disparity map with retained low resource utilization. Established methods for improving area-based matching methods are implemented from a hardware perspective.

The approach significantly improves the performance of the implementation from Paper B and performs on par with recently published real-time dense disparity map components. The resource utilization is kept low and the memory and image size restrictions are maintained.

My contribution I am the main author of the paper, contributing with the state of the art and formulating the approach, as well as performing the hardware implementation and verification. The second author contributed with problem identification, initial testing, development of the approach, and software-based validation. The third author contributed with relevant feedback and insights together with paper revisions. The other authors have contributed by giving feedback on the theory and actively participating in paper revisions.

3.2 Research Methodology

The research is based on literature surveys to perceive the state of the art. Approaches are evaluated based on suitability of implementation through empirical methodologies including analysis of quantitative data by community practice.

Chapter 4

Conclusions and Future Work

This thesis gives a quick overview and introduction to image-processing in reconfigurable hardware. Important aspects for implementing in hardware is the suitability of the algorithm in terms of speed, complexity and resource utilization. We have looked at minimizing the system impact to enable concurrent processing of traditionally computationally expensive operations. The key aspect is to focus on speed and process on the go without retaining data in low-level processing.

4.1 Contributions

The work presented in this thesis enables different levels of depth extraction. For the minimized approach of running next to a feature-based navigation system, the approach can supply 3D data in salient areas in high speed and at very low resource usage. Salient regions are important in a wide range of applications, and feature detectors use these regions to enable everything from autonomous navigation to face-detection. Combining feature-based matching with a compact, fast and potent disparity estimator can relieve some of the need for expensive feature descriptors. The benefits would be higher speed and lower resource usage, enabling higher system integration.

We have shown in this thesis that it is possible to retain the quality of one of the most widely used stereo matching algorithm while removing a few of

its downsides. For approaches with a demand for more dense 3D data, the improved versions can produce semi-dense disparity maps at a high speed, and without a limitation on the size of the images processed.

The removal of matching data introduces noise, which can be removed by filtering, especially in area-based matching. The median-filtered 1-dimensional stereo matching component effectively reduce the resource utilization, but with retained accuracy. Moreover, the median filter does not improve the 2-dimensional approach with any significance, which is why the 1-dimensional implementation in certain aspects actually outperforms its larger counterpart.

4.2 Future Work

Future work includes integration of the feature detector and the disparity estimator to provide feature matching and tracking with high confidence. Another interesting question is if an advanced confidence measurement can invalidate false matches at an early stage, and thereby keep the noise from ever entering the disparity domain. For this to have any relevance, an extended propagation function is required. As is evident in this thesis, removal of data requires compensation.

The next step is to run the autonomous system performing navigation indoors. Coming future work is to adopt the system for outdoors. A whole new range of parameters will then need to be considered, such as motion compensation, radiometric distortion, visual noise, etc.

Bibliography

[1] N.K. Ratha and A.K. Jain. FPGA-based computing in computer vision. *Computer Architecture for Machine Perception, 1997. CAMP '97. Proceedings Fourth IEEE International Workshop on*, pages 128–137, 20-22 Oct 1997.

[2] Nicu Sebe and Michael S. Lew. Comparing salient point detectors. *Pattern Recognition Letters*, 24(1-3):89 – 96, 2003.

[3] C. Harris and M. Stephens. A combined corner and edge detection. In *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151, 1988.

[4] Cordelia Schmid, Roger Mohr, and Christian Bauckhage. Evaluation of interest point detectors. *International Journal of Computer Vision*, Vol. 37(2):151–172, June 2000.

[5] Haichao Li. Feature matching based on corner and edge constraints. *SPIE The International Society for Optical Engineering*, pages 1615–1630, 2007.

[6] Federico Bertolli, Patric Jensfelt, and Henrik I. Christensen. SLAM using visual scan-matching with distinguishable 3d points. *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 4042–4047, Oct. 2006.

[7] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[8] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:1615–1630, 2005.

[9] A. Bensrhair, P. Miche, and R. Debrie. Fast stereo matching for implementation in a 3-d vision sensor. In *Industrial Electronics, Control and Instrumentation, 1991. Proceedings. IECON '91., 1991 International Conference on*, pages 1779 –1783 vol.3, oct-1 nov 1991.

[10] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47:7–42, 2002.

[11] Yasushi Sumi, Yutaka Ishiyama, and Fumiaki Tomita. Robot-vision architecture for real-time 6-dof object localization. *Comput. Vis. Image Underst.*, 105(3):218–230, 2007.

[12] T. Kanade and M. Okutomi. A stereo matching algorithm with an adaptive window: theory and experiment. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(9):920 –932, sep 1994.

[13] Xing Mei, Xun Sun, Mingcai Zhou, Shaohui Jiao, Haitao Wang, and Xiaopeng Zhang. On building an accurate stereo matching system on graphics hardware. *GPUCV'11: 1st IEEE Workshop on GPU in Computer Vision Applications*, 2011.

[14] Kuk jin Yoon, Student Member, and In So Kweon. Adaptive support-weight approach for correspondence search. *IEEE Trans. PAMI*, 28:650–656, 2006.

[15] Takouhi Ozanian. Approaches for Stereo Matching. *Modeling, Identification and Control*, 16(2):65–94, 1995.

[16] Pascal Fua. A parallel stereo algorithm that produces dense depth maps and preserves image features. *Machine Vision and Applications*, 6(1993):35–49, 2004.

[17] Geoffrey Egnal and Richard P. Wildes. Detecting binocular half-occlusions: Empirical comparisons of five approaches. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24:1127–1133, August 2002.

[18] http://vision.middlebury.edu/stereo.

[19] Stephen Se, Timothy Barfoot, and Piotr Jasiobedzki. Visual motion estimation and terrain modeling for planetary rovers. In *i-SAIRAS 2005 - International Symposium on Artificial Intelligence, Robotics and Automation in Space*, September 2005.

[20] Herbert Bay, Tinne Tuytelaars, and Luc J. Van Gool. Surf: Speeded up robust features. In Ales Leonardis, Horst Bischof, and Axel Pinz, editors, *ECCV (1)*, volume 3951 of *Lecture Notes in Computer Science*, pages 404–417. Springer, 2006.

[21] T.D. Barfoot. Online visual motion estimation using FastSLAM with SIFT features. *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 579–585, 2-6 Aug. 2005.

[22] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, Edmonton, Canada, 2002. AAAI.

[23] J. Svab, T. Krajnik, J.Faigl, and L. Preucil. FPGA-based speeded up robust features. In *2009 IEEE International Conference on Technologies for Practical Robot Applications 2009*, November 2009.

[24] Heiko Hirschmüller and Daniel Scharstein. Evaluation of stereo matching costs on images with radiometric differences. *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 1582–1599, 2009.

[25] Miguel Arias-Estrada and Juan Xicotencatl. Multiple stereo matching using an extended architecture. In Gordon Brebner and Roger Woods, editors, *Field-Programmable Logic and Applications*, volume 2147 of *Lecture Notes in Computer Science*, pages 203–212. Springer Berlin / Heidelberg, 2001.

[26] Lee Sunghwan, Yi Jongsu, and Kim Junseong. Real-time stereo vision on a reconfigurable system. *Lecture Notes in Computer Science*, 3553:299–307, 2005.

[27] Lu Zhang, Ke Zhang, Tian Sheuan Chang, Gauthier Lafruit, Georgi Krasimirov Kuzmanov, and Diederik Verkest. Real-time high-definition stereo matching on fpga. In *Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays*, FPGA '11, pages 55–64, New York, NY, USA, 2011. ACM.

[28] Nalpantidis Lazaros, Georgios Christou Sirakoulis, and Antonios Gasteratos. Review of stereo vision algorithms: From software to hardware. *International Journal of Optomechatronics*, 2(4):435–462, 2008.

[29] Minglun Gong and Yee-Hong Yang. Fast stereo matching using reliability-based dynamic programming and consistency constraints. In *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, ICCV '03, pages 610–, Washington, DC, USA, 2003. IEEE Computer Society.

[30] Xun Sun, Xing Mei, Shaohui Jiao, Mingcai Zhou, and Haitao Wang. Stereo matching with reliable disparity propagation. *The First Joint 3DIM3DPVT Conference 3DIMPVT 2011*, 2011.

[31] Christian Richardt, Douglas Orr, Ian Davies, Antonio Criminisi, and Neil A. Dodgson. Real-time spatiotemporal stereo matching using the dual-cross-bilateral grid. In *ECCV (3)'10*, pages 510–523, 2010.

[32] Yuri Boykov, Olga Veksler, and Ramin Zabih. A variable window approach to early vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:1283–1294, 1998.

[33] H Hirschmüller, P R Innocent, and J Garibaldi. Real-time correlation-based stereo vision with reduced border errors. *International Journal of Computer Vision*, 47(1):229–246, 2002.

[34] Chonghun Roh, Taehyun Ha, Sungsik Kim, and Jaeseok Kim. Symmetrical dense disparity estimation: algorithms and fpgas implementation. In *Consumer Electronics, 2004 IEEE International Symposium on*, pages 452 – 456, 1-3, 2004.

[35] Zheng Gu, Xianyu Su, Yuankun Liu, and Qican Zhang. Local stereo matching with adaptive support-weight, rank transform and disparity calibration. *Pattern Recognition Letters*, pages 1230–1235, 2008.

[36] JongSu Yi, JunSeong Kim, LiPing Li, John Morris, Gareth Lee, and Philippe Leclercq. Real-time three dimensional vision. In *Asia-Pacific Computer Systems Architecture Conference'04*, pages 309–320, 2004.

[37] Kristian Ambrosch and Wilfried Kubinger. Accurate hardware-based stereo vision. *Computer Vision and Image Understanding*, pages 1303–1316, 2010.

[38] G. Calin and V. O. Roda. Real-time disparity map extraction in a dual head stereo vision system. *Latin American applied research*, 37:21 – 24, 01 2007.

[39] S. Lefebvre, S. Ambellouis, and F. Cabestaing. A colour correlation-based stereo matching using 1d windows. In *Signal-Image Technologies and Internet-Based System, 2007. SITIS '07. Third International IEEE Conference on*, pages 702 –710, dec 2007.

[40] S. Lefebvre. Approche monodimensionnelle de la mise en correspondance stéréoscopique par corrélation - application à la détection d'obstacles routiers. *Thèse de doctorat, Université des Sciences et Technologies de Lille*, 2008.

[41] Sukjune Yoon, Sung-Kee Park, Sungchul Kang, and Yoon Keun Kwak. Fast correlation-based stereo matching with the reduction of systematic errors. *Pattern Recognition Letters*, pages 2221–2231, 2005.

[42] A. Fusiello, V. Roberto, and E. Trucco. Experiments with a new area-based stereo algorithm, 1997.

[43] Frank Tong and Ze-Nian Li. Backprojection for stereo matching using transputers. *SPIE The International Society for Optical Engineering*, 1992.

[44] Jörgen Lidholm, Giacomo Spampinato, and Lars Asplund. Validation of stereo matching for robot navigation. In *14th IEEE International Conference on emerging Technologies and Factory Automation ETFA 2009*, September 2009.

