Spring 5-1-2011

Real-Time Riverine Particle Image Velocimetry

David William Dobson
University of Southern Mississippi

The University of Southern Mississippi

REAL-TIME RIVERINE PARTICLE IMAGE VELOCIMETRY

by

David William Dobson

Abstract of a Dissertation
Submitted to the Graduate School
of The University of Southern Mississippi
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

May 2011

ABSTRACT

REAL-TIME RIVERINE PARTICLE IMAGE VELOCIMETRY

by David William Dobson

May 2011

A modular particle image velocimetry program was developed and optimized to read and process video of river surface flows from different sensor types. The program was designed for long-term deployment with the ability to sample data continuously in real-time and save the results in a compact format. The time needed to compute a velocity measurement from video input was reduced by using concurrent processing techniques, multi-threading, and a graphics hardware-based correlation algorithm. When used to process field data on a low power Intel Atom based computer the PIV system was capable of computing up to 64 velocity measurements at a rate of 7.5 frames per second. A more powerful computer equipped with a discrete GPU was capable of computing 4800 velocity measurements at a rate of 7.5 frames per second when using the same PIV data and settings. Processing speed of the GPU correlation module was analyzed using a number of different benchmarks. Results show that the GPU-based correlation algorithm has the potential to improve the PIV processing speed of high-end workstations by as much as 2x and low-end portable computers by 10-20x. Methods were also introduced to improve the quality of PIV measurements on river currents. Processing video of river currents with the standard particle image velocimetry technique produced a large number of inaccurate vectors. Most of these inaccurate vectors were correctly identified and removed by using different confidence scoring and filtering techniques. Results from three different experiments were compared to the velocity measurements of other devices to verify the accuracy of the program. These measurements agree to within 16% difference. These results show that accurate PIV measurements of river surface velocity may be computed in real time even on low end and portable computer hardware.

The University of Southern Mississippi

REAL-TIME RIVERINE PARTICLE IMAGE VELOCIMETRY

by

David William Dobson

A Dissertation
Submitted to the Graduate School
of The University of Southern Mississippi
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

Approved:

Andrew Strelzoff

Director

Ray Seyfarth

Todd Holland

Beddhu Murali

Joe Calantoni

Susan A. Siltanen

Dean of the Graduate School

May 2011

ACKNOWLEDGMENTS

My thanks are extended to the Seafloor Sciences Branch of the Naval Research Laboratory. Without the assistance provided by Dr. Todd Holland, Dr. Joe Calantoni, David Young, Tim Kooney, and Tim Sliwinski, most of the work presented in this paper would not have been possible. I would also like to thank my advisor, Dr. Andrew Strelzoff, for his help during my time as a graduate student at USM. Finally, I extend thanks to my USM committe members, Dr. Beddhu Murali and Dr. Ray Seyfarth, and to the staff of the Computer Science Department, especially Crystal McCaffrey, at USM for assisting me over the course of these studies.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure

LIST OF TABLES

Table

LIST OF ABBREVIATIONS

ADCP	-	Acoustic Doppler Current Profiler
AVI	-	Audio Video Interleave
CCOEFF	-	Correlation Coefficient
CCORR	-	Cross Correlation
CUDA	-	Compute Unified Device Architecture
DCC	-	Direct Cross Correlation
DPIV	-	Digital Particle Image Velocimetry
FFT	-	Fast Fourier Transform
FLIR	-	Forward Looking Infrared
FPGA	-	Field-Programmable Gate Array
GPS	-	Global Positioning System
GPU	-	Graphics Processing Unit
LSPIV	-	Large Scale Particle Image Velocimetry
MAD	-	Minimum Absolute Difference
MQD	-	Minimum Quadratic Difference
MSD	-	Minimum Square Difference
NIR	-	Near Infrared
OpenCV	-	Open Source Computer Vision Library
PIV	-	Particle Image Velocimetry
UTM	-	Universal Transverse Mercator

Chapter 1

INTRODUCTION

Particle image velocimetry, PIV, is an optical and computational method used to determine the instantaneous velocity field of a fluid. A typical PIV experiment involves a light source, tracer particles, an imaging device, and a computer. The output of a PIV experiment is a vector map that gives a field of velocity measurements as shown in Figure 1.1. PIV has been successfully used in many different areas of study including fluid mechanics [1] and biology [27]. PIV experiments traditionally take place in a laboratory setting where variables such as particle distribution and lighting can be closely monitored and controlled.

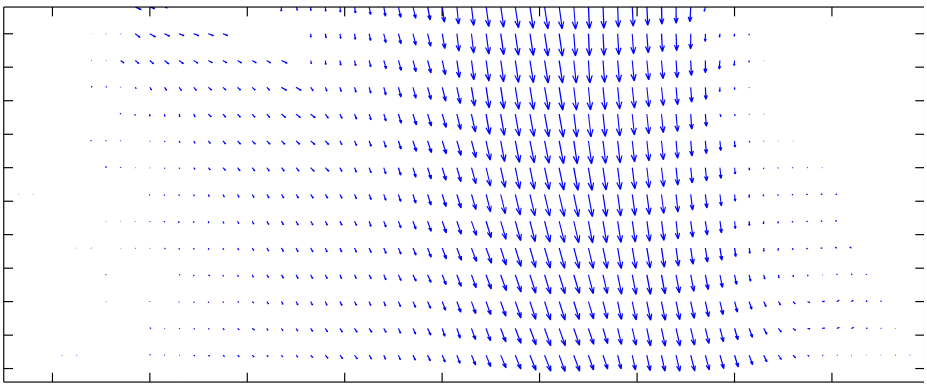

Figure 1.1: Vector Field Generated from PIV.

The use of PIV outside the laboratory is often limited due to the difficulty involved in generating high contrast images in varying lighting conditions, and due to the low density of reflective particles and features in unseeded fluids. Even with these limitations, PIV has been successfully used in a number field experiments to record and analyze the flow of fluids in the natural environment. Surface flow structures in near shore oceanography are analyzed using a video PIV method [13, 12]. PIV is used to collect measurements of underwater turbulent flows in the coastal bottom boundary layer of the ocean [23]. PIV is used to measure fluid flows within stream environments [3], and river discharge is measured using a PIV method [11]. These PIV measurements systems are shown to be accurate, but they are not capable of processing data at the same speed as it is collected from a video sensor.

1.1 Objectives

The ultimate goal of this project is to produce and test all the software components necessary to develop a PIV system capable of being deployed in the field to process data in real-time and over a long period of time. The system would process video from different sensors in real time (currently 7.5 frames per second) and at specific intervals generate output that could be stored to a hard drive or transmitted to a remote machine. Current PIV programs require saving video to a hard drive and later processing it at speeds that are much slower than the capture rate of the video sensor. Because video has to be collected and stored PIV systems cannot be deployed for long periods of time. If the video were processed in real time it would not need to be saved and the device could be deployed for an extended period of time.

