FPGA Implementation of Vision Algorithms for Small Autonomous Robots

J. D. Anderson, D. J. Lee, J. K. Archibald
Brigham Young University, Provo, UT, USA 84602

ABSTRACT

The use of on-board vision with small autonomous robots has been made possible by the advances in the field of Field Programmable Gate Array (FPGA) technology. By connecting a CMOS camera to an FPGA board, on-board vision has been used to reduce the computation time inherent in vision algorithms. The FPGA board allows the user to create custom hardware in a faster, safer, and more easily verifiable manner that decreases the computation time and allows the vision to be done in real-time. Real-time vision tasks for small autonomous robots include object tracking, obstacle detection and avoidance, and path planning. Competitions were created to demonstrate that our algorithms work with our small autonomous vehicles in dealing with these problems. These competitions include Mouse-Trapped-in-a-Box, where the robot has to detect the edges of a box that it is trapped in and move towards them without touching them; Obstacle Avoidance, where an obstacle is placed at any arbitrary point in front of the robot and the robot has to navigate itself around the obstacle; Canyon Following, where the robot has to move to the center of a canyon and follow the canyon walls trying to stay in the center; the Grand Challenge, where the robot had to navigate a hallway and return to its original position in a given amount of time; and Stereo Vision, where a separate robot had to catch tennis balls launched from an air powered cannon. Teams competed on each of these competitions that were designed for a graduate-level robotic vision class, and each team had to develop their own algorithm and hardware components. This paper discusses one team's approach to each of these problems.

Keywords: FPGA, Robotic Vision, Hardware Acceleration

1. INTRODUCTION

Robotics is a growing field that has a large impact on our world. Robots range in type from automated car factories to autonomous ground vehicles. One thing that is impacting how common robots are becoming is how well these robots are able to see and interact with the world. At Brigham Young University, we are exploring the use of on-board vision algorithms on small autonomous robots. This is made possible with the growing ability of Field Programmable Gate Arrays (FPGAs) to contain larger and faster hardware modules. By using FPGAs, we are able to develop vision algorithms that can handle real-time problems especially for small robots. Tracking an object with a robot can become a problem as the object moves faster or grows more distant. FPGAs allow for hardware acceleration[1], allowing a faster frame rate and the ability to move faster to keep up with the object. FPGAs also help in obstacle detection and avoidance, as the increase in hardware speeds allows for faster recognition of the obstacle. FPGAs also help in development, as changes can be made to the hardware, resynthesized, and then redownloaded to the robot. In developing the many algorithms to handle these and other problems, a series of competitions were created that would test the abilities of our algorithms. Five competitions were created to test the hardware/software algorithms that were created. Teams composed of graduate students from Brigham Young University participated in each competition. This paper will explore one of these team's approaches to each of these problems. We will first discuss the robot that we used for our experiments. We will then discuss the competitions that took place at Brigham Young University. We will discuss our approach to each competition next, followed by the hardware acceleration units we designed to aid us in the competition.

2. RELATED WORKS

Much of the research in real-time vision systems is being done with FPGA technology[2]. There has been research in converting current algorithms from sequential programs to FPGA implementations[3,4]. Much of this type of work is geared towards providing platforms and tools to aid in the design of FPGA implementations. Shawky, et al[5], and Hartmann, et al[6], both provide frameworks that, when used by hardware accelerators, provide a database of the parts of the image that the user is interested in keeping track of. Drayer, et al[7], and Benedetti, et al[8], both provide user interfaces

Intelligent Robots and Computer Vision XXIII: Algorithms, Techniques, and Active Vision,
edited by David P. Casasent, Ernest L. Hall, Juha Röning, Proc. of SPIE Vol. 6006
600612, (2005) · 0277-786X/05/$15 · doi: 10.1117/12.632029

which allow the user to build their hardware from the ground up. Such tools allow for design entry, verification, and translation to Hardware Description Language (HDL) code ready to be synthesized for an FPGA. Shelley and Seed[9] provide a similar framework, but don't utilize the advantages of an FPGA. van Inge and Hertzberger[10] provide a similar system developed on an ASIC which provides greater flexibility than most other ASICs.

Torres-Huitzil and Arias-Astrada[11] present an edge detector implementation, allowing for fast and reliable edge detection through an FPGA. Cucchiara, et al[12], and Aranda, et al[13], both present different video-time target-tracking implementations. Seonal, et al[14], and Marino, et al[15], present feature matching implementations. Seonal takes advantage of the ability of the FPGA to facilitate communication, while Marino uses the parallelism abilities of the FPGA to perform multiple searches simultaneously. Darabiha, et al[16], includes a stereo depth measurement implementation, allowing a system with multiple cameras to find matches at video rate. Sudha[17] uses an FPGA to implement an Euclidean Distance Transformation, taking advantage of the FPGAs ability to perform operations in parallel. Chiu, et al[18], uses an FPGA to perform the preprocessing and image retrieval required for an Unmanned Air Vehicle (UAV) to land safely.

Work has also been done in generating compilers specifically geared toward changing existing C code into functional HDL code[19]. Torres-Huitzil, et al[20], provides a system comprised of an FPGA specifically for creating and implementing real-time computer vision algorithms. Tyrell, et al[21], describes related work in a system which allows for real-time vision computer vision algorithms, but is different in that there system does not use an FPGA.