II

Included Papers

Chapter 5

Paper A:
Two Camera System for Robot Applications; Navigation

Jörgen Lidholm, Fredrik Ekstrand and Lars Asplund
In proceedings of the IEEE international conference on Emerging Technologies and Factory Automation (ETFA), Hamburg, Germany, September 2008

Abstract

Current approaches to feature detection and matching in images strive to increase the repeatability of the detector and minimize the degree of outliers in the matching. In this paper we present a new approach; we suggest that a lower performance feature detector can produce a result more than adequate for robot navigation irrespectively of the amount of outliers. By using an FPGA together with two cameras we can remove the need for descriptors by performing what we call spurious matching and the use of 3D landmarks. The approach bypasses the problem of outliers and reduces the time consuming task of data association, which slows many matching algorithms down.

5.1 Introduction

Navigation, object detection and object recognition are fundamental problems in robotics - all which can be resolved with vision. The fundamental task in any of these applications is the reduction of the image complexity. In order to reduce the image information it is necessary to extract salient properties of the image using for example a line, edge or corner detector, or a combination thereof. Various approaches have been made on adapting these principle feature detectors into more complex, high-level detectors with different sets of descriptors. The idea is to increase the invariant properties of the detector and thereby increase what is by many viewed as the most important factor of the detector, the repeatability. However, in general, the more complex the detector the more computational heavy it becomes. When features have been extracted, the next step is to perform some sort of matching, either by tracking a feature in subsequent images or by matching in two cameras. The general idea is that if this is to be performed at a high frame rate (around 30 frames per second) it requires a high repeatability detector and a computationally light matching algorithm with minimal dynamic properties or at least a known worst case execution time.

In this paper we present a new approach. We suggest that it is possible to produce a result adequate for navigating a mobile robot with a lower performance feature detector. We use reprogrammable hardware (FPGA) together with two cameras to generate a real-time, stereo-vision, feature detector and matching application. By using the motion of the robot we can reduce the problems associated with feature matching. The advantages of an FPGA are manifold; the parallel properties of an FPGA makes for a high-throughput, small footprint system, and the comparatively low power consumption makes it ideal for mobile applications.

5.2 Related work

Robot navigation is a well-explored subject with vision based navigation being where the current focus lies. The approach of using an FPGA for the system is also becoming widely adopted as it enables real-time image processing [1] [2], which is a crucial part in mobile applications [3]. For certain applications FPGAs are better suited than desktop computers due to their parallel structure. In [4] an FPGA implementation outperforms a PC by one order of magnitude for the SIFT detector [5]. The power of the FPGA is further shown in [6] where they are unable to run Harris corner detector in real-time on a computer

with an Opteron processor running at 2.6 GHz. This advantage over personal computers is likely to remain as both technologies are evolving in a similar fashion performance-wise.

Navigation by vision requires matching of images, or rather features in separate images. Tracking features in real-time in subsequent images from a camera is not a trivial task, partially due to the fact that it requires a very stable feature detector with high repeatability [7]. The current feature detectors with the highest repeatability, such as SIFT [5] and SURF [8], create descriptors for each feature in order to simplify the matching task. Unfortunately, the high dimensionality of such a descriptor means that it is computationally intense [9].

All matching algorithms are faced with the correspondence problem, i.e., how to match corresponding features from two images without assigning any incorrect matches. A common approach is to use a statistical method to measure how well a matching pattern matches. Examples of methods are *cross-correlation* and *sum of squared differences*, but there will always be outliers, features not correctly matched, or not matched at all. Many have tried to minimize the occurrence of outliers, and in [10] a comparison, and a new approach, is presented.

5.3 Experimental platform

We have designed an FPGA based vision system intended to work as a general purpose research platform. With up to four 5-megapixel cameras and an eight million gate equivalents FPGA.

5.3.1 Image sensors

The system uses the MT9P031 5-megapixel CMOS digital image sensor from Micron. The sensor elements are organized in a Bayer pattern, i.e., the first line consists of green and red pixels and the second line consists of blue and green pixels, see figure 5.1.

The pixels can be read in a number of ways. The readout follows that of the Bayer pattern, however the order can be mirrored and pixels skipped for both the row and column. In skipping mode, a number of row-pairs and/or column-pairs are not sampled, i.e. skipped, thereby reducing the resolution and increasing the frame rate but preserving the field of view.

It is possible to combine the adjacent skipped pixels in order to reduce the effect of aliasing introduced by skipping. This is called binning and results in

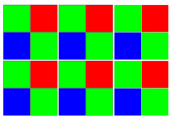

Figure 5.1: The Bayer pattern pixel layout with one row of green (light gray) and red (medium gray) pixels, and one row of blue (dark gray) and green pixels.

a more coherent/smooth image, than with skipping, but also in lower performance as all the pixel elements need to be sampled.

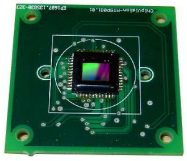

Figure 5.2: MT9P031 image sensor from Micron mounted on our carrier board (the lens is not mounted).

Additionally, one can specify what region of the image sensor to read, which is useful when only a limited field of view is needed and a higher frame rate desired. The imaging sensor is capable of running at 96 MHz, and the frame rate is dependent on the clock frequency and the frame size, i.e., the number of pixels read.

5.3.2 FPGA board

The FPGA board has a size of 70×55mm and is equipped with a Xilinx Virtex II XC2V8000 FPGA together with 256 Mbit flash, 512 Mbit SDRAM and a CPLD. The flash memory stores FPGA configurations and it accommodates 8 different configurations. At power on the CPLD loads the FPGA according to the configuration selector setting. The configuration selector is fitted on the Carrier Board (section 5.3.3) and may be overrun by for example a micro controller. See figure 5.3.

Table 5.1: Micron MT9P031 CMOS image sensor features [11]

MT9P031 features	
Color filter array	RGB Bayer pattern
Maximum data rate	96 Mp/s at 96 MHz
Power consumption	381mW at 14 fps full resolution
Pixel size	2.2μm×2.2μm
Maximum frame rates	
2592×1944	14 fps
1280×720 skipping	60 fps
640×480 with binning	53 fps
640×480 with skipping	123 fps

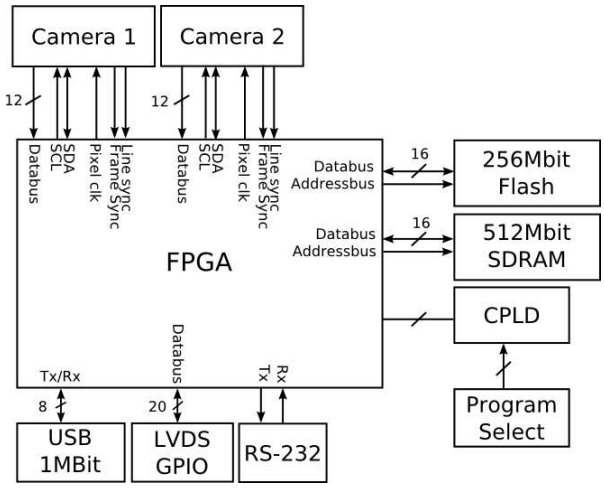

Figure 5.3: Block diagram of the camera system, a maximum of four cameras can be connected.

5.3.3 Carrier board

The carrier board has a size of 110×90mm and have four camera connections, with all signals, individual to each camera and generated by the FPGA. Additionally, the carrier board incorporates a program selector, power supply, a USB controller, serial port and control-IO signals. It is also fitted with a FireWire

connector for future extension.

5.4 Feature detectors

A feature can be a corner, line or any salient region which can be extracted from an image.

One of the first feature detectors was Moravec's corner detector [12], from 1977, which Harris *et. al.* improved in 1988 [13]. Since then, many other feature detectors have been developed with different qualities [14]. Most detectors are designed with repeatability in mind, although some are designed for other properties, such as speed [6]. Repeatability, as defined in [14], is an important property, however, for our application, localization accuracy and speed are paramount. Harris is still one of the most robust detectors available and this together with its speed when implemented on an FPGA makes it a suitable detector for this application.

5.4.1 Stephen and Harris combined corner and edge detector

In [14] the authors concluded that, among the tested feature detectors, (Foerstner, Cottier, Heitger, Horaud, Harris and Improved Harris), the improved version of the Harris corner detector performed best regarding repeatability and information content. The original implementation of the same detector was, however, not far behind.

Moravec's corner detector measures the variation in intensity in an image and looks for low self-similarity in a point. A corner is defined as a point with low similarity to the surrounding region in all directions, i.e., a point where the minimum change in intensity, in any direction, is large (above a certain threshold) [13].

According to Stephen and Harris, Moravec's detector, however, suffers from a number of problems which they try to correct with their combined corner and edge detector. In order to remove the anisotropy and noise of the discrete, rectangular window in which the variation is calculated, they introduce an analytic expansion about the shift origin together with smoothing with a Gaussian filter. By also taking into account the direction of shift they can produce a rotationally invariant detector that is not oversensitive to edges.

Stephen and Harris also introduce a response function in order to select isolated interest points, as opposed to simply classify the region as containing a potential feature. This response function, which includes a structure matrix

calculated from image derivatives, indicates the quality of the detected feature and allows for the filtering out of less distinctive features with the use of a threshold similar to Moravec's.

5.4.2 FPGA implementation of Harris corner detector

We have a VHDL implementation of the Stephens and Harris combined corner and edge detector. It was originally implemented as a undergraduate thesis for an older vision system. We have adapted it to a new, larger FPGA, allowing us to increase the parallelism and thus improve the speed.

Some operations need to be performed sequentially for practical purposes. One of the most limiting factors of the FPGA is the number of multipliers available. Certain steps in the algorithm requires simultaneous multiplications, and the need for multipliers would surpass the available numbers if parallelled to the full extent. In order to save computational resources, the units needs to be "reused", i.e., not exclusive to a single task. Due to the fact that the corner detector measures the intensity in the image and not the saturation or color values, we need only measure the contribution in one point of the Bayer matrix, i.e., the green pixel. We chose to use only one value per color quadrant and thus we only feed the corner detector with a new pixel every other column every other row. This leaves room for sequential operations on four clock cycles for every pixel.

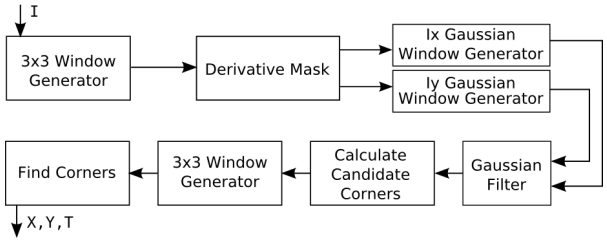

Figure 5.4: A block diagram of our VHDL implementation of the Harris corner detector.

The corner detector uses 3×3 and 5×5 pixel windows. This is the only buffering required, all other processing is performed as the pixel data arrives. Our implementation of the Stephen and Harris combined corner and edge detector can be seen in the block diagram in figure 5.4. The process consists of 7 major, internally piped, blocks. The first block creates a 3×3 sliding window.

When two pixel rows plus one pixel have been extracted from the camera the first window is passed on to the "Derivative Mask" block.

The derivative block calculates the intensity x- and y-gradients. These values, the first derivatives, are then passed onto the window generator for the multiplication/Gaussian stage, which creates a 5×5 sliding window.

In the multiplication stage, the structure matrix is calculated and then run through a Gaussian filter. The Gaussian filter is constructed using shift operations, as opposed to multiplications, in order to save multipliers that can be used for either increased parallelization or multiplier-heavy postprocessing. The filter is not a true Gaussian function as the values are selected to enable shifting, but no performance degradation has been observed for the approximation, which can be supported by [15] that shows that Gaussian weighting need not be the optimal weighting function.