PIV software is typically slow due to the computational complexity of correlation. The implementation of a fast GPU based correlation algorithm for PIV is another objective of this work. The GPU algorithm would be combined with an efficient multi-threaded PIV system that separates frame extraction, correlation, and output generation into separate processes. The resulting high performance PIV system should be capable of processing between 500 and 1000 PIV measurements at a rate of 7.5 frames per second.

A number of adjustments must be made to the current PIV methods in order to produce accurate surface flow measurements. Current PIV software is not very good at removing inaccurate vectors. Because features on the surface of a river are often sparse, a large number of inaccurate vectors may be generated. One objective of this work is to develop a method to remove inaccurate vectors by analyzing the correlation surface. In addition, current PIV methods do not properly take advantage of the steady state nature of river flow. Another objective of this work is to utilize many frames of video to produce a final velocity measurement that is more accurate than a single measurement. Accuracy of the final velocity measurements should be within 15 % difference of other accepted measurement techniques.

1.2 Outline

Included in this work is a general introduction to PIV terminology, methods for filtering noisy vectors out during peak detection, a GPU method for spatially computing correlation, a complete PIV program for computing and displaying velocity measurements in real time, and results from three field experiments conducted at the Wolf River in Mississippi and the Kootenai River in Idaho. Chapter 2 discusses the process used to compute PIV and gives an overview of some of the state of the art PIV techniques. Chapter 3 presents an

experiment designed to test the speed at which a number of different algorithms compute PIV and determines what configurations could be used to support processing in real time. Chapter 4 presents the results of two experiments designed to test the capability of methods used to improve the quality of measurements obtained from PIV. Chapter 5 discusses the results from three field experiments where the riverine PIV software was used to determine the surface velocity of a river. Chapter 6 concludes this work by discussing the limitations of the current riverine PIV implementation and possible future improvements.

Chapter 2

BACKGROUND

2.1 Introduction

Particle image velocimetry or PIV is an optical and computational method used to determine the instantaneous velocity field of a fluid. Digital PIV or DPIV refers to the PIV method that uses digital images and computers. The PIV method discussed in this paper is based on 2D DPIV. The process used to compute 2D DPIV is discussed in this chapter. Section 2.2 introduces image collection and processing techniques required for large PIV experiments. Section 2.3 discusses virtual sensors that are used to specify the location where a velocity measurement should be taken. Section 2.4 introduces the terminology used to describe the different correlation windows. Section 2.5 and 2.6 review the cross correlation and error correlation methods. Section 2.7 discusses seed particles and how they may be used to improve measurements. Finally section 2.8 reviews some of the methods commonly used to improve the speed of PIV computations.

2.2 Image Collection and Processing

Multiple image single exposure methods used in digital particle image velocimetry involve capturing one image at time t and another image at time $t + \delta t$. An object in motion will appear at different positions in each image. PIV uses the different position of these objects along with the change in time, δt to compute a velocity field. Multiple image methods are the standard in current PIV technology due in part to inexpensive high resolution digital cameras and in part to modern computers that are capable of storing and processing high resolution images. The term digital particle image velocimetry (DPIV) describes a PIV system that uses a digital camera to capture images and a computer to process them.

$$R = \begin{pmatrix} m_{11} & m_{21} & m_{31} \\ m_{12} & m_{22} & m_{32} \\ m_{13} & m_{23} & m_{33} \end{pmatrix} \tag{2.1}$$

Images collected from a camera need to be registered before PIV is able to use them to produce useful results. Initially PIV methods produce a vector field that gives measurements in pixels per second. In order to produce a measurement in standard units, a conversion from

pixel space to world space must be known. The projection of 3D points onto a 2D surface makes converting from the pixel space to world space difficult. One solution is to reverse the problem and generate a mapping from world space to pixel space using the rotation matrix of the camera [Equation 2.1]. A new image with a uniform conversion from pixel units to world units may be generated from the world space to pixel space mapping.

Generating a mapping from world space to pixel space requires knowing the position, orientation, and physical properties of a camera [13]. Orientation parameters are used in the rotation matrix according to:

$$m_{11} = cos(a)cos(s) + sin(a)cos(t)sin(s) \quad (2.2)$$
$$m_{12} = -sin(a)cos(s) + cos(a)cos(t)sin(s) \quad (2.3)$$
$$m_{13} = sin(t)sin(s) \quad (2.4)$$
$$m_{21} = -cos(a)sin(s) + sin(a)cos(t)cos(s) \quad (2.5)$$
$$m_{22} = sin(a)sin(s) + cos(a)cos(t)cos(s) \quad (2.6)$$
$$m_{23} = sin(t)cos(s) \quad (2.7)$$
$$m_{31} = sin(a)sin(t) \quad (2.8)$$
$$m_{32} = cos(a)sin(t) \quad (2.9)$$
$$m_{33} = -cos(t) \quad (2.10)$$

where a, s, t represent the azimuth, swing, and tilt of the camera respectively. The rotation matrix describes the orientation of the camera and is used in conjunction with the world location of the camera focal point and the angular field of view of the imaging device in the mapping formula:

$$u_n = u_0 - c\frac{m_{11}(X_n - X_0) + m_{12}(Y_n - Y_0) + m_{13}(Z_n - Z_0)}{m_{31}(X_n - X_0) + m_{32}(Y_n - Y_0) + m_{33}(Z_n - Z_0)} \quad (2.11)$$
$$v_n = v_0 - c\frac{m_{21}(X_n - X_0) + m_{22}(Y_n - Y_0) + m_{23}(Z_n - Z_0)}{m_{31}(X_n - X_0) + m_{32}(Y_n - Y_0) + m_{33}(Z_n - Z_0)} \quad (2.12)$$

where u_n, v_n is a position in the original image coordinate system. X_n, Y_n, Z_n is a position in world coordinates. X_0, Y_0, Z_0 is the position of the camera in the target world coordinate system, and u_0, v_0 is the location of the center pixel in the original image. The term c is a constant scaling factor based on the camera lens angular field of view and is computed using $c = u_0/tan(fov/2)$ where u_0 is the location of the center pixel along the horizontal plane of the image and fov is the angular field of view of the camera.