3. THE ROBOT

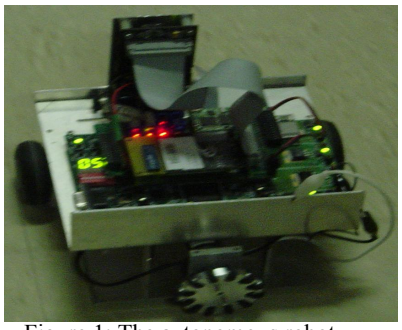

Figure 1: The autonomous robot.

During much of the course of the competition, we used the robot shown in figure 1. This robot consists of a Memec Virtex-II V2MB 1000 Development Board which contains a Virtex-II FPGA board along with different I/O ports and different memory modules, including the DDR SDRAM which is integral to the use of our vision algorithms. The Memec board is connected to a board which was custom made at BYU and contains the physical interface to both the camera and motors. Because we didn't have a processor coupled with our FPGA chip, we needed to synthesize one and place it in the FPGA. We chose the Microblaze softcore processor as our processor of choice and downloaded the µC/OS-II RTOS to allow the robot to run a real time operating system in which to perform its necessary tasks. The Microblaze processor is a Reduced Instruction Set Computer (RISC) based microprocessor which enables the FPGA to function as a general purpose processor in addition to its hardware acceleration role.

The camera is mounted to the front of the robot. We used the MT9V111 1/4 inch SOC VGA CMOS Active Pixel Digital Image Sensor provided to us by Micron technology. This camera streams the image to the FPGA in chrominance and luminance values, which are used by the FPGA in its vision algorithms. Three wheels are used to move the robot around, although only two are driven by motors. The remaining wheel is a hold over from the robots previous iteration as an omni-directional robot used in robot soccer exercises. Its primary purpose is to allow the robot to move consistently regardless of the surface upon which the robot is moving. The other two motors are driven by TI SN754410 Half-H Drivers, which convert the signals from the robot to the pulses needed to turn the Faulhaber 2224 012 SR motors. Each motor also consist of a Faulhaber IE2-512 encoder and a Faulhaber Planetary Gearhead to assist in motor function and feedback control. A digital compass was also used as a means of providing feedback to the robot.

To assist the robot in performing the various competitions, a set of tasks was created which run on the µC/OS-II RTOS. The first of these tasks, the Motor Control task, simply moves the robot the specified distance or rotates it the specified angle. This is the primary task of the robot, but it rests often, allowing the robot to perform the task relating to its control. The secondary task of the robot, the Control task, is to specify the directions the robot is to take, according to the specific competition that the robot is participating in. As the robot moves and acts, the camera data is passed through the FPGA to the Control task, which uses this information as it plans its next action, such as specifying the next position to turn to. It passes this information to the Motor Control task, which then moves the robot as specified. Between these two tasks, all of the operation needed for the competitions takes place and the robot is able to perform as specified.

Communication between the robot and the user is accomplished by using wireless communication and the XCite RF wireless transciever developed by Maxstream. The user uses a simple terminal program developed at Brigham Young University primarily for this type of communication. The user can then select the various options necessary to start the robot on its path and to gather useful information such as path planning and distance from obstacles which are a necessary part in debugging the modules on the FPGA. It should be noted that although tele-robotic communication and control can be used through the wireless transceiver and the terminal, in each competition, the only use of the terminal was to start the robot and to stop it, as necessary.

4. THE COMPETITIONS

There were five competitions developed at Brigham Young University designed to test the robustness and ability of the vision algorithms of the teams. Each of these different competitions was designed to test a different aspect of robot vision. In this section, we will explore these different competitions and explain how each of them is used to test the robotic vision of our autonomous robot. The competition was the same for each team, and each team competed in sequential order on the same day. Each competition took place at the Robotic Vision lab at Brigham Young University. A practice competition took place for each competition starting four weeks before the first competition, so that each team could tell how other teams were doing and their relative position and accomplishments.

4.1 Mouse-in-a-Box

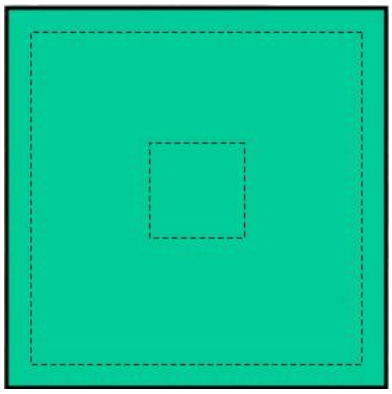

Figure 2: Mouse-in-a-box.

The Mouse-in-a-Box competition was the first competition which took place on April 4, 2005. The competition was used primarily to test motion analysis and obstacle detection. A four foot by four foot box was constructed to contain the area in which the competition was to take place. The walls of this box were eight inches high, allowing them to take up most of the image as the robot approached them. A pattern of black and white diamonds was created and mounted on the walls to provide texture for the robot to detect. Otherwise, it becomes difficult to detect the walls, and a collision becomes eminent. Inside of this box, two other boxes were created out of masking tape to serve as visual markings for the observers to watch the robot progress through the competition. One of these boxes was a three and a half foot by three and a half foot box serving as the boundary line for the robot and the other was a one foot by one foot box serving as the starting box for the robot. This set up is shown in figure 2.