The filtered value is then used in the response function and the result is fed to a new window generator. The last stage of the pipe filters the response value so that only the local maxima within the 3×3 sliding window generates a corner response, as long as it exceeds the current threshold.

5.5 Interest point location

An interest point is, what we call, a stereo matched feature that can be located in a coordinate system as a landmark, that a robot can use for navigation. In this section we describe how we can calculate the location of a landmark from two stereo matched features. The same procedure, in reverse order, can be followed to calculate the pixel coordinate at which a landmark should appear, given the robots current location and attitude.

We use the right-handed coordinate system with positive X to the right, positive Y in front and positive Z above.

The full definition of the robot absolute vector defines the position in three dimensions and the attitude in three dimensions (5.1). The robot center is located at the floor in the center of the robot in the x, y plane.

$$\mathbb{R} = (X_r, Y_r, Z_r, \alpha_r, \beta_r, \gamma_r) \tag{5.1}$$

Since the robot is moving in a controlled indoor environment without slopes, we can consider Z_r constant and zero. The same applies for α_r and β_r. The stereo camera rig has a fixed location on the robot and the constant relative vector of each camera is defined in (5.2), where n marks the camera, left or right. The vector is relative to the robot center. To simplify the stereo matching the $\hat{\beta}$

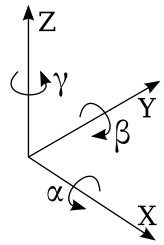

Figure 5.5: Right-handed coordinate system

factor should be zero and the $\hat{\alpha}$ should be the same for both cameras, resulting in that line l in the left camera corresponds to line l in the right camera.

$$\underline{c}_n = \left(x_n, y_n, z_n, \hat{\alpha}_n, \hat{\beta}_n, \hat{\gamma}_n\right) \tag{5.2}$$

The absolute vector of each camera can be calculated by adding the relative camera vector to the absolute robot vector, see (5.3, 5.4).

$$\begin{aligned}\mathbb{C}_n &= (X_r + x_n, Y_r + y_n, Z_r + z_n, \\ &\quad \hat{\alpha}_n, \hat{\beta}_n, \gamma_r + \hat{\gamma}_n) && (5.3)\\ &= (X_n, Y_n, Z_n, \alpha_n, \beta_n, \gamma_n) && (5.4)\end{aligned}$$

Every pixel in an image corresponds to a two dimensional direction which can be calculated from the focal length of the lens f and the pixel separation on the camera chip P_{width} and P_{height}. The two angles θ and ϕ and an unknown length r, form a polar vector (r, θ, ϕ).

(X_p, Y_p) denotes the pixel coordinate, with the camera center at $(0, 0)$.

$$\theta = \arctan\left(\frac{X_p * P_{width}}{f}\right) \tag{5.5}$$

$$\phi = \arctan\left(\frac{Y_p * P_{height}}{f}\right) \tag{5.6}$$

By using (5.5) and (5.6) we can find the angular distance between every pixel. The MT9P031 camera chip has a pixel separation of $2.2\mu m$ (table 5.1)

but we are only sampling every second pixel column and row, thus doubling the pixel separation to $4.4\mu m$. The focal length of the lens is 6 mm. This results in approximately 0.7 milliradians per sampled pixel in both horizontal and vertical directions at the center of the image.

θ of each pixel is the angle from the center line, since the pixels are enumerated with (0,0) at the center of the camera. To apprehend θ_l and θ_r as seen in figure 5.6

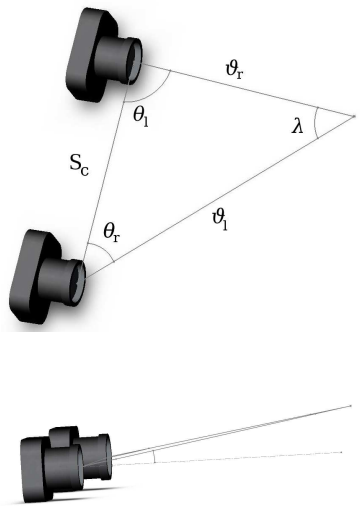

Figure 5.6: The angles from each camera to a feature point. θ for the left and right camera, the camera separation S_c and ϕ, which should be the same for both cameras.

Lets consider the case where we know which feature in the left camera corresponds to which feature in the right camera. By forming a triangle with corners at the two camera centers and the interest point with angles as seen in figure 5.6 we can calculate the distance of the two unknown triangle edges by using the law of sine, see equation (5.10-5.12). The camera separation is known and denoted S_c.

$$\theta_l = \frac{\pi}{2} - \theta + \underline{c}_{l,\hat{\gamma}} \tag{5.7}$$

$$\theta_r = \frac{\pi}{2} + \theta + \underline{c}_{r,\hat{\gamma}} \tag{5.8}$$

$$\phi_l = \phi_r = \frac{\pi}{2} - \phi \tag{5.9}$$

$$\lambda = \pi - \theta_l - \theta_r \tag{5.10}$$

$$\frac{\vartheta_n}{\sin(\theta_n)} = \frac{S_c}{\sin(\lambda)}, \qquad n \in \{l, r\} \tag{5.11}$$

$$\vartheta_n = \frac{S_c * \sin(\theta_n)}{\sin(\lambda)}, \qquad n \in \{l, r\} \tag{5.12}$$

Using ϑ_n, θ_n and ϕ_n we can form a relative polar vector from each camera to an interest point, $(\vartheta_l, \theta_l, \phi_l)$ and $(\vartheta_r, \theta_r, \phi_r)$.

By converting the relative polar vector $(\vartheta_n, \theta_n, \phi_n)$ to a cartesian coordinate and adding it to the absolute cartesian coordinate of the corresponding camera we get absolute cartesian coordinate of the interest point (5.15). Note that the polar vector is rotated to the attitude of the camera, which is necessary when forming the triangle.

C and P marks the cartesian and polar coordinate system respectively or a transformation between the two.
The cartesian location of camera n.

$$C(\mathbb{C}_n) = (X_n, Y_n, Z_n) \tag{5.13}$$

The direction and distance to the interest point k.

$$P(\mathbb{I}_k) = (\vartheta_n, \theta_n, \phi_n) \tag{5.14}$$

The space location of the interest point.

$$C(\mathbb{I}_k) = C(\mathbb{C}_n) + C(P(\mathbb{I}_k)) \tag{5.15}$$

The conversion from polar vector to cartesian coordinate requires the use of *sine* and *cosine* as seen below, where r is the vector length. $\cos(\theta)$ is, however, equal to $\sin(\frac{\pi}{2} - \theta)$ which allows a *sine* only implementation in the FPGA using look-up-table (LUT).

$$x = r * \sin\phi * \cos\theta \tag{5.16}$$
$$= r * \sin\phi * \sin(\frac{\pi}{2} - \theta) \tag{5.17}$$
$$y = r * \sin\phi * \sin\theta \tag{5.18}$$
$$z = r * \cos\phi \tag{5.19}$$
$$= r * \sin(\frac{\pi}{2} - \phi) \tag{5.20}$$

5.5.1 Image sequence feature tracking

To track features in an image sequence is not a trivial problem, feature extractors like Harris corner detector have minor problems with repeatability resulting in features disappearing and reappearing.

Tests have shown that a simple tracker, like nearest neighbor is not reliable enough [7]. To successfully track features in an image a more advanced algorithm is required, possibly where information of the feature neighborhood is known.

A factor which makes it even harder is that we have a resolution of 0.7 milliradians per pixel which at one meter distance corresponds to approximately 1 mm, making minor vibrations result in large displacements of features in the image.

A common method of improving the matching performance is to use feature descriptors. Feature descriptors provide more information about a feature, by including neighborhood data. The descriptor makes the features more distinctive and unique. Even though the stereo matching problem is simplified, descriptor based algorithms require quite a lot of computations. SIFT based navigation systems as an example, often can not manage more than a few frames per second on a regular desktop PC. The most common way of performing stereo matching is by using statistical methods which are not deterministic.

We choose an approach to the stereo matching problem which does not require feature tracking in an image sequence and no statistical methods.

5.5.2 Spurious matching and landmark evaluation

To match a feature in the left image with a feature in the right image is known as the correspondence problem. A common approach is to use a correlation

window around the features and with a statistical method calculate a matching score. The score with the highest value is the most likely to be the correct match. To successfully use statistical methods it is necessary to calculate the matching score for many different matching pairs to find the match with the highest possible score. It is also necessary to find the outliers or false matches.

Our approach is adapted for a real-time vision system where the data is processed as a stream. No image is stored as a whole, line buffers are however used.

A feature appearing at pixel row n in the left camera must appear, if existing, on row $n \pm m$, where m is a camera calibration accuracy value which under the condition that the camera distortion is corrected and that the cameras are perfectly aligned is equal to one, because of the discrete pixel values. The horizontal limitations can be found by knowing the attitude of the cameras. The search window denoted $W_m(F_i)$ represents the maximum area in which a feature in the right image must be located to correspond to feature F_i in the left image.

By matching every feature F_i in the left image with every feature within $W_m(F_i)$ in the right image we get a set of possible landmarks $LMK(F_i)$. Within this set of 3D coordinates there can be only one that corresponds to the actual landmark, which one is unknown. We call this spurious matching. Instead of trying to find the correct stereo correspondences, we try to find which landmarks in the environment are the correct ones. While moving the robot, measuring the location of the robot using wheel based odometry, and continuously calculating the possible landmark location for every feature F_i, the reappearing landmarks are then put in a landmark database with an increasing confidence related to uniqueness and stability of the landmark location.

To rely on odometry can be risky because it is a relative measurement system with no point of calibration. Wheel slip can cause huge faults, which can be hard to recover from. For shorter distances, less than one meter, the accuracy provided by wheel based odometry should be sufficient. As soon as enough landmarks has been located with good confidence the odometry system can be used solely as a support system and is no longer required for the vision based navigation, which can be used for visual odometry.

When a number of landmarks has successfully been located it is unnecessary to try and relocate them in the manner described above. By predicting the robots location and attitude before each iteration, using for example a Kalman filter, we can find the pixel coordinate for each possibly visible landmark and exclude those features from the images. This reduces the amount of features in the images which need to be matched.

Another way of reducing the amount of possible matches is to adapt the discrimination level according to how many features was detected in the previous images in order to have a sufficient amount of features.

The camera system is a resource limited system. A navigation system like this will collect many landmarks, requiring large amounts of preferably volatile memory so that data can be retained during a power down. A robot always has a computer for controlling the high level strategy, taking actions on sensors and planning future strategies. The vision based navigation system presented here is supposed to work like an advanced sensor. The vision system can report all landmarks, confidently located in the environment, to the main computer which stores them in a database and sends them back to the vision system when they will reappear in the visual field. This approach allows the vision system to only keep a minor amount of landmarks in local storage, like block ram or SDRAM, which is available on the FPGA board.

Computational requirements

Calculating the space location of a feature pair, as seen in (5.5-5.15), requires 25 operations. Harris extracts approximately 300 corners from a 320×480 pixel frame without being too cluttered. In average this means less than one corner per line, the maximum number of corners possible on a single line is $\frac{320}{3} = 106$, though very unlikely (see section 5.4.2).