When a world coordinate is plugged into Equation 2.11 and 2.12 a number is produced that gives the location of the world coordinate in the pixel coordinates. A registered image

is generated by iterating through a uniform grid of world coordinates and computing the associated pixel coordinates. These pixel coordinates are used to look-up the intensity value in the original image and copy this value to the registered image. Often the position of the pixel in the original image as computed by Equation 2.11 and 2.12 is not an integer value so an interpolation method must be used to compute an intensity value. If the camera position remains fixed, these calculations only have to be computed once. A mapping table generated from the initial calculations may be used to remap all additional images from the same video stream as long as the position and orientation of the camera do not change.

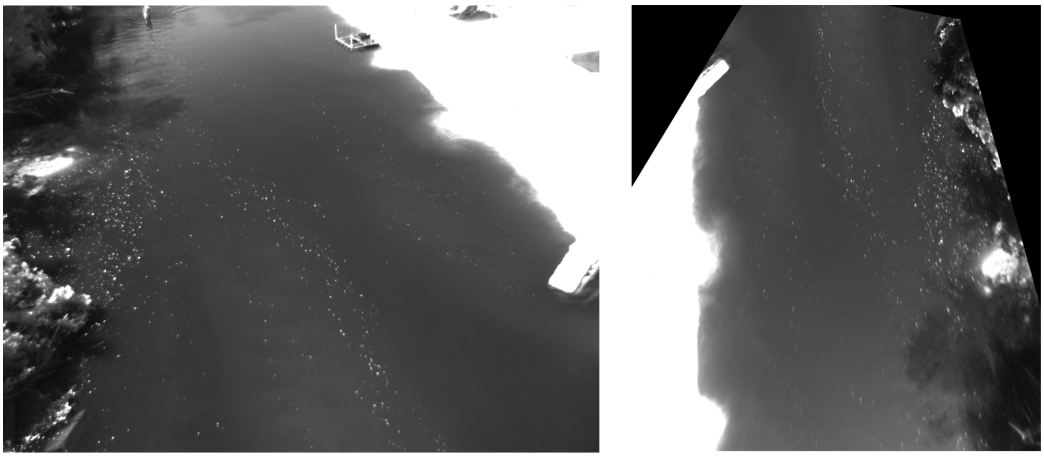

Figure 2.1: Original Image (Left) and Registered Image (Right).

Camera lens distortion should also be taken into account when computing the mapping table. Camera lenses often contain small imperfections that distort the images they collect. Intensity values computed for the remapped image could be incorrect if a significant amount of distortion is present in the original image. Techniques for removing lens distortion are discussed in [13] and [33].

PIV measurements are influenced by the quality of the images used for correlation. Both lighting and image exposure settings play a roll in the accuracy of PIV measurements. An ideal image for correlation would include a large number of features that clearly stand out from the background. When collecting images, both lighting and imaging device exposure should be optimized to produce images with many features.

Preprocessing images may also be helpful in some situations. If the contrast between background and features is low, a threshold algorithm can be applied to the images. Thresholding could result in the loss of some low intensity features. If the number of features is already high this is often acceptable; however, if the number of features is low thresholding may further reduce the number of features available to correlate. A lack of features in an image can not be improved by preprocessing techniques, but may be addressed by adding

seed particles to the fluid. Noise in the image is another problem that can be addressed to some extent by applying a smoothing filter to the images. Like thresholding, applying a smoothing filter to an image may reduce the number or quality of features.

2.3 Virtual Sensors

The PIV algorithm computes velocity measurements at a number of different points within an image referred to as virtual sensors or sensors. Sensors are often placed in a grid pattern but may also be placed at specific locations of interest. At every sensor location a velocity measurement is generated using the PIV algorithm and displayed as one element in a vector field. Sensors are independent and may be processed in parallel. A sensor point remains at the same position throughout the processing stage and is the center for a number of windows used in the correlation algorithms.

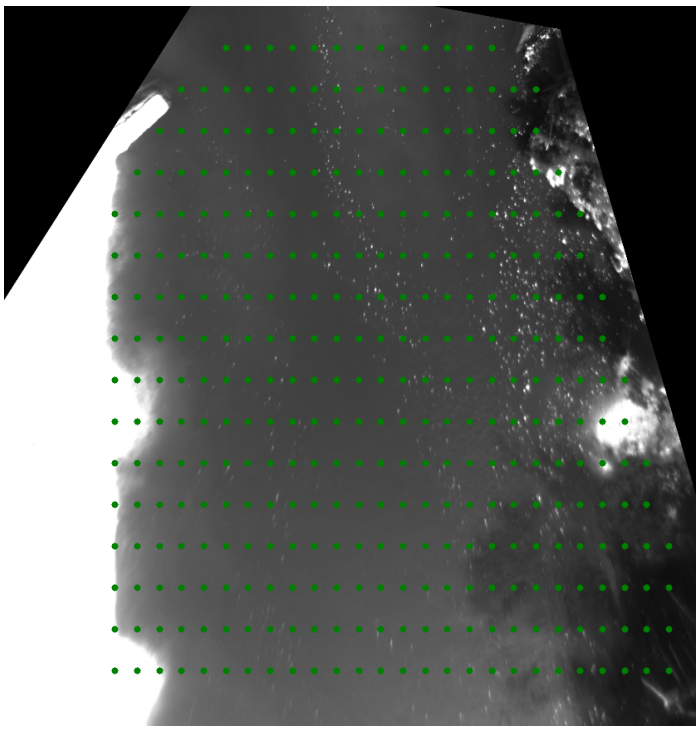

Figure 2.2: Virtual Sensors Placed in a Grid Pattern.

2.4 Correlation Windows

The window of pixels taken from the lag frame centered at the sensor is called a reference window. The reference window represents a feature from the lag image that is to be located in the lead image. The size of the reference window must be set manually and should

generally be larger than 15 pixels. The reference window is compared to the lead image at a number of different locations around the sensor. Each element in the correlation surface represents one of these locations. The number of locations are manually set by specifying a window around the sensor referred to as a correlation surface. A search window is made up of a block of pixels taken from the lead image. The size of the search window is determined from the size of the correlation surface and the size of the reference window according to

$$X_s = X_c + 2 * floor(X_r/2) \tag{2.13}$$

$$Y_s = Y_c + 2 * floor(Y_r/2) \tag{2.14}$$

where X_r, Y_r is the size of the reference window and X_c, Y_c is the size of the correlation surface. Each sensor is independent and may be computed separately from any others. Each vector in the vector field is produced by a sensor so the density of the vector field produced by PIV is determined by the number of sensors.

2.5 Cross Correlation

Finding the best match for a reference window in the search window requires using a statistical technique called correlation. Correlation is an exhaustive search method that generates a score for each possible position of the reference window within the image or within a subset of the image. The best score in the correlation surface represents the best match of the reference window to a sub window of the search window. The correlation surface may be viewed as a height map or as a grayscale image 2.3. The highest peak within the correlation surface is used to generate a velocity measurement for that specific sensor. The most common method used to determine correlation in DPIV is called cross correlation.