The object of this competition was to start the robot in the starting box facing any arbitrary wall. Once started, the robot approaches each wall until it determines that it has come within six inches of a wall. At this point, it returns to center and turns to face another wall. Each time the robot approached a wall, stopped before impacting, and then turned around and returned to center, the robot scored a point. If the robot failed to break the six inch boundary, no point was scored, and if the robot managed to impact the wall, a point was deducted from the robot's score. The team which achieved twelve points the quickest won the competition and scored more points in the overall competition.

4.2 See and Avoid Competition

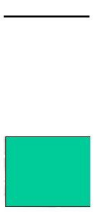

Figure 3: See and avoid.

The See and Avoid competition was the second competition and took place on April 6, 2005. This competition was used to test obstacle detection and avoidance. A one foot by one foot by one foot box was created to serve as an obstacle. A pattern of black and white squares lies on the exterior of the box serving as texture for the robots to detect. A starting line and an ending line lie five feet apart. Each line has a one foot marked section serving as the starting and ending portion, respectively. The box can be placed at any point between the start line and the end line such that it leaves at least nine inches between itself and both of the lines. The box stays in the corridor marked by the one foot portions of each line, forcing the robot to have to avoid the box. This competition is shown in figure 3.

The robot is placed at the starting line. Once the robot is told to proceed, it moves forward

until it detects the box. Upon detection of the box, it needs to avoid it and it is allowed to move in any way such that it avoids the box and returns to the corridor on the far side of the box before continuing on. Due to the mounting of the camera, once we have started to move around the obstacle, we can no longer see it, and so it was decided that encoder feedback could be used to avoid the obstacle once it was detected. Each team was allowed a total of three runs to attempt to move around the obstacle and achieve a faster time. Each team ran against the same placement of the box for a given run, allowing them to see how well they faired against each other. The team which achieved the fastest time in moving around the obstacle placed first on this competition and gained the most points from this competition.

4.3 Urban Canyon Following Competition

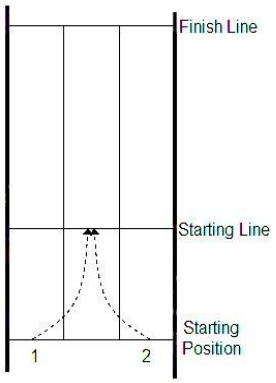

Figure 4: Urban canyon following.

The Urban Canyon Following task was the third competition and took place on April 8, 2005. This competition was designed to test optical flow and path planning using vision. Two six foot long straight walls were created that served as the canyon walls. On the interior of these walls, the same pattern of black and white diamonds used in the Mouse-in-a-Box competition was used to serve as the texture on which vision could take place. The walls each stood eight inches high. Because this task took place in the hallway outside of the Robotic Vision lab at Brigham Young University, we were able to use the one foot by one foot tiles to serve as the visual boundaries without having to lay down any additional markings. The starting position is used as the place to start the robot from, whereas the starting line is defined as the point at which the robot must have found its way to the center of the canyon. The robot then needed to stay roughly centered as it made its way to the finish line. This is shown in figure 4.

The robot was placed in either position one or position two. It did not know inherently which starting position it was placed in, and so the robot required some form of vision to detect where it was. The timer was started as soon as the robot began moving. The time of the robot was only scored if the robot was in the center at both the starting and finish line. The team which finished the canyon in the fastest time placed first and earned points towards the final score. Each team was allowed three runs, one from each side and one from the side chosen by the judge.

4.4 Grand Challenge Competition

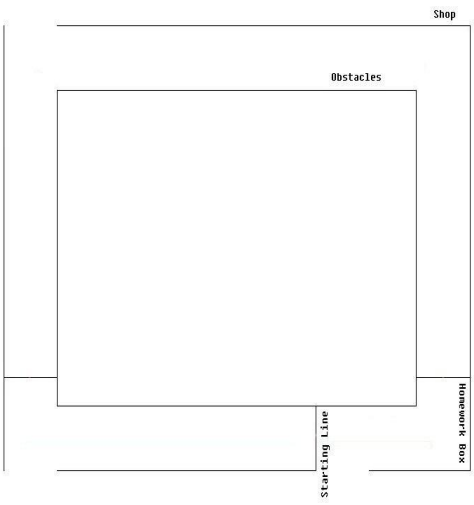

Figure 5: The grand challenge.

The Grand Challenge competition was the fourth competition and took place on April 11, 2005. This competition was designed to test optical flow, motion analysis, obstacle detection and avoidance, and path planning, thus testing everything tested in each of the earlier competitions. This competition took place near the Robotic Vision lab at the Electrical and Computer Engineering shop at Brigham Young University. The hallway shown in figure 5 was chosen because of its rectangular nature as well as the obstacles already in place due to the nearness of the shop. The robot started at the starting line and progressed through the hallways in an attempt to return back to the starting line in the shortest amount of time possible while still avoiding the obstacles that lay in the hallways.

The robot was placed at the starting line and was given the command to start. The robot then progressed through the hallways using whichever method the teams felt was appropriate. Each team was allowed three runs in the Grand Challenge and kept their best score. The team which returned to the starting line the quickest placed first and scored more points towards their final score.