A pessimistic number of matches per feature could be around 20, which would render in 6000 landmark calculations per frame. 25 operations on 6000 landmarks would result in 150'000 operations per frame, which is less than 0.18% of what Stephen and Harris algorithm requires.

5.5.3 Experiments

In the experiments the older system based on OmniVision OV7610 cameras has been used. The OV7610 chip has a pixel separation of $8.4 \times 8.4\ \mu m$, since the same sampling method is used here, the horizontal pixel separation is doubled. The robot has moved in a straight line from 2000 mm from a single point up to the distance 230 mm. The result is shown in figure 5.7. As can be seen there are some discrepancies. The mounting of the cameras does not guarantee that they are in parallel, and separate measurements of the focal length of the lens does not give the expected focal length of 6 mm, but 7 mm. One uncertainty that has been calibrated away is lining of the cameras, i.e. that they have the same forward direction. Other uncertain parameters are lens mounting relative

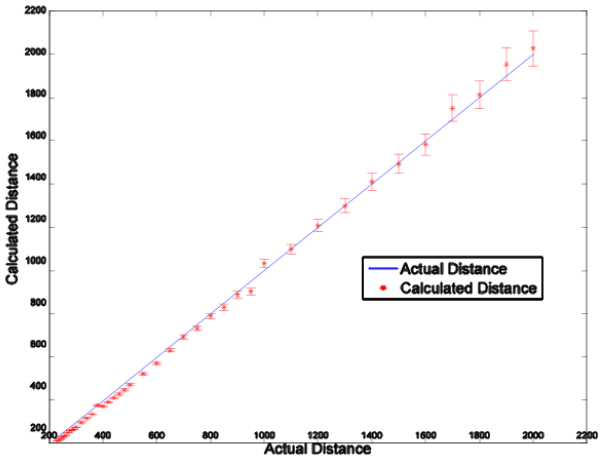

Figure 5.7: Stereo calculated distances to a single point, with plus minus one pixel error bars

to the camera chip, camera separation and camera calibration. The correction of the pointing direction was concluded from the errors at the largest distance. The separation at larger distance is smallest and any error in the orientation will have the largest effect at these distances. The correction from measurement are found to be 11 pixels by minimizing the sum of squared differences of the calculation errors. A separation offset of 11 pixels corresponds to a misplacement of the lens of 0.18 mm. The pixel coordinates are discrete values which corresponds to the pixels location plus minus half a pixel. The separation of a matched corner pair thus have a possible deviation of plus minus one pixel. The deviations are illustrated with vertical bars. It is obvious that a stereo camera system like this requires automated calibration of several parameters, such that the 3D location of a landmark can be calculated with sufficiently high accuracy for the spurious matching to be applicable.

5.6 Results

Our FPGA based stereo vision system is capable of real-time feature extraction, using the implemented Stephen and Harris combined corner and edge detector. To stereo match these features, for landmark location, is not a trivial problem.

We present a novel approach which we call spurious matching allowing us to validate which matches correspond to actual landmarks by moving the robot and extracting the features at different viewpoints.

In the current implementation of Stephen and Harris combined corner and edge detector 75 out of 150 available multipliers are used, this could easily be reduced to 25 by sharing multipliers in the factorization step of the Harris algorithm. For performance results of the corner detector see tables 5.2 and 5.3. See table 5.4 for frame rates of Harris corner detector on our system.

Table 5.2: Computational performance of our implementation of Harris corner detector.

Op/Block	Calc of edge mask	Fact. and Gaussian filter	Calc. repsons function
Add	4	120	1
Sub	6	0	2
Shifts	0	75	0
Mul	0	75	3
Total	10	270	6

Table 5.3: Performance total of Harris corner detector at different frame rates

pixels/frame	fps	Instr./pix	Cameras	MIPS
148’800	27	286	2	2’298
148’800	34	286	2	2’894

Table 5.4: Performance of our implementation of Harris corner detector.

Frame size	**Cam freq.**	**FPGA freq.**	**FPS**
320×480	96MHz	100MHz	65 fps*
320×480	50MHz	100MHz	34 fps
320×480	40MHz	100MHz	27 fps

* Theoretical value which we have not been able to verify.

5.7 Future work

Perform tests on the new stereo camera rig and automate calibration of camera parameters. The proposed spurious matching algorithm has not been fully verified yet, there are several performance factors which need to be evaluated like, camera discrepancy, odometry precision and landmark localization accuracy.

Acknowledgements

This project is supported by Robotdalen. The authors would also like to acknowledge *Xilinx* for their kind donation of our FPGA's and design software tools, *Hectronic* for the design and manufacturing of our FPGA boards.

Bibliography

[1] P.C. Arribas and F.M.-H. Macia. FPGA board for real time vision development systems. *Devices, Circuits and Systems, 2002. Proceedings of the Fourth IEEE International Caracas Conference on*, pages T021–1–T021–6, 2002.

[2] Thomas H. Drayer, Joseph G. Tront, Richard W. Conners, and Philip A. Araman. A development system for creating real-time machine vision hardware using field programmable gate arrays. In *HICSS '99: Proceedings of the Thirty-Second Annual Hawaii International Conference on System Sciences-Volume 3*, page 3046, Washington, DC, USA, 1999. IEEE Computer Society.

[3] D.L. Cardon, W.S. Fife, J.K. Archibald, and D.J. Lee. Fast 3d reconstruction for small autonomous robots. *Industrial Electronics Society, 2005. IECON 2005. 31st Annual Conference of IEEE*, pages 6 pp.–, 6-10 Nov. 2005.

[4] Stephen Se, Ho-Kong Ng, Piotr Jasiobedzki, and Tai-Jing Moyung. Vision based modeling and localization for planetary exploration rovers. In *Proceedings of the 55th International Astronautical Congress 2004*, pages 1–11, 2004.

[5] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[6] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, volume 1, pages 430–443, May 2006.

[7] A. Bissacco, S. Ghiasi, M. Sarrafzadeh, J. Meltzer, and S. Soatto. Fast visual feature selection and tracking in a hybrid reconfigurable architecture. In *Proceedings of the Workshop on Applications of Computer Vision (ACV)*, June 2006.

[8] Herbert Bay, Tinne Tuytelaars, and Luc J. Van Gool. Surf: Speeded up robust features. In Ales Leonardis, Horst Bischof, and Axel Pinz, editors, *ECCV (1)*, volume 3951 of *Lecture Notes in Computer Science*, pages 404–417. Springer, 2006.

[9] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 27(10):1615–1630, 2005.

[10] Paul Smith, D. Sinclair, Roberto Cipolla, and K. Wood. Effective corner matching. In John N. Carter and Mark S. Nixon, editors, *BMVC*. British Machine Vision Association, 1998.

[11] Micron, Boise, ID, USA. *MT9P031 Image Sensor, Product Brief*, 2006.

[12] Hans Moravec. Towards automatic visual obstacle avoidance. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, page 584, August 1977.

[13] C. Harris and M. Stephens. A combined corner and edge detection. In *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151, 1988.

[14] Cordelia Schmid, Roger Mohr, and Christian Bauckhage. Evaluation of interest point detectors. *International Journal of Computer Vision*, Vol. 37(2):151–172, June 2000.

[15] Tristrom Cooke and Robert Whatmough. Using learning algorithms to improve corner detection. In *Proceedings of the Digital Imaging Computing: Techniques and Applications (DICTA 2005)*, page 54, December 2005.

Chapter 6

Paper B: Resource Limited Hardware-based Stereo Matching for High-Speed Vision System

Fredrik Ekstrand, Carl Ahlberg, Mikael Ekström, Lars Asplund and Giacomo Spampinato
In proceedings of the 5th International Conference on Automation Robotics and Applications (ICARA), Wellington, New Zealand, December 2011 (*to appear*)

Abstract

This paper proposes a 1-dimensional implementation of area-based stereo matching with minimal resource utilization. It achieves an acceptable disparity map without the use of expensive resources. The matching accuracy for the approach can in some extent even outperform that of its 2-dimensional counterpart. Additionally, as it excels in terms of frame rate and resource utilization, it is highly suitable for real-time stereo-vision systems.

6.1 Introduction

Image preprocessing aims at a reduction of the amount of data without losing significant information, or to elevate the information density of the collected data before forwarding it to the main application. This often involves extraction of 3-dimensional information from 2-dimensional images. Retrieving the depth information implies solving the correspondence problem, that is, matching a segment, or pixel, in two images from separate views.

Area-based stereo-matching techniques are widely accepted as one of the best approaches for generating dense depth maps for real-time systems. Several approaches exist, with numerous implementations for every approach [1], [2], [3], [4], [5]. For a comprehensive overview of the subject and techniques we refer to Scharstein and Szeliski [6]. Implementations for specific platforms, such as FPGAs, are presented by Lazaros et al. [7]. For off-line processing, the accuracy of the depth map is the most important aspect, and for real-time applications the frame rate is added as an important parameter. The maximum possible depth range detectable also frequents the performance evaluation lists, and for robotics and autonomous applications, the implementation size and resource utilization is also of great importance.

Correctly estimating the separation, or disparity, of the same point in two separate images does not necessarily infer a reduction of the amount of data. The disparity is estimated for every pixel and the bit count between an 8-bit gray-scale image and a depth map is likely to be identical if communicated off chip. Constantly sending large amounts of data off-chip will yield a heavy load on the communication lines. Since one of embedded systems biggest bottlenecks is intercommunication, it is straightforward to argue that minimizing the amount of data necessary to send off-chip would increase the throughput and the speed of the system, similar to that of image and file compression for network streaming. Retaining the data on-chip for further processing and reduction is clearly desirable.

A small and resource limited implementation will enable same-chip co-processing. However, if the implementation causes heavy loading of the memory, it will effectively limit the memory bandwidth for any co-process. Removing the need for external memory - or other limited, exclusive resource - in the preprocessing, extends the range of concurrent algorithms possible. The more data-intense the process, the more the system will benefit from a reduced utilization of shared resources [2]. The goal is thus to minimize the resource usage of the low-level correspondence part in order to increase the resources available for application specific processing.

A possible application of this minimal correspondence component is to increase the match confidence of an embedded, visual odometry-based navigation system [8]. Providing an early depth estimate for the tracked features will significantly increase the confidence in matching as well as reduce the outliers. A direct validation of the matched features will be possible when executed concurrently.

We have examined the implications of resource limitations for two extensively used stereo-matching approaches by removal of the time and resource intense memory utilization. The goal is to see if it is possible to get an acceptable disparity map with a radical reduction of the matching data. We greatly reduce the use of system resources in an FPGA-based implementation to enable extensive on-chip, hardware-based image co-processing. The proposed approach is to remove the inter-scanline dependency of standard square or rectangular area-matching techniques.

6.2 Matching Algorithms

For our implementations, we assume rectified and parallel images with a unified baseline. In a live system, a rectification process is usually required before the matching - an additional motivation for advocating minimal system occupance by the correspondence component.