Cross correlation is a technique that involves taking two different images, lag and lead, of the flow field from the same camera at the same location but at a different point in time. Cross correlation methods rely on knowing precisely the capture rate of the camera to compute a δt measurement. The amount of time allowed to pass between the lag and lead images should be carefully considered because it affects both how much motion may be detected and how large the correlation surface should be. If the separation between images is too large the PIV calculations will be very expensive and inaccurate because a larger correlation surface is needed. If the separation is too small the motion estimate will rely heavily on sub-pixel interpolation which may result in phase locking.

Cross correlation between reference and search areas may be calculated by using methods that include direct cross correlation and FFT cross correlation, and a number of other methods that combine or extend these two methods [19].

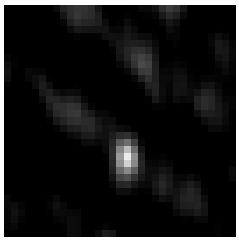

Figure 2.3: Grayscale Representation of a Correlation Surface.

Direct cross-correlation methods search the spatial domain for the best match between the reference and search windows. A number of different correlation methods may be used to determine how well a reference window matches an offset of the search window including cross correlation, minimum quadratic difference, and normalized correlation coefficient. The most simple cross correlation method involves multiplying the pixel intensity values in the reference (r) and search (s) windows according to

$$\sum_{x,y} r(x,y)s(x-u,y-v). \tag{2.15}$$

Normalized correlation coefficient (CCOEFF) is an extension of the cross correlation method where the mean of the search window $\bar{s}$ is subtracted from the search window s and the mean of the reference window is subtracted from the reference window $\bar{r}$ and normalized using autocorrelation according to

$$\frac{\sum_{x,y}[r(x,y)-\bar{r}][s(x-u,y-v)-\bar{s}]}{\sum_{x,y}[r(x,y)-\bar{r}]^2\sum_{x,y}[s(x-u,y-v)-\bar{s}]^2}. \tag{2.16}$$

Removing the mean prevents changes in background intensity from affecting the correlation score. It also improves contrast and removes static features. In PIV the most common method to score correlation is based on normalized correlation coefficient but uses an FFT based method instead of the exhaustive direct search. The FFT method is popular because of its low computational cost with respect to the direct spatial search method. The FFT method involves translating the two correlation surfaces into the frequency domain. Correlation is calculated by multiplying the complex conjugate of the reference window's FFT by the FFT of the search window. The result of this multiplication is translated back to the spatial domain by using an inverse FFT which produces a correlation surface. Normalizing the result of the FFT approximation is computed in constant time using a method called image integral [5].

2.6 Error Correlation

A number of methods for pattern matching are based on error correlation instead of statistical correlation. Error correlation and cross correlation use the same exhaustive search algorithm. The difference between the two is the function used for scoring how well a reference window matches a partition of the search window. A method called minimum absolute difference (MAD) uses the difference between reference and search windows to score each window alignment [8]. The MAD function is given as

$$\sum_{x,y} |r(x,y) - s(x-u, y-v)]|, \tag{2.17}$$

where $r(x,y)$ is the intensity value of the pixel (x,y) in the reference window and $s(x-u, y-v)$ is the intensity value of the pixel $(x-u, y-v)$ in the search window at the offset (u,v). MAD is typically normalized by taking the sum of the reference and search window partition [10]. The MAD method currently does not have a fast implementation and must be computed directly.

Another error correlation method that does have a fast implementation is called minimum square difference (MSD) or minimum quadratic difference (MQD). MQD replaces the absolute value term in 2.17 with a square term. The resulting equation has an FFT approximation. MQD is typically normalized by dividing by the autocorrelation score.

2.7 Seed Particles

Seed particles, often called tracer or scattering particles, are used to create image features when no features exist within a body of particles and are discussed in detail in [24]. In order to compute PIV on a clear fluid, particles must be added that scatter light into the camera. Without the reflective particles there would not be enough variation in pixel intensity to track motion. Seed particles are selected so that they closely follow the flow of the carrier fluid. Illuminating the seed particles enough to be detected by the camera often requires the use of a laser. When used in conjunction with laser lighting the imaging device records the scattered light from seed particles, and scattered light intensity is used as features for the PIV computation. An experiment that uses laser lighting and seed particles correctly will produce data with low amounts of noise which in turn will greatly increases the accuracy of PIV calculations. Adding seed particles to a fluid may not always be possible. If this is the case special care must be taken to ensure that a velocity vector is accurate and not being computed from noise. As long as there is sufficient contrast between the particles and the background, PIV computation may be possible without using laser light or seed particles.

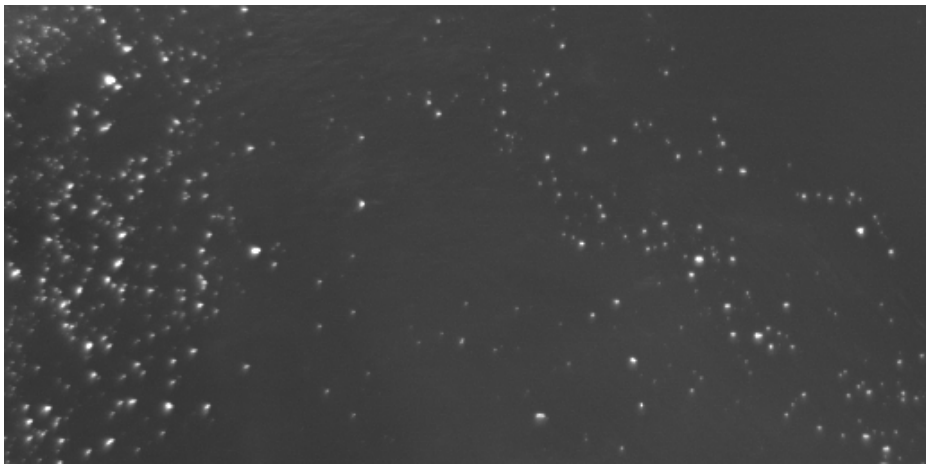

Figure 2.4: Natural particles in the Wolf River.

2.8 Processing Speed

The amount of time required to compute PIV depends on a number of different factors including the method used for correlation and the window size of the interrogation area, but in general PIV computation is expensive. When a direct method is used to compute cross correlation, the computational cost of PIV is $O(n^4)$ where n is based on the size of the reference and correlation surfaces. When an approximation method based on the FFT is used to compute the correlation, the computational cost is reduced to $O(n^2 log(n))$. The FFT method scales better than the direct method but also introduces a significant computational overhead and may in fact be slower for problems that use a small reference window or correlation surface. In some situations the FFT method may not always give a good approximation of the correlation [20].