4.5 Stereo Vision Competition

The Stereo Vision competition was the fifth and final competition and took place on April 13, 2005. This competition was designed to test stereo vision and target tracking. An outer frame was developed and built with motor controlled

axis allowing a smaller "mouth" to move under the direction of a desktop program. Simultaneously, an air cannon was built which would allow the user to fire tennis balls from a distance of about twenty feet away from the frame time to allow the mouth to move to catch the tennis ball. Two cameras were mounted to the frame at a distance of three feet to allow for the stereo vision portion of the project to take place. A group was formed from a member of each team to set the intrinsic properties of the camera to allow each team to develop their algorithms for catching the ball. Each team then had access to the desktop for a limited amount of time to develop the algorithms that would move the X-Y Stage to a position to catch the tennis ball.

Figure 6: Stereo vision.

Each team had a total of three shots from the air cannon. A member of the team would stand at the desktop to prep it and begin the program, following which another member of the team would fire the tennis balls at the frame. The team which caught the most balls placed first in this competition and scored more points for their team for the final scoring. In the case of a tie, each team was allowed three fair shots made by the opposing team. The judge of the competition would decide what a fair shot was.

4.6 Scoring of the Competitions

Each competition was scored independently and the teams final score after all challenges was the sum of the scores they received for each competition. The winner of each competition scored 20 points for their team, second place scored 15 points, third scored 10 points, and fourth 5 points. The winning team won a $40 gift certificate per team member.

5. OUR TEAM'S APPROACH

In this section, we will explore the approach of team three in greater detail. We will first discuss their general approach to each of the competitions, and then we will discuss the hardware acceleration module developed for the robot. Team three managed to do fairly well during each of the practice competitions, managing to place first in Mouse-in-a-Box, See-and-Avoid, and Urban Canyon Following with a good showing at both the Grand Challenge and Stereo Vision. However, having set the tone for the competition, the team was unable to keep up with the advances made by other teams and did not place as highly during the final competition.

5.1 Mouse-in-a-Box Strategy

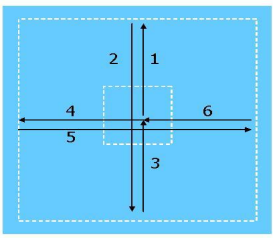

Figure 7: Mouse-in-a-

Our approach was to use a simple Time to Impact (TTI) approach that would allow us to determine how far away the wall was. Once we were a specified number of frames away, we would stop, and then turn around to return home. After testing and debugging the software and hardware, it was determined that twenty frames away from the wall equated to roughly six inches, thus achieving the goal for the bounce.

Once at home in the center, we would face the next wall in our path and approach it. Figure 7 includes a diagram of our approach to this task. This approach was followed through all six steps until the twelve bounces had been achieved.

6.2 See and Avoid Strategy

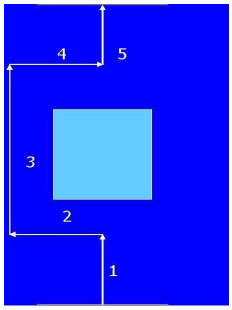

Figure 8: See and avoid strategy

We used the same TTI algorithm as used in the Mouse-in-a-Box task to determine when we were the correct distance away from the obstacle. We then turned to the left to allow us to move around the obstacle and followed an encoder value to finish moving around the obstacle before returning back to the center and to the finish line. Figure 8 includes a diagram of our approach to this task.

In the course of the competitions, it was also suggested that other patterns and textures be placed on the obstacle to test the robustness of the algorithms used. One of these other textures was aluminum foil. Surprisingly, the robot reacted well to the foil, detecting it slightly later than it detected the square pattern previously found on the obstacle.

6.3 Urban Canyon Following Strategy

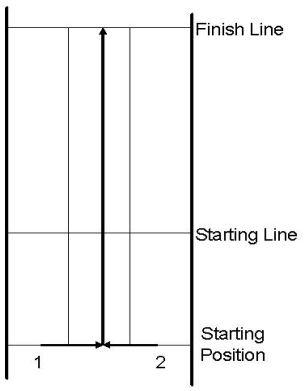

Figure 9: Urban canyon following strategy

Our approach was very specific for this challenge – we looked at the lights on the ceiling. By noting that the lights on the ceiling would always be in a specific place in our representation of the image viewed by the robot, we were able to determine where the robot should be if it were in the center of the canyon. The first step in our approach to this task included calibrating the robots image, specifying the location of the light from both the right and left sides of the canyon.

We could then calculate how far off we were from the center and turn to the center so that we could move the specified amount to place us in the center. Once centered, we had a clear shot to the end of the canyon, allowing us to move forwards. However, due to inaccuracies in our feed back, we occasionally overshot our mark and ended outside the final area. Figure 9 includes a diagram of our approach to this task.

6.4 Grand Challenge Strategy

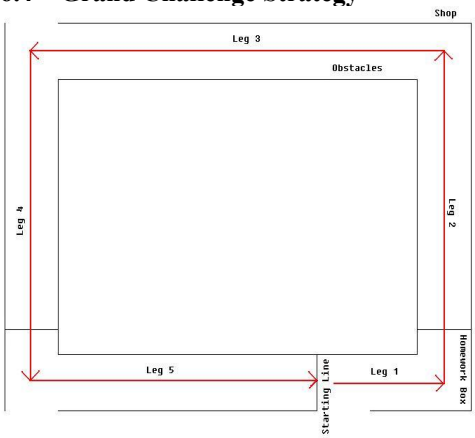

Figure 10: Grand challenge strategy

Our approach followed a similar vein to the Canyon Following task. By using the lights in the ceiling, we were able to stay centered in the hallway. Unfortunately, we could only look at the lights that lay twenty two feet in front of our robot, which was not always possible, as only one set of hallways went beyond the turning point.