Among the most common approaches in area-based matching are SAD (*Sum of Absolute Differences*), SSD (*Sum of Squared Differences*), and NCC (*Normalized Cross Correlation*). The difference in outcome for these approaches are for a basic implementation minimal, hence the complexity and system suitability are more important factors for our purpose. The SSD and NCC approaches require multipliers, an often limiting resource in FPGAs, heavy used in image rectification and other image-processing [9]. The SAD is the preferred choice due to its minimal resource utilization (no multipliers) and straightforward implementation composed of a series of operations, according to (6.1), requiring only adders and comparator - abundant in FPGAs. The I_L and I_R in (6.1) are the intensity values of left and right images at pixel x and $x - d$, respectively, with d being the disparity under consideration, and w denoting the width of the correlation window.

$$SAD = \sum_{x=0}^{w-1} |I_L(x) - I_R(x-d)| \tag{6.1}$$

Another similar approach popular for hardware implementations, due to its robustness to radiometric distortion and straightforward implementation (no multipliers), is the Census Transform [10]. The Census Transform is not a matching algorithm, but a non-parametric local transform evaluating the local structure of the area. It is acting like a preprocessing step to enhance the image before matching.

The intensity value of each pixel in the image is replaced by a bit string composed of the result of a boolean comparison between itself and its surrounding pixels according to comparison function (6.2). ζ is a bit to be placed in the string, and is decided by the pixel intensity values i_s in relation to the center pixel's intensity i_c in the center pixel neighborhood.

$$\zeta(i_c, i_s) = \begin{cases} 1, & i_c < i_s \\ 0, & i_c > i_s \end{cases} \tag{6.2}$$

The bit-string values are then correlated between the two images for every pixel within a matching window by calculating the Hamming distance, which is the number of differing positions between two strings of equal length. In both the Census and SAD approaches, the best match is found by finding the least sum of differences.

Woodfill and Zabih [10] claims that the relative ordering of the intensity values of the Census algorithm is less susceptible to noise and outliers, and matching implementations using the transform are reported to outperform standard correlation techniques, such as SAD and SSD [11],[12].

6.3 Related Work

SAD and matching using a Census-transformed image can be categorized as area-based correspondence methods. The matching window for area-based approaches are usually a square, and the matching localization improves with increased window size. The reverse holds for the accuracy, especially in areas with sharp depth-discontinuities, due to the averaging effect of the window. However, as the shift between the two images to be matched is assumed to be purely horizontal, the vertical information holds less weight in the disparity estimation.

Lee [13] concludes that a rectangular window for SAD correspondence saves close to half of the implementation cost without sacrificing the quality. Although stating that the surrounding vertical pixels only function as noise

reduction, and presenting matching scores for various window sizes and ratios, Lee does not show any results for 1-dimensional windows.

Calin describes an implementation of a 1-dimensional disparity map estimation by use of SSD [14]. It is capable of producing dense disparity maps of 160x120 pixel images at 30 fps running in an FPGA. The presented results are moderate as the disparity objects are excessively bloated, as to be expected when using a wide correlation window, and the depth resolution is limited, partially due to the small image size.

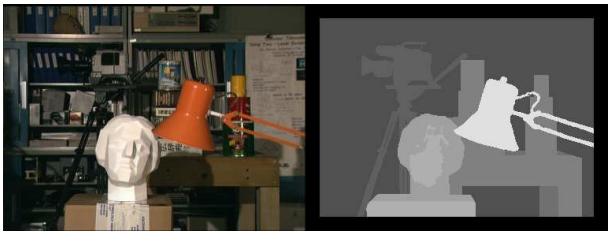

Figure 6.1: The Tsukuba original and its groundtruth.

6.4 Implementation

We implemented 1-dimensional versions of both the SAD and the Census methods in VHDL for execution in an FPGA. As a comparison, we also implemented the regular square-window approaches for both methods. Additionally, we verified the implementations in MATLAB and received practically identical results as for the VHDL.

6.4.1 SAD

Our implementation of the SAD algorithm is straightforward. The difference calculations are performed over the entire disparity range in parallel through multiple instances of the pixel-wise AD calculation unit. The input to the ADs are the latest pixels from the left image according to the window size, together with the in registers stored pixels for the disparity range from the right image. The least sum is then found through a tournament selection process.

For the square implementation the window-generator from [8] was used and the above implementation multiplied by the window size and run in parallel.

6.4.2 Census

As a local transform, the Census Transform rely heavily on its neighbors and minimizing the neighborhood significantly reduces the accuracy of the approach. The fact that the Census correspondence is calculated in two separate window-sets - 1: the transform; 2: the Hamming distance - makes the information reduction two-fold and causes the performance to deteriorate more than for the SAD. To compensate for this loss of data, we substituted the intensity with its underlying RGB values and separated them into parallel matching channels. Thus the matching data is increased three-fold (together with the resource usage), and an increased matching accuracy can be observed in figure 6.2 (see next chapter for how to evaluate the image). This contradicts the data presented by Bleyer [15]; that substituting intensity with color in the Census Transform will not improve the matching. The issue needs further investigation, but it is evident in our matches that color yields an improved result. However, with increased window width, this difference between intensity and RGB diminishes, although still remains.

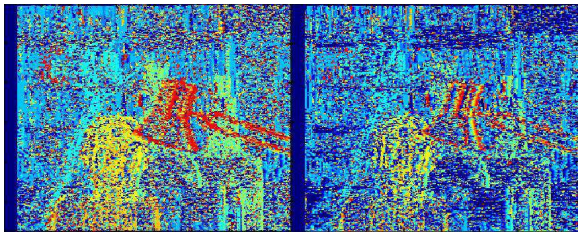

Figure 6.2: The 1-dimensional Census Transform matching improves with RGB (left) over intensity (right).

The 1-dimensional Census implementation is similar to the SAD with the main difference being the three parallel matching stages for handling the color channels in the Census. The image data is shifted into three arrays sized according to the matching window. The Census Transform is performed in parallel for all pixels, and the referencing is for the same channel center pixel only, no cross-color evaluation. The Hamming distance is then performed in two steps for all disparity positions for all channels in parallel - 1: the channels are individually compared; 2: and subsequently summed for every possible disparity in the range. The last step is to select the minimum value within the disparity range.

The square Census implementation is using the same implementation with the RGB channels multiplied with the window size and filled with data from the previously mentioned window generator. This significantly increases the resource usage, which requires a dramatic improvement to be warranted.

Figure 6.3: The colormap for the disparity images. Blue indicates a small disparity, and red a big disparity.

6.5 Results

The commonly used stereo pair from the Tsukuba University [16] was used to evaluate the implementations at an image size of 384 by 288 pixels. The original scene is shown in figure 6.1 together with the ground truth, where lighter means closer, supplied with the test images. The disparity maps presented in this paper have been colormapped according to figure 6.3 to better illustrate the differences in disparity. The range of the colormap represents the disparity in the image, the blue color is a small disparity (futher away) and the red is a large disparity (closer).

As opposed to the results from Hirschmüller [12], the Census correspondence is outperformed by the SAD in the standard implementations. For larger windows the differences are marginal, but for smaller window sizes the differences increase, as can be seen in figure 6.4.

The Census disparity maps are far more noisy, which is even more prominent in the 1-dimensional implementations. The performance reduction resulting from flattening the window height to a single row can be seen in figure 6.5, where resulting images from two matching windows of the same width but differing height for both SAD and Census are shown. The same type of reduction exists in both approaches, but it is more evident in the Census. It appears that the Census Transform suffers more from the window reduction than the SAD does.

However, as the width of the windows increase the differences diminish, and although the 1-dimensional SAD version suffers from a loss of surface continuity, it is actually better than the square version in terms of precision for valid pixels, especially around object boundaries, as can be seen in figure 6.6.

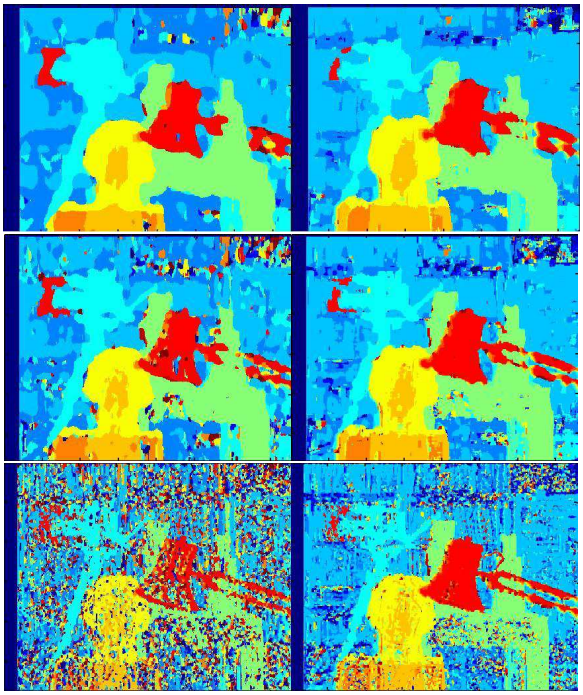

Figure 6.4: Square matching window versions for Census (left) and SAD (right). The window sizes are, from the top: 11, 7 and 3.

This improvement is due to the absence of vertical smoothing generated by the multiple scanlines of a square window.

Table 6.1 shows the resource utilization for standard implementations versus flattened, for the SAD and the Census at a maximum disparity of 16 on images of 384 by 288 pixels. The listed resources are for the correspondence components only, the surrounding system components used for reading, storing and sending of the test images and match data are not included. In a more extensive implementation of the 2-dimensional SAD matching method, a large memory is usually included in the design. Its purpose is to hold the pixel and disparity data of previously processed pixels in order to perform right-left consistency check, to further remove matching errors and improve the disparity map. Thus, the real implications of flattening the correlation window are even greater for the resource utilization, as been discussed previously, than the direct

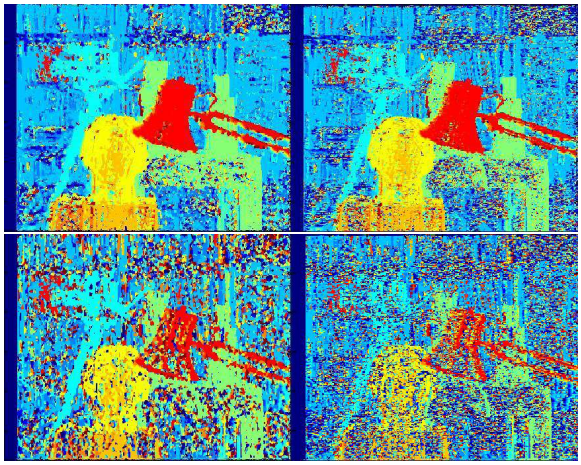

Figure 6.5: The information loss for SAD (top) and Census (bottom) when reduced from 3x3 (left) to 1x3 (right).

reduction shown in table 6.1.

Note: For the purpose of evaluation, external memory is used in the implementations to store the test images. However, those resources are not included in the data of resource utilization.

6.6 Conclusions

In this paper we propose an approach to a limited correspondence implementation. It is limited in the use of system resources to enable extensive on-chip/embedded high-level post-processing. The implementation is based on one of the most commonly used approaches for stereo correspondence, the SAD (Sum of Absolute Differences).

The limited implementation of the SAD performs remarkably well, and even challenges the original square window version for larger widths, due to the absence of smoothing errors brought on by the averaging effect of the window. It is evident that although there is a reduction in disparity map quality, it is possible to achieve acceptable disparity maps without extensive memory usage. With further post-processing or inclusion of a cost-function for disparity continuity, the result may be improved toward that of the costly square

Resource	Slices	LUTs
SAD 3x3	1,434	5,581
SAD 1x3	611	1,746
SAD 1x7	761	3,489
Census 3x3	2,013	11,676
Census 1x3	1,035	3,553
Census 1x7	1,579	7,489
Available	**33,280**	**33,280**

Table 6.1: Resource utilization

window-matching implementations. Even without this post-processing, the 1-dimensional implementation is more than adequate for autonomous applications such as navigation, mapping, and obstacle avoidance. It is particularly suitable for applications requesting sparse depth information, such as for features, by providing a confident match in salient regions.

