6.375 Proposal: Adaptive PIV

Robin Deits
rdeits@csail.mit.edu

March 31, 2013

1 Background: PIV

Particle Image Velocimetry (PIV) is an optical approach to measuring the flow field of a fluid, and has been used in the study of combustion, water flow, robotics, and many other fields. It involves seeding a fluid with tracking particles and using a laser or other planar lighting system to capture sequential images of the particle positions in a single thin 2D slice of the fluid. By comparing the change in position of groups of particles between the subsequent frames, a measurement of the local flow vector can be computed for each region of the fluid. This process of determining the movement of each section of the image is extremely time-consuming in a sequential programming system, but can be readily parallelized to significantly improve performance [3]

Each PIV computation is performed on a pair of sequential images. Computation of the fluid flow begins by dividing the image up into small windows of, for example, 64px on a side. A small window size helps ensure that all of the particles within the window move with the same velocity between the two frames. For each window, we extract the subimage corresponding to that window from the first image in the pair. We will call this subimage A. We then extract a set of subimages $B_{\Delta x, \Delta y}$ by shifting the original window in two dimensions and extracting the corresponding subimages from the second image in the pair. We can then perform a cross-correlation between A and each $B_{i,j}$ and determine the shift in x and y which maximizes the correlation. This gives the most likely location of the particles from window A in the second frame, and thus indicates the movement of that section of the fluid between the frames.

2 Adaptive PIV

Standard PIV algorithms involve an even spatial distribution of interrogation windows A with a fixed window size and some fixed overlap, such as 64 px windows beginning every 16 px. However, in order to achieve sufficient accuracy in busy fluid flows, it can be necessary to choose very small windows or very high degrees of overlap, which increases the computational demands by requiring far more cross-correlation computations. Theunissen et al. proposed a method for

improving the performance of PIV in sub-optimial conditions, called Adaptive PIV [2]. Their method uses information about the current density of seeding particles and the prior estimate of the velocity field to update the size and spatial frequency of the interrogation windows A. This has the effect of increasing the number of data points in the busiest (highest particle density and highest velocity) parts of the fluid and reducing the number of samples in the most stable areas of the fluid, which can improve the amount of relevant data collected per computational unit.

In this project, I will focus on implementing Adaptive PIV on an FPGA to improve computational performance, with the ultimate goal of allowing accurate real-time fluid tracking. I will be expanding on prior work implementing a standard PIV algorithm on an FPGA [3]. I will also be using a recent MATLAB implementation of the Adaptive PIV algorithm by Samvaran Sharma of the Robot Locomotion Group at MIT CSAIL as the reference code for my implementation.

The primary benefit of this project should be the parallelization and speedup of the Adaptive PIV algorithm. In order to achieve the desired image size and accuracy, Sharma's current software requires approximately 2.5 seconds per pair of frames, which makes real-time analysis of the fluid flow impossible. In contrast, Yu et al. were able to compute 15 image pairs per second using their FPGA implementation. My goal will be to achieve this result with the added benefits of the adaptive algorithm's focus on the most important areas of the fluid flow.

3 Design Plan

Over the course of this project, I will iterate through the transformation from a reference sequential program into an FPGA implementation of the PIV algorithm. At each step, I will use a consistent set of reference images to ensure correct PIV results. A rough outline of the design iterations is as follows:

- Adapt the reference algorithm to use fixed-point arithmetic. This will be accomplished by the use of the MATLAB fixed point toolbox, described in [1].
- Move cross-correlation computation to the FPGA. Using the existing reference code, I will select the windows A and a single $B_{i,j}$ and pass them as arrays over the Scemi interface to the FPGA. The chip will then compute the cross-correlation and return a fixed-point result.
- Parallelize/pipeline cross-correlation computation. Rather than transmitting a single pair of windows A and $B_{i,j}$, I will transmit both whole images and the position of window A over the Scemi interface and determine the position of the window $B_{i,j}$ with the highest cross-correlation.
- Parallelize/pipeline window selection. At this point, I should be able to implement the adaptive window selection algorithm on the FPGA. The

reference algorithm will send only the image pair and the prior velocity estimate, and the FPGA will perform the selection of each window A and the calculation of its best corresponding window B.

