Low-Cost Stereo Vision on an FPGA

Chris Murphy, Daniel Lindquist, Ann Marie Rynning
Thomas Cecil, Sarah Leavitt, Mark L. Chang
Franklin W. Olin College of Engineering
Needham, MA 02492
mark.chang@olin.edu

Abstract

We present a low-cost stereo vision implementation suitable for use in autonomous vehicle applications and designed with agricultural applications in mind. This implementation utilizes the Census Transform algorithm [5, 6] to calculate depth maps from a stereo pair of automotive-grade CMOS cameras. The final prototype utilizes commodity hardware, including a Xilinx Spartan-3 FPGA, to process 320x240 pixel images at greater than 150 frames per second and deliver them via a USB 2.0 interface.

1. Background

Obtaining an accurate, three-dimensional model of unstructured outdoor environments remains a primary challenge of field robotics. Despite widespread robotics use, at $7,000-$20,000 LIDAR remains a cost-prohibitive technology for many applications. Our work seeks to develop a distance sensor package at much lower cost than contemporary options, enabling a variety of new applications and research where cost is an important design factor. Our prototype achieves performance comparable to an ASIC, yet costs less than a dedicated PC and software stack.

Related work includes custom ASIC and software implementations, and only one significant FPGA implementation. In [4] a multiple FPGA implementation on a PCI card was developed, consisting of 16 Xilinx 4025 FPGAs connected in a partial torus with 16 one-megabyte SRAMs. This solution achieved 42 frames per second at 320x240 pixel resolution. Our implementation can be thought of as a modern, low-cost and high-performance descendant of this initial FPGA-based design.

1.1. Stereo Vision

Stereo vision systems use a pair of cameras oriented much like human eyes; each camera views approximately the same "scene", but from a different viewpoint. By examining features and objects from one camera image and finding their relative locations in the other camera image, it is simple to calculate the range of those objects from the camera. Objects that are very far from the cameras will appear in the same relative location within the two frames, while objects that are very close to the camera will have a large horizontal shift, or disparity, from one camera's viewpoint to the other.

We chose to implement the Census Transform [6] after evaluating algorithms from [1–3, 6], due to its high performance on our agricultural datasets. The Census Transform compares a "window" of pixel intensities around a central pixel to several candidate "matching" windows in a second image. This comparison is made by creating a bitstring for each window, which represents the brightness of each pixel in the window relative to the central pixel. The "window" from the second image with the smallest Hamming distance from the primary "window" is determined to be the best match for the pixel. The difference in the pixel "window" positions between the two camera frames is inversely related to the distance of an object from the camera. The greater the distance between matching "windows", the closer an object is to the camera pair. The highly parallel nature of the Census Transform makes it an excellent candidate for FPGA acceleration.

2. Implementation

Our hardware implementation was done entirely with a single Xilinx Spartan-3 FPGA. We determined experimentally that a 13x13 pixel window and a maximum search disparity of 20 pixels provided a good balance between computational complexity and algorithm performance for our application. The top-level algorithmic block diagram of the *correspond* module in the FPGA is shown in Figure 1.

The *correspond* module takes data streaming from our left and right CMOS imagers, synchronizes them through a series of delay units, and performs a Census Transform

0-7695-2940-2/07 $25.00 © 2007 IEEE
DOI 10.1109/FCCM.2007.44

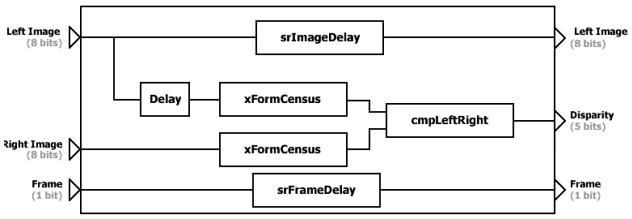

Figure 1. The Correspondence Module.

(*xFormCensus*). The *cmpLeftRight* module calculates and finds the minimum Hamming distance. The two delay modules *srImageDelay* and *srFrameDelay* simply delay a single whole output image and the frame synchronization signals, respectively, in order to synchronize results with the disparity output.

We eliminated the need for a framebuffer by performing calculations as image data is streamed in from the cameras. We need 13 pixels of each row for each window comparison, while the rest of the row of pixels needs to be stored for future comparisons but not individually addressable. Block RAM-based shift registers *srDualBram* are used to store a full row of image pixels minus the pixels in the comparison window in the Census Transform module (shown in Figure 2). The "exposed" pixels are stored as conventional shift registers (*srWindowWidth*). Between these two storage elements, all 320 pixels in 12 rows of each image are stored, plus 13 additional pixels (from the 13th row, necessary to complete a 13x13 "window" of pixels).

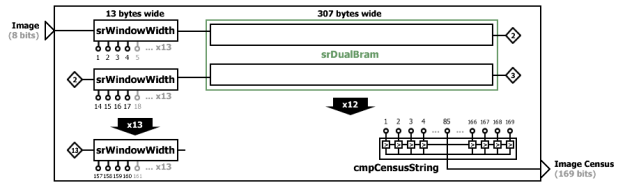

Figure 2. Data storage within the Census Transform module.

3. Results

Our final prototype utilizes a NuHorizons Spartan-3-2000 board with a Xilinx Spartan-3 XC3S2000, two low-cost OmniVision CMOS imagers designed for automotive applications, and a Bitwise Systems QuickUSB module to provide a USB 2.0 interface between the FPGA and a host system. The total cost of the prototype hardware was just over $1,000 using development kits in single quantities. In modest quantities, we expect the cost per system to be far less than $1,000.

Our current design achieves camera-rate performance—approximately 40 320x240 frames per second—from image capture to display on the host computer. This design utilizes 57% of the logic resources on the Xilinx XC3S2000, 26 of the 40 BRAMs, and can be clocked up to approximately 26MHz. With appropriate cameras, the design could process 320x240 8-bit greyscale images at over 150 frames per second.

Figure 3. Example results along an access road. Bright pixels are closer.

Example results are shown in Figure 3. These images do not include any post-processing, which would further improve results. Edge effects have resulted in erroneous results to the left side of the disparity map. In both figures, one can see accurate depths for major features such as the road, trees, and bushes.

The authors thank John Deere for providing technical guidance and financial support.

References

[1] M. Z. Brown, D. Burschka, and G. D. Hager. Advances in computational stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8):993–1008, 2003.

[2] K. Mühlmann, D. Maier, J. Hesser, and R. Männer. Calculating dense disparity maps from color stereo images, an efficient implementation. *International Journal of Computer Vision*, 47(1-3):79–88, 2002.

[3] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1-3):7–42, 2002.

[4] J. Woodfill and B. V. Herzen. Real-time stereo vision on the parts reconfigurable computer. In *IEEE Symposium on FPGAs for Custom Computing Machines*. IEEE, April 1997.

[5] R. Zabih. *Individuating Unknown Objects by Combining Motion and Stereo*. PhD thesis, Stanford University, 1994.

[6] R. Zabih and J. Woodfill. Non-parametric local transforms for computing visual correspondence. In *ECCV '94: Proceedings of the third European conference on Computer Vision (Vol. II)*, pages 150–158, 1994.