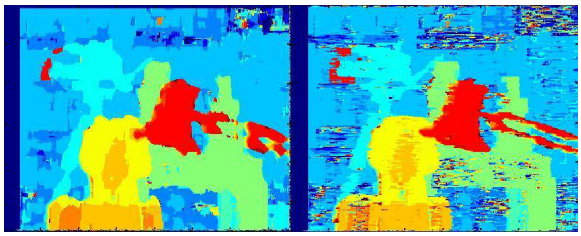

Figure 6.6: The SAD versions for 11x11 (left) and 1x11 (right).

Operating this approach on larger images, even of several Megapixels, will not affect the throughput or the resource utilization. Image data is only stored in registers in the FPGA and will not demand any external memory or other resource utilization regardless of the image size. It is implemented with a maximum disparity range of 64 for a window size of 21 pixels.

The implementation run at 125 MHz, the system clock of our FPGA-board [8]. As the implementation is fully piped, the frame rate is dependent on the speed of the cameras and the size of the frame. Theoretically, it is capable of processing over 100 frames per second for Megapixel images.

Acknowledgements

The authors would like to acknowledge *Xilinx* for their kind donation of our FPGA's and design software tools, *Hectronic* for the design and manufacturing of our FPGA boards, and *The Knowledge Foundation* for providing funding for the project.

Bibliography

[1] B. Boufama and K. Jin. Towards a fast and reliable dense matching algorithm. Technical paper, School of Computer Science, University of Windsor, 2002.

[2] Kristian Ambrosch, Wilfried Kubinger, Martin Humenberger, and Andreas Steininger. Flexible hardware-based stereo matching. *EURASIP Journal on Embedded Systems*, 2008:1–13.

[3] D.K. Masrani and W.J. MacLean. Expanding disparity range in an FPGA stereo system while keeping resource utilization low. *Computer Vision and Pattern Recognition, 2005 IEEE Computer Society Conference on*, 3:132–132, 20-26 June 2005.

[4] J. Woodfill, G. Gordon, D. Jurasek, T. Brown, and R. Buck. The tyzx deepsea g2 vision system, a taskable, embedded stereo camera. In *Proceedings of the IEEE Computer Society Workshop on Embedded Computer Vision, Conference on Computer Vision and Pattern Recognition, (New York, NY), June 2006*. IEEE, 2006.

[5] Ahmad Darabiha, Jonathan Rose, and W. James MacLean. Video-rate stereo depth measurement on programmable hardware. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 1:203, 2003.

[6] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47:7–42, 2002.

[7] Nalpantidis Lazaros, Georgios Christou Sirakoulis, and Antonios Gasteratos. Review of stereo vision algorithms: From software to hardware. *International Journal of Optomechatronics*, 2(4):435–462, 2008.

[8] Jörgen Lidholm, Fredrik Ekstrand, and Lars Asplund. Two camera system for robot applications; navigation. In *13th IEEE International Conference on Emerging Technologies and Factory Automation, 2008. (ETFA 2008)*, pages 345–352. IEEE, September 2008.

[9] J. Lidholm, L. Asplund, M. Ekstrom, and G. Spampinato. Hardware support for image processing in robot applications. Technical paper, School of Innovation, Design and Engineering, Malardalen University, 2011.

[10] R. Zabih and J. Woodfill. Non-parametric local transforms for computing visual correspondence. In *European Conference on Computer Vision*, pages 151–158, 1994.

[11] H Hirschmüller, P R Innocent, and J Garibaldi. Real-time correlation-based stereo vision with reduced border errors. *International Journal of Computer Vision*, 47(1):229–246, 2002.

[12] Heiko Hirschmüller and Daniel Scharstein. Evaluation of stereo matching costs on images with radiometric differences. *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 1582–1599, 2009.

[13] Lee Sunghwan, Yi Jongsu, and Kim Junseong. Real-time stereo vision on a reconfigurable system. *Lecture Notes in Computer Science*, 3553:299–307, 2005.

[14] G. Calin and V. O. Roda. Real-time disparity map extraction in a dual head stereo vision system. *Latin American applied research*, 37:21 – 24, 01 2007.

[15] Michael Bleyer and Sylvie Chambon. *Does Color Really Help in Dense Stereo Matching ?* Number 1. 2010.

[16] Yuichi Nakamura, Tomohiko Matsuura, Kiyohide Satoh, and Yuichi Ohta. Occlusion detectable stereo – occlusion patterns in camera matrix. In *IN CVPR*, pages 371–378, 1996.

Chapter 7

Paper C: Utilization and Performance Considerations in Resource Optimized Stereo Matching for Real-Time Reconfigurable Hardware

Fredrik Ekstrand, Carl Ahlberg, Mikael Ekström, Lars Asplund and Giacomo Spampinato
Technical Report

Abstract

This paper presents a quantitative evaluation of a set of approaches for increasing the accuracy of area-based stereo matching methods. It is targeting real-time FPGA systems focused on low resource usage and maximized improvement per cost unit to enable concurrent processing. The methods are applied to a resource optimized correspondence implementation and the individual and cumulative costs and improvements are assessed. A combination of the implemented approaches perform close to other area-matching implementations, but at substantially lower resource usage. Additionally, the limitation in image size associated with standard methods is removed. As fully piped complete on-chip solutions, all improvements are highly suitable for real-time stereo-vision systems.

7.1 Introduction

Applications such as navigation and object avoidance require real-time information about their environment, and vision is a good provider due to its versatility and speed. This visual perception includes depth information, implying real-time matching of multiple view-separated 2D images, preferably from a stereo camera setup due to the reduced complexity with a known relation between the images.

The extraction of depth data through the localization of the same point in two images is not trivial. Stereo matching of an entire scene is computationally intense, and processing a scene, or an image pair, 30 times per second (real-time) is computationally demanding, and require high-performing hardware. Implementations range from regular computers, to specialized hardware such as GPUs and FPGAs. Lazaros et al. [1] make a thorough presentation of various implementations.

FPGAs, often referred to as reconfigurable parallel hardware, are utilized in mobile applications using vision, as they outperform other approaches in terms of speed, size, and power requirements [1]. The major obstacle is the limited resources, which restricts which algorithms are possible to implement. Non-iterative approaches with small memory footprints, such as pipelined processing of streamed high frame-rate camera data, are preferred.

Stereo matching is a well-covered problem with many suggested approaches, and a comprehensive overview on the subject is presented by Scharstein and Szeliski [2]. In general, the approaches are divided into global and local methods. Global approaches consider the entire image when estimating the disparity, usually with high accuracy. Their iterative nature and high memory bandwidth requirements [3] make them not optimal for FPGA implementation. Local approaches consider only small parts of the image, called support windows, which are matched using a correlation measure. Basic local algorithms do not attain the accuracy of global methods, but they have been the preferred real-time stereo matching approach for a long time due to ease of implementation and speed [1].

Two popular correlation measures of local methods for stereo correspondence are SAD (*Sum of Absolute Differences*) and SSD (*Sum of Squared Differences*). Both are of similar performance [4] and straight-forward to implement in an FPGA, but the SAD consumes less resources than the SSD [5], an important factor in hardware systems.

An important parameter with hardware implementations is the limited resources available. A complete vision system residing in an FPGA requires

several processing components just for preprocessing the image. For instance, image rectification, motion compensation, and depth estimation are all low-level tasks. Additionally, higher-level tasks, such as tracking, object recognition, or navigation, should also fit. Enabling more computations in the FPGA, by reducing the components for preprocessing, will improve the capability and flexibility of the system.

We examine the impact of heavily reducing the resource usage of a stereo matching approach in [6]. The 1D implementation shows an acceptable result for the application at hand - disparity estimation around edges - at a very small footprint. The natural drawback is excessive noise, especially in known difficult regions of low-texture and/or low signal-to-noise ratio. Positive aspects, besides the resource usage, is the removal of external memory regardless of image size, and the removal of the image size restriction implicit in 2D approaches - a larger image does not increase the resource usage of the correspondence component. In this paper we will present the improvements gained and the costs incurred by a number of resource optimized implementations of established approaches for improving this basic correspondence method. An evaluation can then be made on whether these improvements are justified considering the increased resource usage, depending on the needs of the targeted application.

This work is part of the Two Camera-project at Malardalen University. The aim is to construct a compact, vision-based autonomous system encompassing both sequential and parallel processing units [7]. The majority of the low-level data-intense processing will reside in an FPGA, while high-level computation-intense processes are handled in an embedded sequential computer. The code composing the components in this paper will be made available as an open source project to promote FPGA-based image processing on our publicly available vision system.

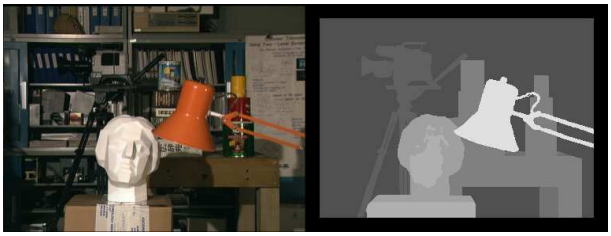

Figure 7.1: The Middlebury test image Tsukuba with the associated ground truth.

7.2 Background

This section will look at some common approaches of improving basic area-based matching and how they can be adopted to our system. The end-goal is to incorporate as much of the processing for an autonomous system as possible in the FPGA. In order to encompass concurrent processing, priorities and trade-offs need to be made. As a result, we have adopted a system approach to stereo matching with the explicit aim of reducing system impact with an implementation. We have constructed an FPGA-based system [8] for autonomous machine applications, and implemented a resource optimized basic stereo matching algorithm [6].

Local correspondence algorithms consider a limited area surrounding the point under evaluation for disparity estimation - the support window. The SAD matching cost for two points residing in two different images is calculated through an aggregation of the element-wise absolute differences of the support windows for respective point. A common assumption of stereo-matching algorithms is that of rectified and parallel images with unified baseline and common scan-lines. With this assumption it is safe to reduce the correspondence problem to a 1-dimensional search [2]. As such, correlation is performed by evaluating the difference between windows in different images but of same scanline, no disparity perpendicular to the scanlines needs to be taken into account. For all approaches, we assume rectified and parallel images with a unified baseline.

The implementations are evaluated using the stereo images and online tool provided by the vision department at Middlebury University [9]. One of the images used in our tests is shown in figure 7.1 along with its true disparity map. The disparity maps generated by our implementation are using a maximum disparity range according to the stated disparity range of the evaluation images.

In [6], we show that SAD is a good candidate for a resource optimized correspondence implementation for real-time systems. The 1D approach produces a disparity map with preserved details and reduced foreground fattening compared to the standard 2D implementation. The increased noise is primarily located in low-texture areas with low signal-to-noise ratio, as can be seen in figure7.2.

7.3 Related Work

Since the aim of our work is to achieve an acceptable performance at a low resource usage, we need to specify what low resource usage is. An FPGA

Figure 7.2: The areas with the most noise (red circles) have low-texture and/or low signal-to-noise ratio.

consists of different elements that can be configured in a multitude of ways. Resource utilization is normally expressed in slices and LUTs (LookUp-Table which realize boolean operations). The 1D implementation from [6] produced the disparity map of figure 7.2 from 1221 slices when implemented in a Spartan-3 FPGA. This is just above 4% of the available slices in the chip.