Summed area table is a technique that may be used to reduce the cost of computing the denominator of the CCOEFF method [5]. To compute the summed area table, create an array the same size as the image being processed. Each element in this array should be filled with the sum of the corresponding pixel and all pixels before it.

$$S = \sum_{x<X, y<Y} I(x,y) \tag{2.18}$$

where I is the intensity value of pixel x, y. Once computed the summed table may be used to compute the mean of any sub-region in the image using

$$\sum_{x_1<=x<x_2, y_1<=y<y_2} I(x,y) = S(x_2,y_2) - S(x_1,y_2) - S(x_2,y_1) + S(x_1,y_1). \tag{2.19}$$

The mean values in 2.16 may be computed using a summed table. The denominator may be computed by generating a square table in the same method as the summed table is generated.

Chapter 3

PROCESSING METHODS

3.1 Introduction

The most significant overhead involved in PIV computations comes from correlation. Correlation when computed directly uses a brute force method that checks every possible offset of the reference window within the search window to find the best match. The direct method has a computation complexity of $O(n^4)$. An alternative FFT based method is commonly used for simple cross correlation and has been extended to work for the correlation coefficient method [4]. The FFT method scales better with a complexity of $O(n^2)$ but has significant overhead and does not always meet or exceed performance requirements for real time systems. Specialized hardware has been proposed as a possible method to further increase the performance of correlation. In this chapter the processing performance of a GPU based direct correlation algorithm is compared to a CPU FFT algorithm and a CPU direct algorithm. The purpose of this chapter is to determine if an efficient PIV software program is capable of processing PIV in real time and what, if anything, would be gained by using specialized hardware to compute correlation.

The remainder of this chapter is divided into four sections. Section 3.2 titled "Related Work" reviews other implementations of correlation on alternative hardware. Section 3.3 titled "EPIV Implementation" discusses the implementation details of a software system called extensible particle image velocimetry or EPIV that was designed to generate PIV measurements of surface flow in real time. Section 3.4 titled "Benchmarks" gives processing performance results for EPIV when used with a number of different settings. The final section, 3.5, describes a mobile platform that could be used to capture video and host the EPIV software.

3.2 Related Work

The FPGA platform has been suggested as a possible hardware solution to improve the processing speed of PIV computations [32, 2]. Tests show that under ideal conditions these FPGA systems are capable of of outperforming the CPU [32, 28]. The FPGA implementation has some limitations with respect to PIV calculations. Multiplication on the FPGA takes

up a large amount of hardware resources limiting the FPGA PIV implementation to direct correlation methods that use maximum absolute difference (MAD). In addition, FPGA programs are not very flexible and would only be able to provide a limited number of adjustable parameters.

Another hardware platform that has been proposed as a possible solution to accelerate PIV computations is the GPU. A modern GPU includes a large number of specialized processors. In the Tesla C1060, for example, there are 240 processing elements. Direct and FFT implementations of PIV on the GPU have been shown to outperform the CPU [28, 22]. Unlike FPGA PIV implementations, GPU based methods are not limited to error correlation. Cross correlation algorithms should process nearly as fast as error correlation methods. More complex PIV techniques such as window deformation and multisampling have also been successfully implemented on the GPU [25].

The work listed above shows that hardware may be used to improve the performance of the correlation components of a PIV system. However, there are other significant sources of computational overhead in PIV not related to correlation that could reduce the software speed. In addition, advanced CPU PIV methods should also be used for speed comparisons with specialized hardware to determine actual benefit. For this reason a complete system using advanced CPU and GPU correlation techniques was developed and tested to measure the speed of computing a large number of PIV measurements on actual data collected in field experiments.

3.3 EPIV Version Three

All the measurements recorded in this section were made using EPIV. The initial version of EPIV was designed to test the speed of PIV calculations when using different advanced processing techniques with the goal of determining if real-time processing is feasible and if any significant speedup is achieved by using specialized hardware. It was also designed to support as many correlation methods and image processing techniques as possible to study the accuracy of these method when applied to computing riverine PIV measurements. EPIV was designed to efficiently utilize multi-core processor by supporting multithreaded correlation calculations and dividing the PIV computation process into three independent modules which may also run concurrently. The threading model and the three different processing modules of EPIV are discussed in detail in the following section.

3.3.1 Threading Model

EPIV is a multithreaded application capable of scaling to at least 16 CPU cores. Threads are implemented using the pthreads library to run the individual modules and with OpenMP within the PIV module. The EPIV program is broken down into five components, a graphical configuration and control interface, a processing control module, an image processing module, a PIV module, and an output module. PIV configuration settings are collected and displayed in the GUI module. The control module creates threads that are used to run the processing modules. The image processing module is responsible for decoding video from a stream and processing the frames to produce a registered image. The PIV module is responsible for computing the placement and correlation score of sensors within the image. Results from the PIV computations are displayed and optionally written to a file in the output module. Each of these modules runs on a different thread. All three modules are executed in parallel and synchronized at the end of processing a single frame pair. Data from the video is extracted, processed, and passed to the PIV modules which in turn passes the results to the output module for display and writing. The output module displays the result of the first frame set while the input module is reading the third frame set, and the PIV module is processing the second frame set. A sleep lock is used to keep the threads synchronized if they do not complete at the same time.

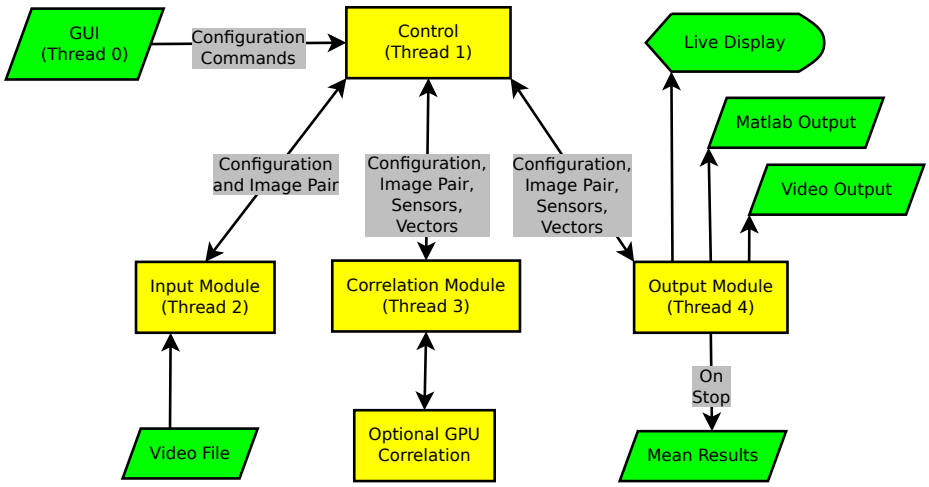

Figure 3.1: EPIV High Level Flowchart.

