Stereo Video Processing for Depth Map

Harlan Hile and Colin Zheng

University of Washington

Abstract

This paper describes the implementation of a stereo depth measurement algorithm in hardware on Field-Programmable Gate Arrays (FPGAs). This system generates 5-bit pixel disparities on 512 by 480 pixel images at video rate (30 frames/sec). The algorithm implemented is a window based scan-line correlation search technique. Hardware implementation speeds up the performance more than 100 times that of the same algorithm running in software. In this paper, we describe the programmable hardware platform, the base stereo vision algorithm and the design of the hardware. We try various video inputs and show sample outputs from the functioning hardware.

Introduction

High-level computer vision tasks, such as robot navigation and collision avoidance, require 3-D depth information about the surrounding environment at video rate. Current general purpose microprocessors are too slow to perform stereo vision at video rate. For example, it takes several seconds to execute a medium-sized stereo vision algorithm for a single pair of images on a 2 GHz general-purpose microprocessor. To overcome this limitation, designers in the last decade have built custom-designed hardware systems to accelerate the performance of the vision systems.

Hardware implementation allows one to exploit the parallelism that usually exists in image processing and vision algorithms, and to build systems to perform specific calculations very quickly compared to software. By processing several parts of the data in parallel, we can speed up the overall functioning and achieve video-rate performance.

Custom-designed hardware, however, has two major disadvantages:

1) The design cycle is slow: The time required for fabrication and test of a typical Application Specific Integrated Circuit (ASIC) is in the order of a few months;

2) The fabrication process of the chip and also circuit board is expensive, on the order of hundreds of thousands of dollars.

Over the past decade a third option between software and custom hardware has become viable: using re-programmable chips called Field-Programmable Gate Arrays (FPGAs). These devices consist of programmable logic gates and routing that can be re-configured to implement essentially any hardware function [2]. This method combines the advantage of custom-designed hardware with the advantages of software implementation, which are reprogrammability and rapid design cycle. The reprogrammability feature also allows the designers to use the same hardware system for other vision or non-vision tasks, which again can amortize the cost of the system.

This paper describes the development of a quite complex stereo vision algorithm on a reconfigurable platform. Section 2 describes related work and section 3 explains the window based correlation search algorithm implemented in this work for stereo matching.

Section 4 illustrates the implementation process of the algorithm on hardware. Finally, section 5 presents the overall performance of the hardware system with some results. Conclusion and future work are discussed in the last section.

Related Work

To compare with other stereo systems, a number of fast algorithms that do not use reconfigurable hardware also exist in the literature. Some of these take advantage of special hardware, specifically SIMD (single instruction, multiple data) instructions in Intel MMX processors [4][6]. A number of intensity-based cross-correlation techniques are reported [7][8][9]. Approaches to speed up algorithms include the use of image pyramids [8][5] or simplified algorithms [1].

The two fastest algorithms both use MMX processors. Hirschmuller et al. [4] achieve 4.7 frames/second on 320 x 240 images using intensity correlation in 7x7 windows with Gaussian pre-filtering. Their method includes both rectification and left-right consistency checking (without these the frame rate is estimated at 5.7 frames/second). Their hardware is a 450MHz Pentium running Linux. Muhlmann [6], using an 800MHz MMX processor, achieves less than 5 frames/second on 348 x 288 color images, again using intensity correlation. A trinocular disparity system is implemented by Mulligan et al. [7] on a 4-processor system utilizing Intel IPL libraries. This system takes 456 ms to compute disparity for three 320 x 240 images up to a maximum disparity of 64 pixels. Sun [8] computes disparity (up to 10 pixels) in 450ms on a 500MHz Pentium processor based on fast intensity correlation in an image pyramid. Birchfield and Tomasi [1] use a simplified algorithm that minimizes a 1-D cost function based on pixel intensity differences. They report speeds of 4 seconds/frame on a 333 MHz Pentium for 640 x 480 images. Given a faster processor, their algorithm would doubtlessly perform better than one frame/second. It has the advantage of explicitly accounting for occlusion as well as propagating disparity information between scan lines.