Several other stereo matching approaches with low resource usage exists, such as the one proposed by Arias-Estrada et al. [10]. The utilization is only 4.2K slices on a Virtex-II, but the disparity map is only fair. The implementation is capable of 71 fps with images of 320x240 pixels. Lee et al. [11] present an implementation below 10K slices in resource usage. The resulting disparity map is moderate with extensive blurring of edges and noise. For higher quality disparity maps, the resource usage goes up. Very good results are presented by Zhang et al. [12], but the utilization is 95K slices, which is 3 times the total number of slices available in our FPGA. Additionally, they use a large amount of special ALUTs and DSP blocks, leaving little room for concurrent processing.

A comparison between software- and hardware-based correspondence algorithms, show a clear separation of focus [12]. In general, software-based methods produce much more accurate disparity maps, but fail to produce real-time output at high quality. Real-time software implementations generally do not produce much better result than hardware-based ones. On the other hand, hardware-based implementations produce frame rate far exceeding those of software-based. This is starting to change, though, as software-based algorithms are being transformed to fit hardware, and software-based systems are starting to achieve high throughput by utilizing GPUs. However, PC and GPU approaches are still too bulky for embedded systems.

7.4 Improvements

We will now present a set of common approaches to improve disparity images. They are not specifically oriented at reconfigurable hardware, but are more or less established in the area of stereo matching.

7.4.1 Support Window

Being rudimentary for the performance of area-based approaches, the window size selection is of great importance. Generally, smaller windows increase the matching precision, but exhibit more noise. Consequently, large windows reduce noise and produce better results in low-textured areas, but blur in areas of depth discontinuities, and cause an effect known as foreground fattening [13]. Common window sizes for the 2D SAD range from 9x9 to 17x17 [13]. Beyond this size the accuracy is actually starting to degrade, as shown by Ambrosch et al. [14], and there is no global optimal selection due to the scene dependency.

Several proposals for improving the lack of conformity of square static windows to the dynamic conditions of a scene [3] have been proposed, such as adaptive [15], multiple [4], or weighted [16] windows. However, the adaptive window approaches suggested by Kanade and Okutomi [15] or the one by Boykov et al. [17] does not offer much improvement, only different problem areas (ill-defined edges, noisy surfaces, sensitivity to low-texture areas, non-realtime performance). Additionally, they rely on models with empirically derived parameters unique to every scene, which is not a huge improvement from the empirical selection of window sizes. More recent adaptations of this notion produce better results, such as the work by Sun et al. [18], but they are still iterative and slow.

The approach by Hirschmüller et al. [4] suggests the use of multiple windows for good depth discontinuity performance. It is based on SAD, but unfortunately requires high memory bandwidth. The bi-level window refinement proposed by Chonghun et al. [19] first calculates a disparity map using a large 16x16 window and then refines it with a small 5x5 window. Their reason for refinement is that of an overly-smoothed first estimation, which is rather noise-free but blurred, as opposed to our noisy first estimation.

7.4.2 False Matches

False matches occur from the fundamentals of matching two images from different viewpoints. Parts of the image which are visible in one view are not

visible in the other because of projective distortion, Kanade and Okutomi [15]. Fusing two views together will leave areas where depth estimation is impossible as they are occluded in one of the views. Other conditions, such as lighting and image noise, give rise to ambiguities in the matching. This affects all area-based approaches, but is even more evident for smaller support windows as they have lower signal-to-noise ratio. Post-processing of the estimated disparity map is usually adopted to remove noise and false matches. Popular post-processing methods include left-right consistency check (LRC) [20], propagation, confidence evaluation and median filtering.

7.4.3 Consistency Check

The left-right consistency check verifies that only disparity values with mutual correspondence are accepted as matches. Mutual correspondence occur when the matched pixels in the two images select each other as the best match. This requires matching of the whole image, using both images as the reference to account for pixels not visible in the other view. The check invalidates false matches, and either simply voids the disparity value or employs a subsequent correction stage, as detailed by Fusiello et al. [21].

The left-right consistency constraint is very robust but expensive as it requires a second pass over the images [22]. A common implementation approach, as used by Mühlmann et al. in [23], is to store all matching costs for all possible disparities as a Disparity Space Image (DSI). The DSI cuboid consumes a large amount of memory as its volume is given by the width and height of the images and the disparity search range. The left-right consistency check is straightforward when using a DSI by simply traversing the cuboid, but time consuming.

7.4.4 Confidence Evaluation

Areas of low-texture exhibit low variance and are difficult to match as unique identifiers do not exist. A common approach to solve this is to look at the matching confidence of the estimated disparity [18]. The confidence is often estimated by relating to the difference in correlation cost of the best matching candidates. Mei et al. [24] adopt an approach that compares the estimated disparity with that of its previous neighbor. The concept is that the disparity is supposed to vary smoothly for but a fraction of the entire image, according to the surface continuity constraint formulated by Marr and Poggio [25]. Mei et

al. use this constraint-based smoothness term in a cost-function to select highly confident matches to be used in a well-performing propagation approach.

7.4.5 Textureless Areas

Textureless areas and areas without sufficient information for a confident match cannot be matched, and their disparity must be estimated otherwise. Non-consistent matches are labeled to be either removed or improved in a subsequent correction component. For dense disparity maps, the replacement option is the preferred, but it is also the most difficult due to occlusion. A common approach is to interpolate or propagate disparity values from nearby confident matched pixels. Yoon et al. [5] perform a spatial interpolation after error reduction by the use of median filtering. Sun et al. [18] propose a propagation approach where the propagated value comes from a pixel of similar color. The approach performs well, however, they rely on a relatively good first estimate by way of the Census. The approach is not real-time due to iterative processing and time-consuming segmentation. A simpler approach suggested by Fusiello et al. [21] fills non-valid positions with the minimum value in a propagation window. The minimum value is used to limit the foreground fattening common in area-based methods, which hopefully have been removed with the left-right consistency check. The approach successfully fills areas cleared by the consistency check, but it too is dependent on a confident initial estimate.

7.4.6 Filtering

The obvious improvement that was left out in our previous work is the application of a filter. Median filtering is a well-known approach to remove sporadic noise and is frequently used in post-processing to improve disparity maps [23]. Several of previously mentioned implementations used a median filter: [5], [15], [13].

7.4.7 Color

A natural step from the minimal approach is to match color values instead of intensity. It has been reported that the use of color information can increase the signal-to-noise ratio by roughly 20% [26], which should substantially improve the matching. Implementation of an RGB-based matching is accomplished simply by invoking three parallel SAD channels and sum the individual SADs

for every disparity. The cost, however, is substantial as we triple the amount of computations performed in parallel and thus also the resource usage.

7.5 Our Implementation

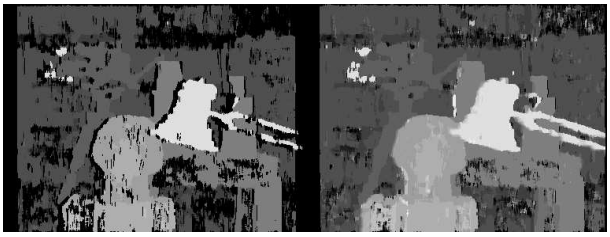

Figure 7.3: Impact of applying LRC (left), or not (right). Both images have been median filtered as the last stage.

7.5.1 Support Window

All of the suggested window-altering approaches could improve on the basic 2D implementation, but have limited impact on the 1D implementation. The much larger area in square windows make them lose precision, and adapting the window shape and size is a way of reducing this loss. For 1D windows the effect is much smaller because of the reduced area. Additionally, adaptive approaches are slow and introduce scene dependent parameters that add to the complexity and reduce the application scope. Transformation of the support window is primarily aimed at improving the matching accuracy, but not with a retained speed. Moreover, our challenge lies in too much noise, which voids the stated approaches as they look for refinement and we aim for selective smoothing.

7.5.2 Consistency Check

The basic memory-intense approach of left-right consistency check is neither suitable for a parallel or resource constraint system, nor necessary. The check will always be relatively expensive, as it needs to match the images both ways, it is not necessary to retain more data than covers the matching of the pixels

in the disparity range at any given time. Our implementation practically doubles the resource usage for the matching process (1221 vs 2399 slices), but no external memory nor any reduction in system performance need to result from it.

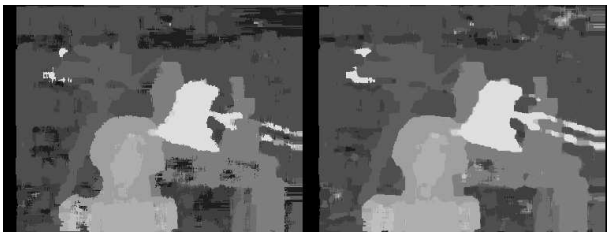

Figure 7.4: The median filter preceding the propagation (left) does not perform as well as the reverse (right). LRC-check was performed initially on both images.

At any given clock cycle, a SAD is calculated for every pixel in the disparity range in the right image. These SADs make up the disparity space for one pixel in the left image. The least sum is then extracted from that disparity space through a tournament selection. The SADs for a pixel in the right image are calculated over time as it traverses the disparity range, one step every clock cycle. The least sum for a pixel in the right image is then calculated piece-wise by a simple compare function. Register delays assure synchronized output with a disparity for the left pixel after cleared check.

The effect of the consistency check can be observed in figure 7.3. The images are with and without consistency check, but both have median filtering performed at the end, to minimize the empty regions. The consistency check identifies almost all of the occluded areas. However, it also removes pixels that are not occluded but still differ due to poor correlation data. Noteworthy is the deterioration of the lamp arm, partly due to the check but also due to the filter. The removal of data in the disparity map reduces the quality, and it is evident that the median filter (here a 7x1) is not filling the empty areas. For this to happen we need to propagate.

7.5.3 Propagation

With propagation, the underlying data is important. A logic assumption is that it is important to remove as much noise as possible before performing

propagation, to avoid propagating false matches. As can be seen in figure 7.4, there is a difference between performing median filtration before or after the propagation. Propagation directly after the consistency check followed by a median filter produces a disparity map of the highest accuracy. However, some areas deteriorate, such as the lamp arm, when compared to a not consistency checked image.

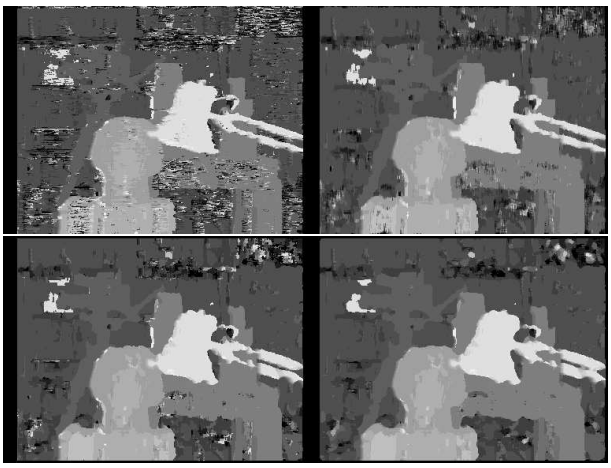

Figure 7.5: The median filter (right column) significantly reduces the noise for the 1D (top) but not the 2D (bottom) approach.