3.3.2 Graphical Configuration Interface

Riverine PIV requires a large number of configuration parameters. A graphical interface helps to display these settings and allows for quick adjustments. The graphical interface in

EPIV uses the GTK toolkit for the onscreen display and the glib key file methods for saving and loading configuration settings from a text file. The graphical interface also controls EPIV with start and stop operations that interface with the control module. All graphical operations are run on a dedicated thread so the interface never locks up while waiting for computations.

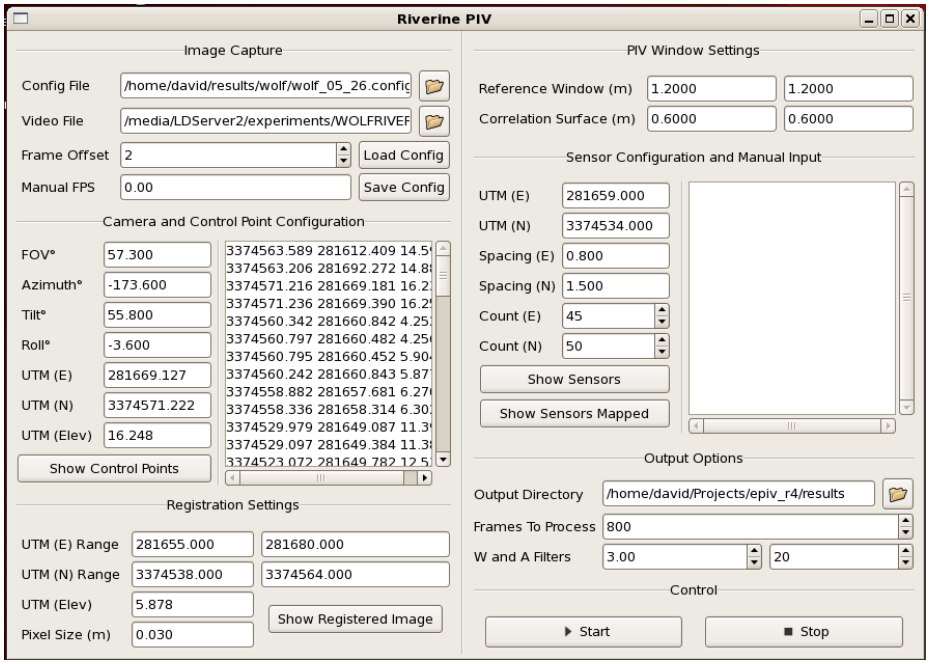

Figure 3.2: Graphical User Interface for EPIV.

3.3.3 Processing Control Module

The control module creates and controls the objects, data, and threads used in the three processing modules whenever a PIV computation is started. Each of the three processing modules requires the configuration object generated by the GUI and additional memory objects used to save and access results. The control module is responsible for passing the needed object references to each individual modules. Synchronization of the three processing modules is also enforced in the control module by using pthread barriers.

3.3.4 Image Processing Module

The image processing module performs three tasks, extraction, mapping, and processing of images. FFMPEG and OpenCV are used to extract images from a video stream recorded by the color and near-infrared (NIR) devices. EPIV also supports direct input from a firewire camera using the dc1394 library. For the forward looking infrared (FLIR) camera, video

is stored in an uncompressed format called sfmov. The image processing module supports reading FLIR video directly from the uncompressed source files. Typically extracting images from video files is the most time intensive operation in the image processing module. Once the video is extracted and converted to gray scale it is remapped to a specific area in world coordinates. The remapping process involves looping through the desired world coordinates and accessing a mapping table that gives the location of the world coordinate in the original image. The mapping table is generated during initialization using camera properties, position, and a number of control points to check for accuracy. The intensity value is interpolated from the original image and stored to a new grayscale image that has a direct mapping from pixel to world space. The final task for the image processing module is to apply any optional preprocessing methods such as smoothing, threshold, and equalization.

3.3.5 PIV Module

The PIV module is responsible for sensor placement, correlation, and vector computation. Sensors may be automatically placed at positions in a grid by specifying the starting point and spacing. The PIV module also supports sensor placement at specific UTM points. Sensors that are too close to the edge of the mapped image or not in the mapped image at all are disabled by the PIV module. The remapped image may include a number of pixels that do not map to the original image. These invalid pixels are marked by the image processing module and passed to the PIV module which removes any sensors that appear in areas of the remapped image that have no intensity information from the original image. All sensor processing is done during an initialization phase of the PIV module.

Correlation is computed in the execution phase of PIV. At this point the sensors are divided into groups that may be processed by different threads. Three different methods are available for the correlation processing. The direct CPU method is a brute force correlation process that checks pixel by pixel for the best possible match of the reference window inside the search window. To simplify the processing within the inner most loop the arrays search-offsets, search-corners, reference-offsets, and reference-corners are generated. Each element in reference-corners points to the corner of a reference window for a specific sensor. The reference-offset array includes one element for every pixel within the reference window and when added to the reference-corner gives the position that pixel within the image. Each element in the search-corners array gives the position of a search window corner for a specific sensor. The search-offset array includes the same number of elements as the correlation surface. When the search-corner and search-offset array are added together they give the corner position used for one comparison of the search and reference windows.

Within the inner most loop computing the location of the search and reference pixel requires only one addition and two array access operations. A call to the direct PIV methods produces a correlation array that contains a score for each element of the correlation surface.

Another option for correlation is the FFT method. The FFT method is implemented using the template matching routines in OpenCV. For each sensor, the reference window and search window are extracted from the image and passed into the template matching routine. OpenCV's template matching implementation supports FFT correlation using cross correlation, coefficient correlation, and error correlation. For multiprocessor systems the loop used to process each sensor is divided and computed on a number of different threads. Sum tables are also used to help improve the processing speed of the FFT methods. The FFT correlation method returns a correlation array with one score for each element in the correlation surface.

A motion vector is generated by determining the location of the peak within the correlation array. Every element in the correlation array is visited and the position and value of the maximum correlation score is extracted. A linear interpolation method is used to compute the sub pixel location of the actual peak. The sub pixel location of the peak is passed through a scoring method that involves computing a confidence measurement of the peak. The computed vector and peak score are stored and passed to the output module.