In order to get around this obstacle, we used the encoders to tell us when we had moved the specified distance. We also used the timer provided in the Microblaze to note when we had traveled down the hallways far enough when we couldn't see any lights at all. Doing this allowed us to actually navigate the course almost every time. Sometimes, though, the code which centered our robot would miss a step and cause our robot to approach the walls at too steep of an angle. We were then unable to finish the course as our robot ended up perpendicular to the wall. Figure 10 includes a diagram of our approach to this task.

Leg 1 consisted of moving until we couldn't see a light any more and then moving an encoded distance. Leg 2 consisted of moving for a certain time period before turning to Leg 3. Leg 3 consisted of being slightly off center to allow the robot to avoid the obstacles in the hallway. Once we could no longer see a light, we would begin moving an encoded distance towards Leg 4. Leg 4 consisted of moving a specified time amount before turning to Leg 5. Leg 5 consisted of moving until we couldn't see any more lights, thus finishing the course.

One side effect of using the lights as our measure was that if we didn't see the light, then we wouldn't move. This allowed us to avoid any people that walked in the way as our robot would come to a dead stop when it could no longer see the light. While not quite obstacle avoidance, it definitely worked as obstacle detection. The unfortunate side effect, however, is that our approach to this task is constrained in that it will only work in an environment similar to the one we started with.

6.5 Stereo Vision

This competition involved an air cannon launching a tennis ball towards an XY stage that would move its mouth to the location that our software predicted the ball would go. This was the most difficult task in terms of actually accomplishing anything, as the resources were constrained and there was limited time to accomplish this task. Scoring was done by the number of balls caught verses the number of balls launched.

Our approach was fairly straight forward. After calibrating the two cameras using Tsai's camera calibration code, found at http://www-2.cs.cmu.edu/~rgw/TsaiCode.html, we would match the tennis ball's trajectory to a parabolic curve, and then use this curve to predict the location of the tennis ball when it reached our location.

We used the following determinates to determine the midpoint of the ball:

$$t1 = \frac{Det\{(o2 - o1), d2, d1 * d2\}}{\|d1 * d2\|^2} \quad (1),$$

$$t2 = \frac{Det\{(o2 - o1), d1, d1 * d2\}}{\|d1 * d2\|^2} \quad (2),$$

$$Mid = \frac{(o1 + d1 * t1) + (o2 + d2 * t2)}{2} \quad (3).$$

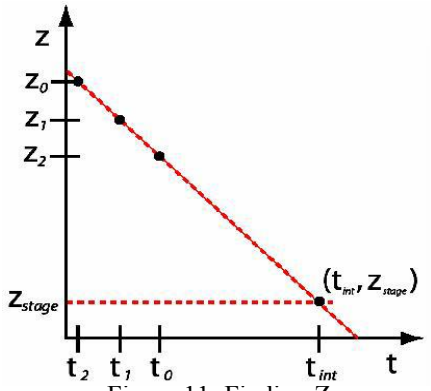

Figure 11: Finding Z

This midpoint represents the actual 3D location of the tennis ball. By then solving for Z = 0, we are able to predict where the X and Y positions of the tennis ball would be when the tennis ball crossed the XY stage. We then moved the XY stage to this location and waited for the ball to cross. Figure 11 includes how we calculated our Z location.

7. HARDWARE ALGORITHM AND IMPLEMENTATION

In this section, we will discuss the Feature Density and Distribution algorithm and its hardware implementation. The FDD algorithm was originally developed under the direction of D J Lee[22] and the Multiple AGent Intelligent Coordination and Control (MAGICC) lab at Brigham Young University. It continues to be used in the Robotic Vision lab.

7.1 Algorithm

Because of the computation demands of stereo vision and motion analysis to discover the depth away from an object, it becomes important to find ways to minimize the computation or to find other solutions to the problem. Because of the inherent limitations of the autonomous robots being used for this competition, the FDD algorithm was used to calculate distances from an object. This allowed us to avoid using complicated computations to calculate the focus of expansion, as this focus does not play a direct role in the calculation.

7.1.1 Rate of Feature Distribution Expansion

The first step in the algorithm is to create an intensity histogram of the columns of the image. This allows us to use columns as features, providing a reference point to look for similar features. The original algorithm included the calculation of the rate of expansion, *a*, and the shift, *d*, through the use of minimizing the mean squared error over the x-values 0 through *N*, where *N* is the number of image columns.

This approach, while simpler than calculations involving the focus of expansion, is still complex. Through careful experimentation, we discovered that if we performed a simple linear search between a small range of *a* and *d* values, we were able to reasonably calculate the time to impact. We choose the *a* and *d* which minimize the Sum of Absolute Differences (SAD) between the current frame and the previous frame. In other words,

$$\arg\min_a \left(\sum_i | c[a \times i + d] - p[i] | \right), \quad (4),$$

where *c* is the current histogram, *p* is the previous histogram, *i* is the element of the histogram we are searching for the match, and *p*[*i*] is the method of accessing the *i*th element of the histogram. The range for *a* is from 1 to 1.0625, while for *d* it is from -5 to 5.