Besides software and specialized hardware solutions, a variety of reconfigurable stereo machines have been introduced in recent years. The PARTS reconfigurable computer [10] consists of a 4 x 4 array of mesh connected FPGAs with a maximum total number of about 35,000 4-input LUTs. A stereo system was developed on PARTS based on the census transform, which mainly consists of bitwise comparisons and additions [11]. In [3], a 4 x 4 matrix of small FPGAs is used to perform the cross-correlation of two 256 x 256 images in 140 ms. In [4], a combination of FPGA and Digital Signal Processors (DSPs) is used to perform edge-based stereo vision. They use FPGAs to perform low level tasks like edge detection and DSPs for high level image processing tasks.

Stereo Matching Algorithm

The heart of any stereo vision system is stereo matching, the goal of which is to establish correspondence between two points arising from the same element in the scene. Stereo matching is usually complicated by several factors such as lack of texture, occlusion, discontinuity and noise. Researchers have proposed techniques to solve and improve the performance of stereo matching that can be categorized into three major groups: 1) Intensity-based; 2) Feature-based; and 3) Phase-based. Our algorithm goes into group of

intensity-based techniques which assume that image intensity corresponding to a 3-D point remains the same in binocular images. We used window based search as respect to pixel based method, hoping to get less noisy results.
We assume the image pair is rectified such that the epipolar lines are horizontal, so searching along the epipolar line is equivalent to doing scan-line searching. The algorithm is as follow:

```
For each frame
    Pre-filter image
    For each scan line
        For each pixel in the right image
            For each window on the same scan line in the left image
                    Compare the window around the pixel on the left to the right
            Pick the window with the minimum difference
            Associate the difference with disparity and depth
Output the depth map
```

There are several reasons for choosing this algorithm. First of all, it's a simple algorithm which makes it a good candidate for hardware implementation. Secondly, it's primarily composed of linear operations which are easier to implement. And finally, the algorithm doesn't have a complicated control, which makes the real-time flow of data possible through the hardware system. We will describe design of the hardware system in more details in the next section.

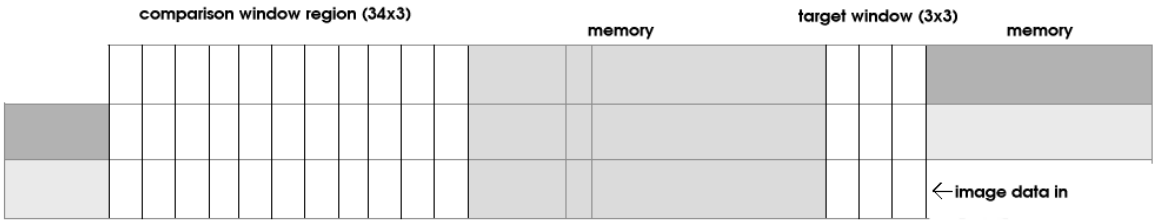

Figure 1. Memory Structure

Hardware Implementation

The main challenge of implementing a complex algorithm on reprogrammable hardware is that there is a fixed amount of hardware available. Achieving the best overall performance requires efficient usage of all hardware resources.
Our design is broken up into several components. The first stage is to down-sample the input data. This is done by storing one line, and on the next line taking the average of the values in memory and incoming values to output a single average value per four original values. We decided to down-sample our data for several reasons. First, we felt that due to hardware constraints, window compares larger than 3 by 3 would be infeasible. Small windows will likely give inaccurate results, and by down-sampling first we increase the amount of data in that small window size. Moreover, down-sampling serves as a pre-filtering step, which smooth the data for better window correspondence. Finally, down-sampling also allows the next stage more time between incoming values.
The next stage of our design is to do window comparison. In order to have data available for it, we have a memory structure like that in Figure 1. The right side contains a 3 by 3 register group which will be the target window. The right side contains 34 by 3 register

group, and will be used to do 32 window comparisons against the target group. Cyclically accessed memory blocks fill in the spaces between the register groups. When new data arrives, it is shifted through the structure. Our window comparison function is just a sum of absolute differences (L1 norm) of the pixels Y value. Once all the windows have a match metric, the best match is chosen and the location of this match is the disparity between the two images, which is related to depth.