3.3.6 GPU Correlation Module

Direct correlation methods implemented in CUDA are available in the GPU correlation module. The PIV module can optionally offload the correlation directly to the GPU. The GPU implementation of direct correlation uses the same precomputed variables as does the CPU method. These variables along with the lag and lead images and correlation parameters are passed to the GPU correlation module which returns an array of correlation surfaces. The array containing the correlation surfaces is processed by the CPU peak detection and scoring algorithms.

Correlation on the GPU is separated into CUDA blocks and threads to exploit the parallel nature of the GPU. A separate block is created for every sensor. Within the block the work load is divided using threads that have access to a fast shared memory. Each thread processes a number of iterations of the inner most loop. In order to utilize all of the GPU processing power, the number of threads must be set correctly based on usage of shared memory and registers. The default number of threads used in the GPU module is 256; at this setting the GPU is fully utilized during the PIV computation.

A number of advanced GPU features are used to improve memory access patterns and

increase performance. Reference windows are stored in GPU shared memory. The access time for shared memory is four clock cycles while global memory is in excess of 100 clock cycles. Because pixels in the reference window are accessed many times during the correlation computation this improves performance by a factor of 5-10x. A GPU feature called constant memory is also used to increase the speed of access to index lookup tables. Constant memory is a type of global memory access that is cached. A further speedup of almost 2x was achieved by placing one of the lookup table in constant memory. Texture memory access is another hardware accelerated memory access method available on the GPU. A texture memory lookup uses a hardware cache in addition to alignment methods to improve memory read performance. Using texture memory to access elements in the search window improved performance of the correlation method an addition 2-5x. The final algorithm produced a GPU utilization measurement of over 90%.

There are a number of size restrictions in the GPU implementation. The GPU has 32KB of shared memory available for each block of threads. The reference window is stored in shared memory and must not exceed 32KB in size. Each element in the reference window is an unsigned char so the total number of elements must not exceed 32,000. Constant memory is also limited. Current NVIDIA GPUs have 64KB of constant memory storage available. The lookup table stored in constant memory must also not exceed 64KB. Each element in the lookup table is an unsigned int so the lookup table must not exceed 16,000 elements. Because the size of the lookup table is dependent on the size of the reference window, the reference window must not exceed 16,000 elements in size.

3.3.7 Output Module

The output module shows the result of the PIV computation on a live display and also optionally writes the results to a file or video. Vectors and sensors on the live display are drawn on top of the remapped image. Sensors are represented by a circle and vectors by a line. The length of the line represents the velocity measurement. The lines representing vectors may also be colored according to the velocity measurement. Optionally vectors may be drawn on top of the original unmapped image by using the image mapping function to compute the location of the starting and stopping point of the vector in the original image. A time average of the vectors is also computed and may be displayed in the live display. The optional text output file includes the position of sensors and vectors for each frame and the peak score, *wscore*. The text file is designed to be imported into Matlab for additional post processing.

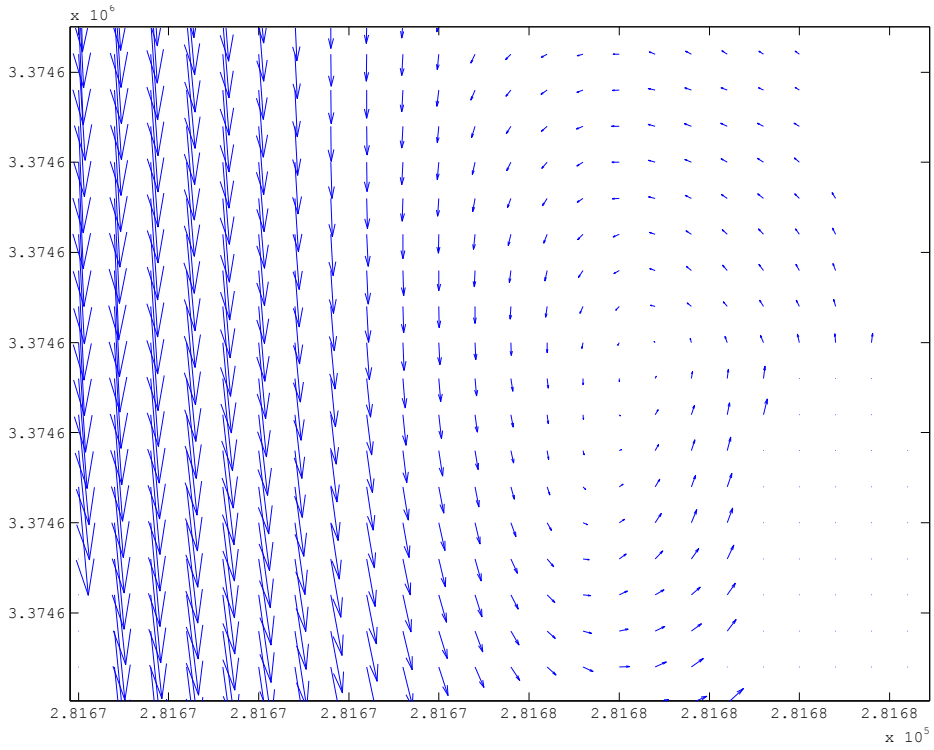

Figure 3.3: Circular Flow Detected by EPIV and Displayed in Matlab.

The live output display records information about the maximum possible velocity and vector color information. During initialization the output module searches for a position that minimizes the amount of image overwritten to place reference vectors. Reference vectors may appear in any corner of the live display. The set of reference vectors include five vectors of different length and a number representing the velocity of the longest vector. The velocity of each of the other vectors is approximately one-fifth smaller than the vector immediately before it.

Once the processing is complete the output module generates a number of images and result files. Two time average images are generated. One image shows the time average vectors drawn on top of the last frame in the video sequence and the other image shows the time average results drawn on an unmapped grayscale image taken of the last video frame. For each of the mapped images an associated world file is written that contains mapping information that may be used by Matlab to place the image on a world grid. Other files saved by the output module include an image file that shows the computed location of the control points in the original image, an image that shows the placement of sensors, and a file that shows the registered image without any objects drawn on top of it. A video file is also generated during the processing that records the live display output of EPIV.

3.4 Benchmarks

Table 3.1: *Benchmark System Specifications*.

Operating System	Red Hat Enterprise Linux 5.5, 64 bit
Processor	Intel Xeon E5530 @ 2.40GHz
Memory	6GB DDR3 @ 1066 MHz
GPU	Nvidia Tesla C1060
Bus	PCI Express x16 Gen 2

Benchmarking EPIV involves setting a number of parameters that have a significant impact on performance. The reference window size and correlation surface size should have a big influence on the performance of any direct processing method because of the $O(n^4)$ scaling. Methods that utilize the FFT to solve correlation will also see a lesser but still significant decrease in performance as window size is increase. Performance is also dependent on the number of sensors at which correlation is computed.