7.1.2 Time to Impact

Once we have calculated the rate of expansion of the object, we can calculate the time that our robot will impact with the object if we continue at the same velocity. Note that for these calculations, it makes no difference whether the camera approaches the obstacle or the obstacle approaches the camera. For a calibrated camera, we also know the focal length, *f*, and the other intrinsic properties of that camera. The object's size, *X*' and *X*, and the distance to the object, *z*' and *z*, are unknown. Because we know the rate of expansion of the image from (4), we know that *x*' and *x* are related as *x*' = *ax*. From geometry, we know that

$$\frac{x}{f}=\frac{X}{z},\ \frac{x'}{f}=\frac{X'}{z'}=\frac{X}{z'},\ \text{and}\ z'=X\frac{f}{x'}=\frac{zx}{f}\cdot\frac{f}{ax}=\frac{z}{a} \tag{5}.$$

The time to impact, τ, is the distance, *z*, divided by the velocity, *v*:

$$\tau=\frac{z}{v}=\frac{z}{z-z'}=\frac{z}{z-z/a}=\frac{a}{a-1} \tag{6}.$$

From (4), we find our rate of expansion. Once we've found our rate of expansion, we can use (6) to find the time to impact, which indicates the number of frames needed before we impact with the obstacle we have detected. This is the same technique used in previous work[22]. As an example, consider the images in Figures 12 and 13 with their corresponding histograms. After completion, the FDD algorithm yields $a = 1.0352$, $d = -4$, and $\tau = 29$ frames.

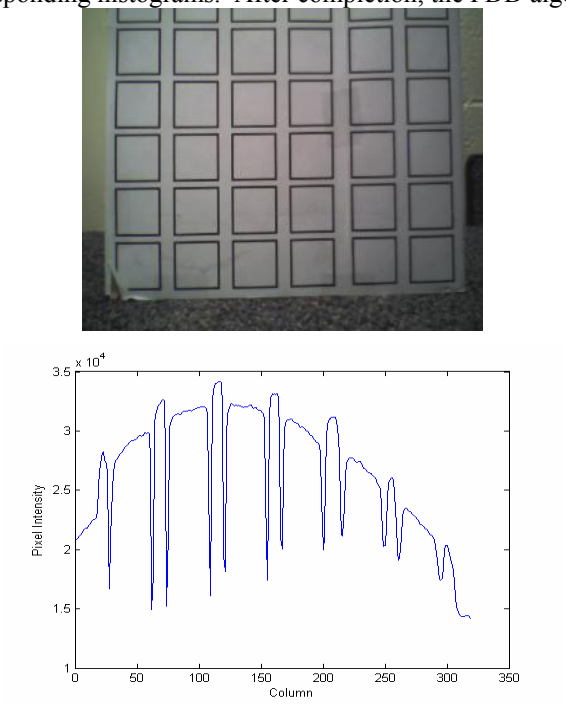

Figure 12: First image with histogram.

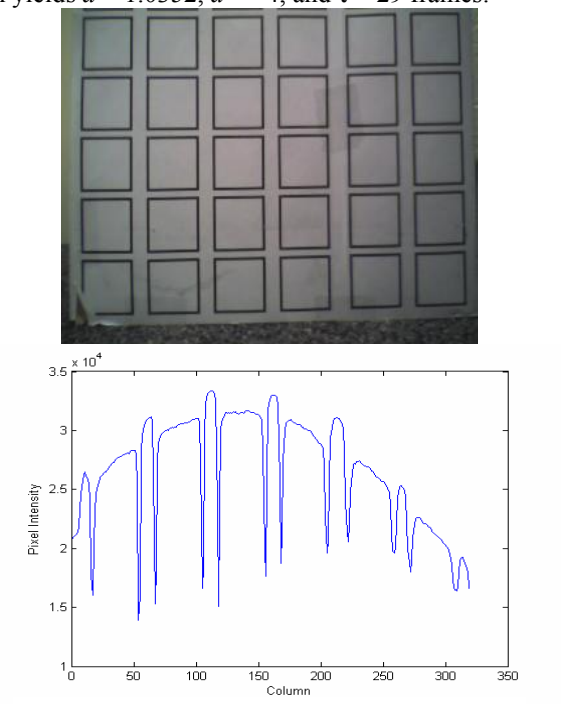

Figure 13: Second image with histogram

7.2 Hardware Implementation

This FDD implementation has been developed at the Robotic Vision lab at Brigham Young University. The architecture is shown in figure 14. It was designed with speed in mind, and as such, each module is pipelined for efficiency. We use two histograms to represent the previous and current histograms. To save on space and time, we simply alternate back and forth between which histogram is the previous histogram and which is the current histogram. The *prev* signal, provided by the controller, is used to specify which of the two histograms the previous histogram is. The other, by default, is the current.

Each histogram is stored in a dual port BRAM. This component is part of the FPGA board and provides synchronous reads and writes. The pixel data comes in to our FDD module in row major order. The first incoming pixel data is the

upper right pixel of the image, followed by the next pixel in the row, and so on until the row is finished. The pixel data came in pairs of chrominance and luminance, of which only the luminance value is of interest to us.