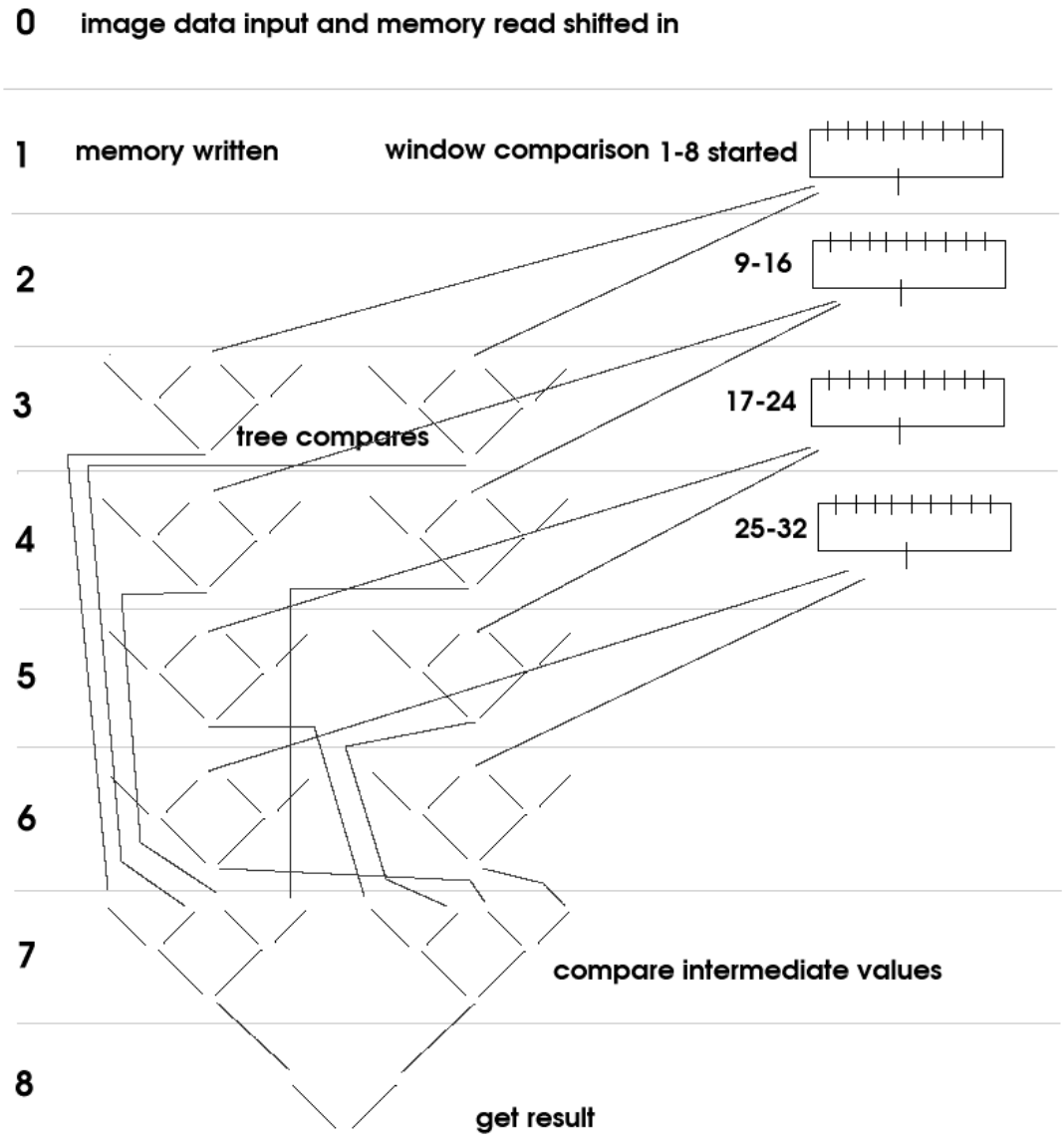

Figure 2. Pipelined Stages

Fitting this design into the FPGA hardware required significant pipelining (see Figure 2). On cycle 0, when data arrives, it is shifted into the memory structure, so computation can begin the next cycle. On cycle 1, data that shifted out of the register groups is saved back into memory where necessary. We found that 8 copies of the window comparison function fit reasonably well in hardware, so cycle 1 also starts the first 8 comparisons.

Our function had to be pipelined, so the result does not arrive until several cycles later, but the next cycle begins the next 8 window comparisons. When the window results

arrive, the values are compared in a tree structure. We fit two comparisons in a clock, condensing 4 values down to 1. Carried along with the window comparison metric is the relative location of the best match. When we get to cycle 7, comparisons of previous groups of 4 are done, to get the best match for the first 16 and the second 16 windows. In the last cycle, these final two are compared, the best is chosen and a relative disparity between that window and the target is output as a depth map.