4 System Diagram

A general outline of the Adaptive PIV system is as follows:

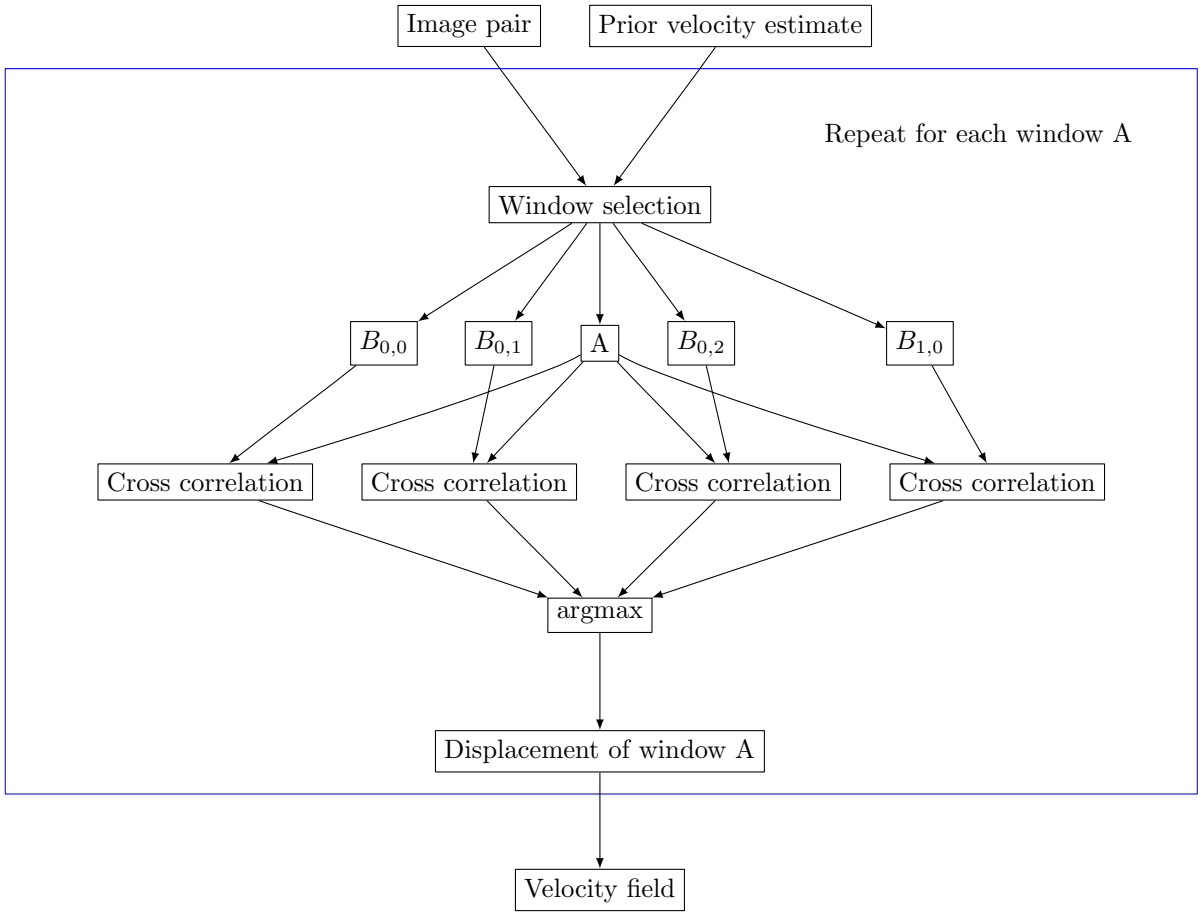

Figure 1: An outline of the adaptive PIV system. Initially, only the cross-correlation will be implemented on the FPGA, but this eventually be expanded to encompass everything within the blue box.

5 Testing Plan

There are a number of existing options for testing PIV algorithms using standardized experimental data or synthetic data. The following resources all provide sample test input for PIV algorithms:

- `http://pivlab.blogspot.com/`: a MATLAB package which provides sample data and a tool to generate synthetic data corresponding to vortices and linear flows.
- `http://www.pivchallenge.org/`: a large collection of sample images, both experimental and synthetic, which are used to compare PIV algorithms.
- `http://www.meol.cnrs.fr/LML/EuroPIV2/SIG/doc/SIG_Main.htm`: a standalone C program to generate synthetic images as input to PIV algorithms

I will choose a suitable set of images which can provide both the image pair for analysis and the prior velocity field estimate.

References

[1] `http://www.mathworks.com/help/fixedpoint/index.html`.

[2] Raf Theunissen, Fulvio Scarano, and Michel L Riethmuller. Spatially adaptive PIV interrogation based on data ensemble. *Experiments in Fluids*, 48(5):875–887, November 2009.

[3] Haiqian Yu, Miriam Leeser, Gilead Tadmor, and Stefan Siegel. Real-time Particle Image Velocimetry for feedback loops using FPGA implementation. *Journal of Aerospace Computing, Information and Communication*, 3(2):52–62, 2006.