The active histogram module sums the incoming pixel data with either zeros when this is the first row or with the output of the BRAM otherwise. The result is a running total of the pixel data as it comes in to our system. The write enabler module determines which histogram is active. The address select module selects the address to use when reading from or writing to the BRAM. Information is provided by both the SAD module and the controller to access the data in the histograms when the data is needed.

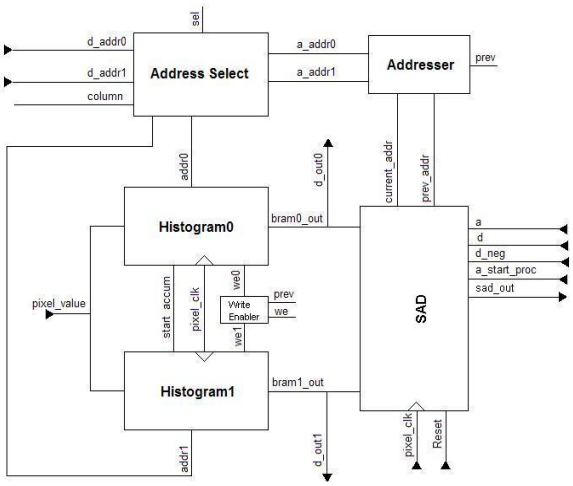

Figure 14: FDD Block Diagram

The SAD module calculates the Sum of Absolute Differences (SAD) for each *a* and *d* sent by the user to the FDD module. Because it requires additional control beyond that provided by the original FDD module, it includes its own embedded controller, which will be discussed later. The absolute difference of each image pair is accumulated and eventually output at the conclusion of the process. The last module of our FDD architecture, the addresser, is responsible for matching the current address and the previous address with the histogram which it is associated with.

As mentioned previously, there are two control modules in our architecture. The first, the FDD controller, controls the writing of the current histogram and is coupled with the frame grabber for improved performance. The second, titled the SAD controller, controls the generation of the SAD required to calculate the time to impact.

By using the signals coming from the camera, we can accurately histogram the image. For instance, on the reception of the *frame_valid* signal, the row counter is set to zero. As long as the row counter is zero, the *start_accum* signal is set high, allowing for the initialization of the histogram. Once the entire image has run through the frame grabber, the *prev* signal is changed so that the histogram module to be used for the current image will be ready to receive it.

The SAD controller is a bit more complex. As mentioned previously, it is necessary to ensure that the values used to index the histograms are in bounds. This is accomplished by implementing a simple state machine. The machine sits in an idle state until it receives an *a* and *d* from the user. It then calculates the current histogram's index, as specified in (1). If this value is less than 0, meaning the feature has moved off the left side of the image, or if it is more than 319, meaning the feature has moved off the right side of the image, than the address isn't changed and the value coming from the subtraction module is ignored.

Until a difference is considered valid, the state machine sits in a start state, during which the *start_accuracy* signal remains high. This ensures that the first difference is stored in the register. It moves to the summing state after leaving the start state, and while in this state, the SAD module continues to sum the differences.

8. CONCLUSIONS

By using FPGA technology, we were able to implement several robot vision algorithms in real time. The competitions created for a graduate level class at Brigham Young University allowed us to test these algorithms in a controlled environment. One of the main problems that our team had with the competitions was that our team did not have the experience necessary to control the robot as we desired to. As we attempted to speed our robot up, we were unable to drive straight or turn accurately. In order to go where we wanted, we had to move more slowly than we desired. We attempted several different forms of feedback to control our movement, but the reactions of the robot to our changes seemed sporadic and random, and without the necessary experience in control, we were unable to improve our performance.

The competitions, including all they entailed, served as a good test for the various real time vision algorithms we wished to explore. For instance, the FDD algorithm was tested quite thoroughly in the Mouse-in-a-Box competition and in the See and Avoid competition, where we were able to cleanly avoid collisions. In creating the hardware module, we had to deal with many constraints, including the space limitations on the FPGA, and we look forward to faster and larger FPGAs being made available that will assist in the future of this work.

We are also looking at revising the competitions to allow for a more thorough testing of the algorithms. The Mouse-in-a-Box competition and the See and Avoid competition will be combined to form a simple maze with obstacles to test obstacle detection, obstacle avoidance, and path planning. The Stereo Vision Competition will remain largely unchanged. The Grand Challenge will include more obstacles placed in the course. Finally, the Urban Canyon Following will include curves and turns rather than being a sharp line. We hope that these changes will ensure a better iteration of hardware/software algorithms for autonomous robots.