We found some strange behavior in Synplicity that we had to work our way around. Originally, we had our disparity values (associated with each window comparison result) filled in completely (a 5 bit number). This means at any one stage, only the lower two bits would change. Synplicity would notice this, and optimize the top bits away. While this seems valid, when Synplicity removed the bits it would replace them with zero's, and in many cases the top bits had an important meaning (even though they were constant). We got around this problem by using as few bits as possible at each stage. That is, our first tree compare only gives out a 2 bit number. When we come to later compares (in cycle 7), another bit is added on top for each compare. This allows us to get the final correct 5 bit number, without Synplicity optimizing them away. Also note that assigning the smallest number of disparity bits at each compare would allow us to use the same hardware for many of the compares, so the compares on cycles 3, 4, 5, and 6 could use the same hardware, and just save the result to different registers.

Experiment Setup & Results

The FPGA stereo system is able to produce a dense disparity map of size 240 x 480 pixels with 5-bit subpixel accuracy at the rate of 30 frames/sec.

Figure 3 shows the setup of a stereo system. Given any 3D point in the scene, the two corresponding 2D points in the image plane are constrained by the epiploar line (scan line here).

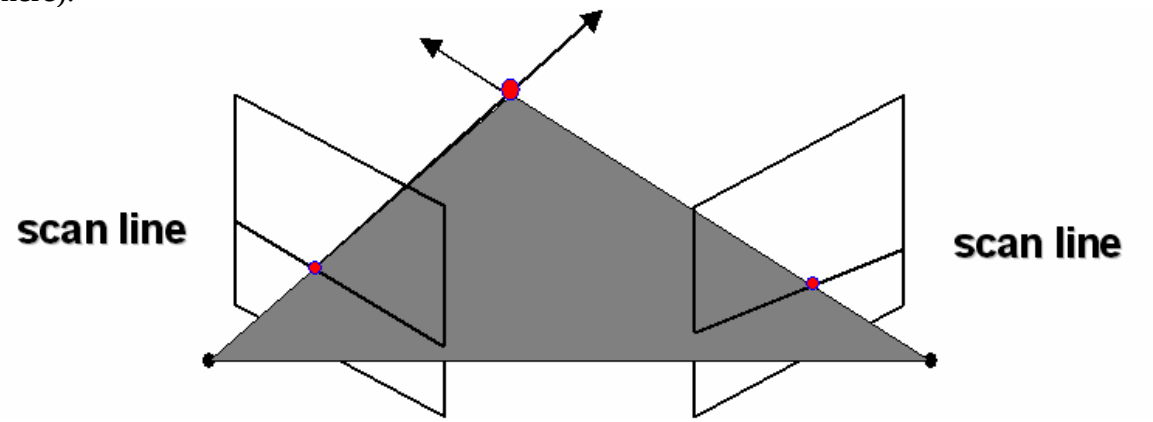

Figure 3. Stereo System Setup

As for our setup, instead of having two cameras, we use only one camera with a stereo adaptor, which produces two views in one image. Figure 4a is an example of it. It has two views separated by a beam. Our system takes in images of size 512 x 480, and we assume the beam between two views is 32 pixels wide. So each down-sampled view would have resolution 120 x 240. And that's the size of our depth map. Each pixel has disparity of 5 bits. We implemented this window based scan-line searching algorithm in C++, and it runs about 3 seconds per frame on a 2 GHz Pentium4 machine. Our hardware

implementation gives the exact same results as the software does, yet it's running at 30 frames per second. Figure 5 and Figure 6 are results of different scenes. They convey the right relative depth information, as the lighter the color, the closer the object is to the camera.

