

# Calculating Optical Flow using FPGAs

Michael Chisholm (u4212270)

March 20, 2008

## 1 Project proposal

### 1.1 Background and motivation

Vision processing is computationally expensive. Large amounts of data in the form of frames of pixels must be processed to extract meaningful information from raw visual data. For example, to achieve real-time performance when processing 240 rows of 320 columns of pixels at 30 frames per second requires new data to be examined at over 2.3 million pixels per second. This computational speed requirement translates physically into weight, size, and cost.

One such visual task is extracting optical flow information from a series of images. Optical flow is the representation of the motion of objects in a visual scene. Motion is typically represented as a vector field, with each vector approximating the apparent motion of a pixel or group of pixels within an image. Many optical flow extraction algorithms operate on local neighbourhoods of pixels across a series of images, with many such neighbourhoods being processed at once. Since the neighbourhoods can be considered independently of each other, many parts of the image can be processed in parallel. Also, the algorithms are often pipelined, where a series of processing steps are performed on the images in a strictly linear fashion and results from one step are fed into the next.

For such a task, Field-Programmable Gate Arrays (FPGAs) are ideal. FPGAs are microchips that contain a very large number of logic blocks. These logic blocks operate simultaneously and can be modified and connected in various ways so that an FPGA can be programmed to act like virtually any digital circuit that will fit within the provided number of logic blocks. Due to their nature, FPGAs are well suited to tasks that can be pipelined and/or parallelized. They also draw less power and consume less space than typical desktop machines.

It is therefore desirable to find a way of implementing optical flow algorithms on an FPGA with the eventual goal of creating a self-contained camera and processing unit that can be used in embedded applications.

### 1.2 Requirements

This project will investigate the suitability of using FPGAs for calculating optical flow. Current optical flow algorithms will be evaluated for their suitability to the task. Other group's efforts at implementing optical flow algorithms on FPGAs will be examined and benefits and deficiencies in their implementations

will be identified. Following the review, the feasibility of implementing a specific algorithm using an FPGA will be assessed. Various efficiency versus accuracy trade-offs will be examined with the eventual aim of creating a stand-alone device that can give accurate optical flow information from its environment in real time.

From an initial literature review five completely different techniques of calculating optical flow were found[3]. There were various algorithms that applied the techniques with variations in assumptions and constraints. As discussed by Liu [6], differential techniques are the least computationally intensive while still giving accurate results and are projected to be capable of running in real time on modern hardware. Two differential algorithms, one by Horn and Shunk[2] and one by Lucas and Kanade[1] have previously been implemented using FPGAs with varying degrees of speed and accuracy[7, 4].

Another differential algorithm, created by Uras et al.[8], offers a better trade-off of accuracy and density of vectors[5] at only a slight increase in computational cost[6]. It has not previously been implemented on an FPGA, so this research will explore the issues involved in doing so. Using the same benchmarks as Barron et al.[5], the accuracy of the resulting design will be tested and compared to previous FPGA optical flow implementations.

The implementation will be done in hardware, with simulations performed on a PC during the design phase. Image data will be inputted and results outputted via data transfers to and from a PC. Interfacing a camera to the FPGA is not an immediate goal of this research. Real time performance will be evaluated based on pixels per second, frames per second, and the maximum image size that the design can handle.

## 2 Time line

- Weeks 1-3
  - Background research into various optical flow algorithms
  - Requirements specification
- Week 4
  - Initial presentation
  - Research and learn a language used to program FPGAs
- Weeks 5-7
  - Write literature review draft
  - Design
  - Begin implementation
- Mid-semester break
  - Write final copy of literature review
  - Further implementation
- Weeks 8-11
  - Refine implementation
  - Test various
- Week 9
  - Write outline of final report
- Weeks 12-13
  - Finalise design and results
  - Write draft report
- Week 13
  - Final presentation
- Week 14
  - Finish and submit report

## References

- [1] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.
- [2] B. K. P. Horn and B. G. Shunck. Determining Optical Flow. *Artificial Intelligence*, 17:185–203, 1981.
- [3] S. S. Beauchemin and J. L. Barron. The computation of optical flow. *ACM Computing Surveys*, 27(3):433–467, 1995.
- [4] J. Diaz, E. Ros, F. Pelayo, E. M. Ortigossa, and S. Mota. FPGA-based real-time optical-flow system. *IEEE Transactions on Circuits and Systems for Video Technology*, 16(2):274–279, February 2006.
- [5] J.L. Barron, D.J. Fleet, and S.S. Beauchemin. Performance of Optical Flow Techniques. *International Journal of Computer Vision*, 12(1):43–77, 1994.
- [6] Hongche Liu, Tsai-Hong Hong, Martin Herman, Ted Camus, and Rama Chellappa. Accuracy vs efficiency trade-offs in optical flow algorithms. *Computer Vision and Image Understanding: CVIU*, 72(3):271–286, 1998.
- [7] P. C. Arribas and F. M. H. Maciá. FPGA implementation of the Horn & Shunk Optical Flow Algorithm for Motion Detection in real time Images. *Proceedings of the XIII Design of Circuits and Integrated Systems Conference*, pages 616–621, 1998.
- [8] S. Uras, F. Girosi, A. Verri, and V. Torre. A Computational Approach to Motion Perception. *Biological Cybernetics*, 60:79–87, 1988.