The propagation rules covering the execution of our implementation are: If the median value of the filter window is separate from zero (that is, a majority of the pixels in the window have valid data), the median of the non-zero values is propagated. Should the number of non-zero elements in the filter be subordinate, the minimum value will be propagated. When no non-zero values exist, the latest propagated value is used. Since the propagation relies on median value it is realized with a copy of the median filter component.

7.5.4 Filtering

Realization of median filtering is a search and rank problem with large filters being difficult to implement for real-time [27]. However, filters of limited size have been utilized with good results [23]. We have implemented the median filter as a classic systolic array, according to Vega-Rodriguez et al. [27], for

Approach	Non-Occ %	All %	Disc %
SAD 7x1	29.1	30.7	27.4
SAD-MED 7x1	22.2	23.9	24.3
SAD 7x7	22.7	24.4	28.6
SAD-MED 7x7	21.9	23.6	28.1

Table 7.1: Performance and cost for median filtering. The % is the errors in the image compared to the ground truth for Non-occluded, All, and regions near Discontinuities.

sorting 9 elements. This translates to a 9x1 median filter for 1D and a 3x3 filter for 2D.

The improvement with a median filter are quite significant for the 1D approach, but not so much for the 2D, as can be seen in figure 7.5 and in table 7.1. The filter removes noise and the 2D implementation is already noise reduced by design. It is obvious that the noise in the 1D approach fits the characteristics of a median filter. Noteworthy is the fact that the 1D approach outperforms the standard 2D in regions of discontinuities, due to the lack of vertical summing. This is the case already with the basic 1D, but is even more improved with the added filter. The cost of the filter is very low, only 247 slices, an increase of 20%. As a conclusion, median filtering closes the gap between 1D and 2D implementations.

7.5.5 Color

The effects of using separate color channels instead of intensity are presented in figure 7.6. As can be observed, the substantial increase in resource utilization for color (double for matching component) is not warranted unless it is critical for the application, such as for scenes with radiometric distortion as shown by Hirschmüller and Scharstein [28].

7.6 Result Summary

Table 7.2 shows the improvement for the stereo matching component with the implemented approaches, both individually and combined. The listed resources are for the evaluated stereo components only, the system components used for reading, storing and sending of the test images and match data are not

Approach	Non-Occ %	All %	Disc %	Slices	LUTs
SAD	29.1	30.7	27.4	1,221	6,086
LRC	40.5	41.9	40.1	2,399	7,689
Median	22.2	23.9	24.3	1,468	6,371
LRC-Med	38.6	39.9	39.5	3,135	8,204
LRC-Prop	27.2	28.4	24.9	3,174	8,237
LRC-Med-Prop	31.1	32.0	28.8	3,986	8,844
LRC-Prop-Med	20.4	21.8	21.7	3,986	8,844
Available				**33,280**	**33,280**

Table 7.2: Impact of improvements; individually and combined. All values are for a 7x1 implementation

included. The improvements are evaluated with the Middlebury stereo evaluation tool [9] which show the error percentage in the disparity image. Three different parameters are presented: Non-occluded pixels which are visible in both images; Pixel at or around discontinuities in the image; All of the image. From observing the matching scores in table, it can be concluded that certain tools have certain aspects, and when combining tools for improvement, their individual order is important.

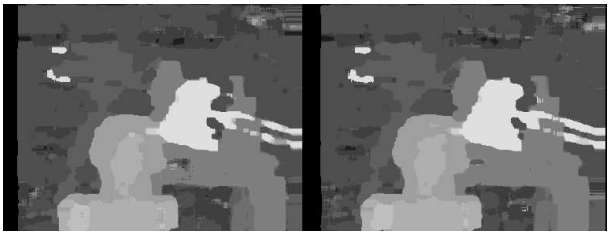

Figure 7.6: Substituting intensity (left) with color (right) does not improve significantly to warrant the doubling in resource usage.

Note: For the purpose of evaluation, external memory is used in the implementations to store the test images. However, those resources are not included in the data of resource utilization, nor do they affect the performance.

7.7 Conclusions

In this paper we assess approaches to improve a resource optimized correspondence implementation. The approaches have been evaluated quantitatively by showing the cost and quality enhancement achieved when implemented. The 1D implementation is approximating the accuracy of the standard 2D version by a set of established enhancement methods. The performance difference between the 1D and 2D implementations is rather small, but the 1D achieves it at a low resource usage.

Utilizing an inexpensive median filter effectively closes the gap to the 2D approach. From a cost/performance perspective, only using a median filter is the best approach. However, there is only so much a 1D median filter can do with noisy data. For further improvement noise reduction is a must. A function removing, or never allowing, false matches in the disparity map, through confidence assessment, could render a substantial improvement together with a competent propagation method. Implementing a small confidence measurement would be a good continuance of this work.

It is further evident that it is possible to achieve acceptable disparity maps without extensive memory usage and without a limitation on image size. Megapixel images will not affect the throughput or the resource utilization. Image data is stored in a shift register approach without the need for multi-scanline retention. Furthermore, the proposed 1D implementation is more than adequate for autonomous applications such as navigation, mapping, and obstacle avoidance, and can be fitted to practically any FPGA. It has been implemented with a maximum disparity range of 64 for images of 1024x1024 pixels.

The implementations run at 125 MHz, the system clock of our FPGA-board [8]. As the implementations are fully piped, the frame rate is dependent on the speed of the cameras and the size of the frame. Theoretically, it is capable of processing over 100 frames per second for Megapixel images.

Acknowledgements

The authors would like to acknowledge *Xilinx* for their kind donation of our FPGA's and design software tools, *Hectronic* for the design and manufacturing of our FPGA boards, and *The Knowledge Foundation* for providing funding for the project.

Bibliography

[1] Nalpantidis Lazaros, Georgios Christou Sirakoulis, and Antonios Gasteratos. Review of stereo vision algorithms: From software to hardware. *International Journal of Optomechatronics*, 2(4):435–462, 2008.

[2] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47:7–42, 2002.

[3] Zheng Gu, Xianyu Su, Yuankun Liu, and Qican Zhang. Local stereo matching with adaptive support-weight, rank transform and disparity calibration. *Pattern Recognition Letters*, pages 1230–1235, 2008.

[4] H Hirschmüller, P R Innocent, and J Garibaldi. Real-time correlation-based stereo vision with reduced border errors. *International Journal of Computer Vision*, 47(1):229–246, 2002.

[5] Sukjune Yoon, Sung-Kee Park, Sungchul Kang, and Yoon Keun Kwak. Fast correlation-based stereo matching with the reduction of systematic errors. *Pattern Recognition Letters*, pages 2221–2231, 2005.

[6] Fredrik Ekstrand, Carl Ahlberg, Mikael Ekström, Lars Asplund, and Giacomo Spampinato. Resource limited hardware-based stereo matching for high-speed vision system. In *5th International Conference on Automation Robotics and Applications, 2011. (ICARA 2011)*, December 2011.

[7] Carl Ahlberg, Giacomo Spampinato, Jörgen Lidholm, Fredrik Ekstrand, Mikael Ekström, and Lars Asplund. Gimme - a general image multiview manipulation engine. Technical paper, School of Innovation Design and Engineering, Mälardalen University, 2011.

[8] Jörgen Lidholm, Fredrik Ekstrand, and Lars Asplund. Two camera system for robot applications; navigation. In *13th IEEE International Conference on Emerging Technologies and Factory Automation, 2008. (ETFA 2008)*, pages 345–352. IEEE, September 2008.

[9] http://vision.middlebury.edu/stereo.

[10] Miguel Arias-Estrada and Juan Xicotencatl. Multiple stereo matching using an extended architecture. In Gordon Brebner and Roger Woods, editors, *Field-Programmable Logic and Applications*, volume 2147 of *Lecture Notes in Computer Science*, pages 203–212. Springer Berlin / Heidelberg, 2001.

[11] Lee Sunghwan, Yi Jongsu, and Kim Junseong. Real-time stereo vision on a reconfigurable system. *Lecture Notes in Computer Science*, 3553:299–307, 2005.

[12] Lu Zhang, Ke Zhang, Tian Sheuan Chang, Gauthier Lafruit, Georgi Krasimirov Kuzmanov, and Diederik Verkest. Real-time high-definition stereo matching on fpga. In *Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays*, FPGA '11, pages 55–64, New York, NY, USA, 2011. ACM.

[13] Kristian Ambrosch, Wilfried Kubinger, Martin Humenberger, and Andreas Steininger. Flexible hardware-based stereo matching. *EURASIP Journal on Embedded Systems*, 2008:1–13.

[14] Kristian Ambrosch and Wilfried Kubinger. Accurate hardware-based stereo vision. *Computer Vision and Image Understanding*, pages 1303–1316, 2010.

[15] T. Kanade and M. Okutomi. A stereo matching algorithm with an adaptive window: theory and experiment. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(9):920 –932, sep 1994.

[16] Kuk jin Yoon, Student Member, and In So Kweon. Adaptive support-weight approach for correspondence search. *IEEE Trans. PAMI*, 28:650–656, 2006.

[17] Yuri Boykov, Olga Veksler, and Ramin Zabih. A variable window approach to early vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:1283–1294, 1998.

[18] Xun Sun, Xing Mei, Shaohui Jiao, Mingcai Zhou, and Haitao Wang. Stereo matching with reliable disparity propagation. *The First Joint 3DIM3DPVT Conference 3DIMPVT 2011*, 2011.

[19] Chonghun Roh, Taehyun Ha, Sungsik Kim, and Jaeseok Kim. Symmetrical dense disparity estimation: algorithms and fpgas implementation. In *Consumer Electronics, 2004 IEEE International Symposium on*, pages 452 – 456, 1-3, 2004.

[20] Pascal Fua. A parallel stereo algorithm that produces dense depth maps and preserves image features. *Machine Vision and Applications*, 6(1993):35–49, 2004.

[21] A. Fusiello, V. Roberto, and E. Trucco. Experiments with a new area-based stereo algorithm, 1997.

[22] Geoffrey Egnal and Richard P. Wildes. Detecting binocular half-occlusions: Empirical comparisons of five approaches. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24:1127–1133, August 2002.

[23] K Muhlmann, D Maier, Jurgen Hesser, and R Manner. Calculating dense disparity maps from color stereo images, an efficient implementation. *International Journal of Computer Vision*, 47(1):30–36, 2002.

[24] Xing Mei, Xun Sun, Mingcai Zhou, Shaohui Jiao, Haitao Wang, and Xiaopeng Zhang. On building an accurate stereo matching system on graphics hardware. *GPUCV'11: 1st IEEE Workshop on GPU in Computer Vision Applications*, 2011.

[25] D. Marr and T. Poggio. Cooperative computation of stereo disparity. *Science*, 194:283–287, 1976.

[26] Michael Bleyer and Sylvie Chambon. *Does Color Really Help in Dense Stereo Matching ?* Number 1. 2010.

[27] M A Vega-Rodríguez, J M Sánchez-Pérez, and J A Gómez-Pulido. *An FPGA-based implementation for median filter meeting the real-time requirements of automated visual inspection systems*. Citeseer, 2007.

[28] Heiko Hirschmüller and Daniel Scharstein. Evaluation of stereo matching costs on images with radiometric differences. *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 1582–1599, 2009.