Figure 4a. Stereo Image Pair

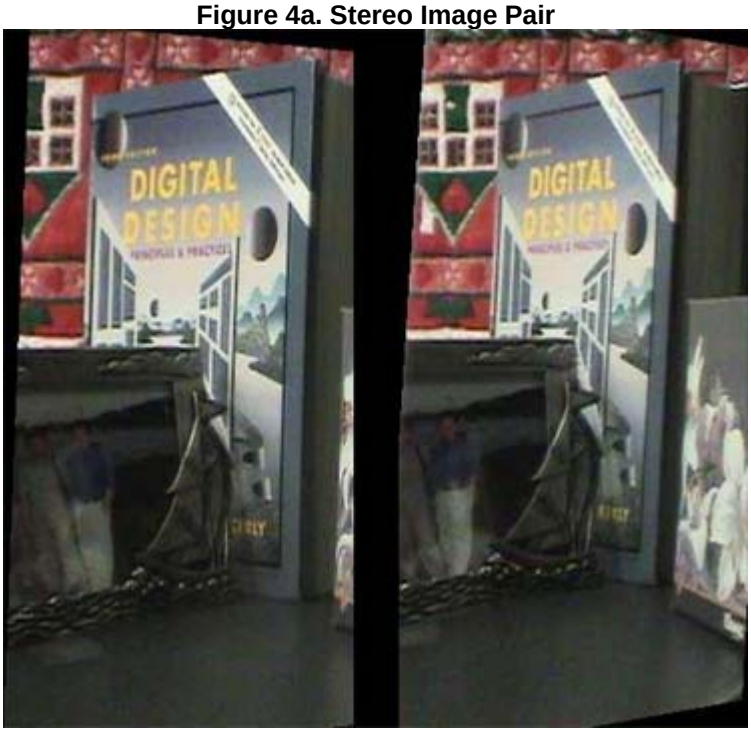

Figure 4b. Rectified Image Pair

Figure 5 Left View, Right View, Depth Map

Figure 6 Left View, Right View, Depth Map

Since images produced by the camera equipped with the stereo adaptor are not well aligned, we have to apply rectification as a pre-step for the input video. A projection rectification algorithm is implemented according to [12]. Based on a few point correspondences, we generated the homographies to transform the images such that the stereo pair is now well aligned. Figure 4b shows how it transforms from Figure 4a to the rectified version. These homographies can be applied to each frame thorough the whole video to rectify them correctly. And this is definitely achievable even on the board. So we

can do image rectification on FPGA as well if we can specify the homography matrix in advance. This would be our future work.

Conclusion & Future Work

By doing this project, we have shown the feasibility of implementing challenging vision applications on FPGAs to achieve real-time performance. Due to the time limitation, we have left quite large space for improvement as future work. This includes bounds detection, image rectification etc.

Reference:

[1] S. Birchfield and C. Tomasi, Depth discontinuities by pixel-topixel stereo. *IJCV*, 35(3):269-293, 1999.

[2] S. Brown, R. Francis, J. Rose, Z. Vranesic, *Field-Programmable Gate Arrays*, Kluwer Academic Publishers, May 1992.

[3] O. Faugeras, et al., Real time correlation based stereo: algorithm, implementations and applications, Research Report 2013, INRIA Sophia-Antipolis, 1993.

[4] H. Hirschmuller, et al., Real-time correlation-based stereo vision with reduced border errors. *IJCV*, 47(1/2/3):229-246, 2002.

[5] G. Van Meerbergen, M. Vergauwen, M. Pollefeys, and L. Van Gool. A hierarchical symmetric stereo algorithm using dynamic programming, *IJCV*, 47(1/2/3):275-285, 2002.

[6] K. Muhlmann, D. Maier, Jurgen Hesser, and R. M. Anner, Calculating dense disparity maps from color stereo images, an efficient implementation. *IJCV*, 47(1/2/3):79-88, 2002.

[7] J. Mulligan, et al, Trinocular stereo: A real-time algorithm and its evaluation, *IJCV*, 47(1/2/3):51-61, 2002.

[8] C. Sun. Fast stereo matching using rectangular subregioning and 3d maximum-surface techniques, *IJCV*, 47(1/2/3):99-117, 2002.

[9] O. Veksler. Stereo correspondence with compact windows via minimum ratio cycle, *IEEE Trans. on PAMI*, 24(12):1654-1660, Dec 2002.

[10] J. Woodfill and Von Herzen, Real-time stereo vision on the PARTS reconfigurable computer, *The 5th Symposium on FCCM Proceedings*, Pages:201-210, 1997.

[11] R. Zabih and J. Woodfill, Non-parametric Local Transforms for Computing Visual Correspondence, *Proceedings of 3rd European Conf. on Computer Vision*, pp. 150-158, May 1994.

[12] R. Hartley, Theory and Practice of Projective Rectification, *International Journal of Computer Vision*, 1999