REFERENCES

1. T.H. Drayer, W.E. King, IV, J.G. Tront, R.W. Conners, P.A. Araman, "Using Multiple FPGA Architectures for Real-Time Processing of Low-Level Machine Vision Functions," in *Proceedings of the 1995 IEEE IECON 21st International Conference on Industrial Electronics, Control, and Instrumentation*, pp. 1284-1289.
2. W. Luk and T. Wu, "Hardware-Software Codesign of Multidimensional Programs," in *Proceedings of the 1994 IEEE Workshop on FPGAs for Custom Computing Machines*, pp 82-90.
3. S.H. Hajimowlana, G.A. Jullien, R. Muscedere, and J.W. Roberts, "Efficient Pre-Processing Algorithms for an FPGA Based In-Camera Video-Stream Processing System for Industry Inspection," in *Proceedings of the IEEE 1997 Canadian Conference on Electrical and Computer Engineering*, Vol. 2, pp. 835-838.
4. J. Velten and A. Kummert, "High-Speed FPGA-Implementation of Multidimensional Binary Morphological Operations," in *Proceedings of the 2003 International Symposium on Circuits and Systems*, Vol. 3, pp III-706 – III-709.
5. M. Shawky, S. Bonnet, S. Favard, and P. Crubille, "A Computing Platform and its Tools for Features Extraction from On-Vehicle Image Sequences," in *Proceedings of the 2000 IEEE Conference on Intelligent Transportation Systems*, pp. 39-45.
6. G. Hartmann and B. Mertsching, "A Hierarchical Vision System," in *Proceedings of the Intelligent Vehicles 1992 Symposium*, pp. 18-23.
7. T.H. Drayer, J.G. Trong, R.W. Conners, and P.A. Araman, "A Development System for Creating Real-Time Machine Vision Using Field Programmable Gate Arrays," in *Proceedings of the 32nd AnnualHawaii International Conference on System Sciences*, pp. 5-9.
8. A. Benedetti and P. Perona, "A Novel System for Real-Time Low-Level Vision," in *Proceedings of the 1999 IEEE International Symposium on Circuits and Systems*, pp. 500-503.
9. A.J. Shelly and N.L. Seed, "Animate Vision Demonstrator Utilising Reconfigurable System Designs," in *Proceedings of the 1995 Fifth International Conference on Image Processing and its Applications*, pp 500-504.
10. A. van Inge and L.O. Hertberger, "A Hybrid Architecture for a High Performance and Physical Small Low-Level Image Processing System," in *Proceedings of the 11th IAPR International Conference on Pattern Recognition*, pp. 70-74.
11. C. Torres-Huitzil and M. Arias-Estrada, "An FPGA Architecture for High Speed Edge and Corner Detection," in *Proceedings of the 2000 IEEE International Workshop on Computer Architectures for Machine Perception*, pp. 112-116.
12. R. Cucchiara, M. Piccardi, A. Prati, and N. Scarabottolo, "Real-Time Detection of Moving Vehicles," in *Proceedings of the 1999 International Conference on Image Analysis and Processing*, pp. 618-623.
13. J. Aranda, J. Climent, and A. Grau, "A FPGA Implementation of a Video Rate Multi-Rate Tracking System," in *Proceedings of the 1998 Euromicro Conference*, pp. 70-73.
14. S. Choi, Y. Chung, and V.K. Prasana, "Configurable Hardware for Symbolic Search Operations," in *Proceedings of the 1997 International Conference on Parallel and Distributed Systems*, pp. 122-131.
15. F. Marino, E. Stella, A. Branca, and A. Distante, "Specialized Hardware for Real Time Navigation," in *Proceedings of the 1997 IEEE Conference on Intelligent Transport Systems*, pp. 111-116.

16. A. Darabiha, J. Rose, and J.W. Maclean, “Video-Rate Stereo Depth Measurement on Programmable Hardware,” in *Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Vol. 1, pp I-203 – I-210.
17. N. Sudha, "An Area-Efficient Pipelined Array Architecture for Euclidean Distance Transformation and its FPGA Implementation," in *Proceedings of the 17th Annual International Conference on VLSI Design*, pp. 689-692.
18. C.H. Luk, C. Gao, D. Hammerstrom, M. Pavel, D. Kerr, “Biologically Inspired Enhanced Vision (EVS) for Aircraft Landing and Guidance,” in *Proceedings of the 2004 IEEE International Joint Conference on Neural Networks*, Vol. 3, pp. 1751-1756.
19. H. Ziegler, B. So, M. Hall, and P.C. Diniz, “Coarse-Grain Pipelining on Multiple FPGA Architectures,” in *Proceedings of the 10th Annual IEEE Symposium on Fiel-Programmable Custom Computing Machines*, pp. 77-86.
20. C. Torres-Huitzil, S.E. Maya-Rueda, and M. Arias-Estrada, “A Reconfigurable Vision System for Real-Time Applications,” in *Proceedings of the 2002 IEEE International Conference on Field-Programmable Technology*, pp. 286-289.
21. J.A. Tyrrell, J.M. LaPre, C.D. Carothers, B. Roysam, and C.V. Stewart, “Efficient Migration of Complex Off-Line Vision Software to Real-Time System Implementation on Generic Computer Hardware,” *IEEE Transaction on Information Technology*, vol. 8, iss. 2, June 2004, pp. 142-153.
22. D.J. Lee, R.W. Beard, P.C. Merrell, and P. Zhan, “See and Avoidance Behaviors for Autonomous Navigation,” *SPIE Optics East, Robotics Technologies and Architectures, Mobile Robot XVII*, vol. 5609-05, October 2004, pp 23-34.
23. B. Webb and R. Harrison, “Eye and Ears: Combining Sensory Motor Systems Modelled on Insect Physiology,” in *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, pp. 3913-3918.
24. A. Borggi, M. Cellario, P. Lombardi, and M. Porta, “An Evolutionary Approach to Visual Sensing for Vehicle Navigation,” *IEEE Transactions on Industrial Electronics*, vol. 50, iss. 1, Feb. 2003, pp. 18-29.
25. K.M. Iftekharuddin and G. Power, “A Biological Model for Distortion-Invariant Target Recognition,” in *Proceedings of the 2001 International Joint Conference on Neural Networks*, pp. 559-56.