Time spent decoding the video file or displaying results may in some cases cause a bottleneck to occur because the PIV threads must wait on the reader or writer threads. In EPIV the most common bottleneck occurs in the input module while decoding video from a compressed stream. The maximum video decode rate may be determined by running EPIV with no configured sensors. The video reader is a bottleneck if the frame rate of EPIV remains the same when sensors are added back to the configuration. To maximize the decoder performance, an sfmov file was used as a video source for these tests. Sfmov files are stored uncompressed and encoded at a lower resolution than video from the NIR and color sensors.

Table 3.2: *Default PIV Benchmark Settings*.

Sensors	400
Reference Window	35x35
Correlation Surface	35x35
Correlation Method	MSD
PIV Threads	16
Averaging Time	800 frames

All processing performance measurements used the same sfmov video file and EPIV settings except where otherwise specified. In all tests the video file was loaded into cache

memory before taking any performance measurements. Other PIV specific settings used in the performance tests are listed in Table 3.2. EPIV was designed with the performance goal of processing color and NIR video at 7.5 FPS and FLIR video at 15 FPS. The ultimate goal of these performance measurements was to determine if EPIV is capable of processing video in real time and what the limitations of real time processing are.

System specifications for the workstation test computer are given in Table 3.1. The Intel Xeon E5530 is a quad core processor with 8MB of L3 cache running at 2.40GHz. The Tesla C1060 is a CUDA compute capability 1.3 card with 240 processing elements. During all measurements the system was idle except for the EPIV process. In all CPU experiments the PIV module is configured to run 16 processing threads. EPIV sets the priority of the input/output threads higher than the PIV threads. Setting the PIV module to use more threads than CPU cores should not reduce performance significantly when an input or output bottleneck exists, but should improve performance when one does not exist.

Benchmarks are separated into four subsections. Benchmark results generated from changing the size of the reference and search window are given in the subsection titled "Window Sizes." These results determine both the scalability of the different processing methods and any possible significant overhead introduced from changing the size of the correlation or reference window. The next section titled "Sensor Count" benchmarks the performance effect introduced when the number of sensor are changed. Results from this section are expected to show a linear decrease in performance as the number of sensors are increased. The section titled "Methods" investigates the performance differences among the three correlation methods, cross correlation, correlation coefficient, and minimum square difference. A small difference in performance may be expected when computing the different correlation methods directly, but very little difference in performance is expected when using FFT approximation. The final section titled processor count looks at the performance of EPIV as the number of processors is increased. All performance measurements were taken directly from the average frames-per-second display of EPIV.

3.4.1 Window Sizes

Figure 3.4 shows the performance of the PIV computation using the parameters given in Table 3.2. A number of different reference window sizes are given to show how each of the correlation processing techniques scales. The GPU processor computes correlation faster than the CPU in all these tests. With window sizes of $15x15$ and $25x25$ the performance of the GPU was limited by the speed of the video reading thread. GPU correlation performance begins to decline quickly with reference windows of size $35x35$ and up.

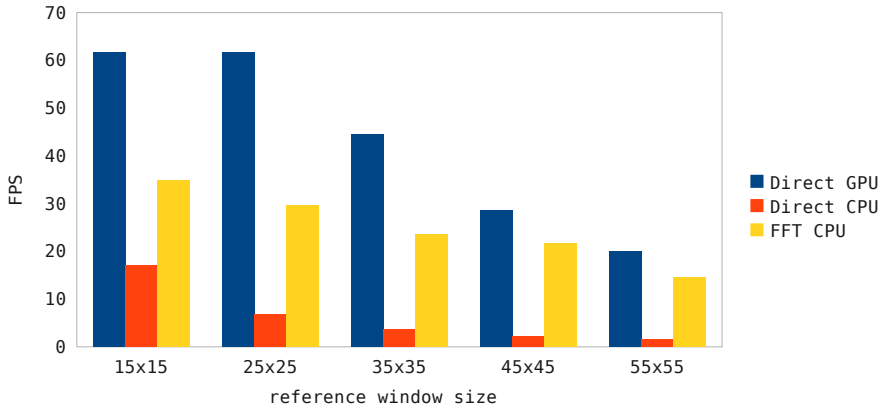

Figure 3.4: Performance of EPIV with different reference window sizes.

If the window sizes were increased beyond 55*x*55 the CPU FFT method would eventually outperform the GPU. Even with a high end four core CPU, the direct CPU correlation method does not give good performance numbers at any window size, and should only be used if the FFT method is not acceptable and no GPU is available. For all window sizes less than 55*x*55 both the FFT CPU method and GPU method are capable of processing the sfmov in real time.

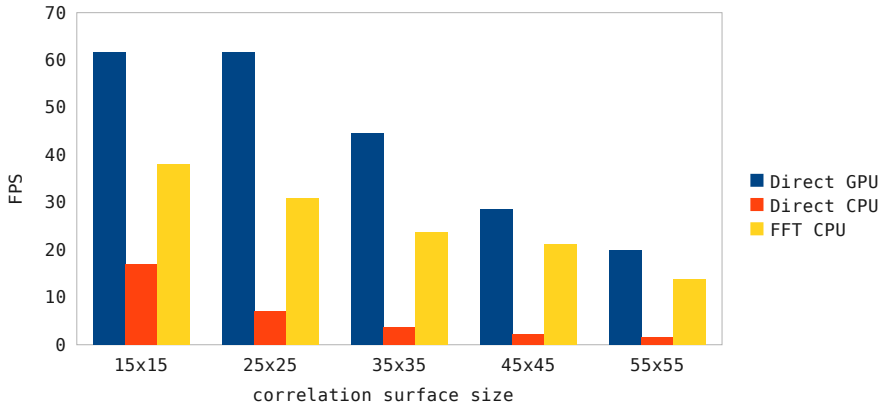

Figure 3.5: Performance of EPIV with different correlation surface sizes.

Figure 3.5 is similar to 3.4 but the size of the correlation surface is changed while the size of the reference window remains constant. As expected the performance impact of changing the size of the reference and correlation surfaces is almost identical. The number of iterations the inner most loop must complete is equal to the size of the reference window, but the number of times the inner most loop is executed is equal to the number of elements in the correlation surface. There are some small performance differences in

the FFT method because of different amounts of padding used. These differences are not, however, significant. Because these two graphs are almost identical, approximations may be made for combination of window sizes not tested. For example, the performance of a PIV configuration with a reference and correlation surface of size 45x45 would be similar to the performance of a PIV configuration with a reference window of size 35x35 and a correlation surface of size 55x55.

3.4.2 Sensor Count

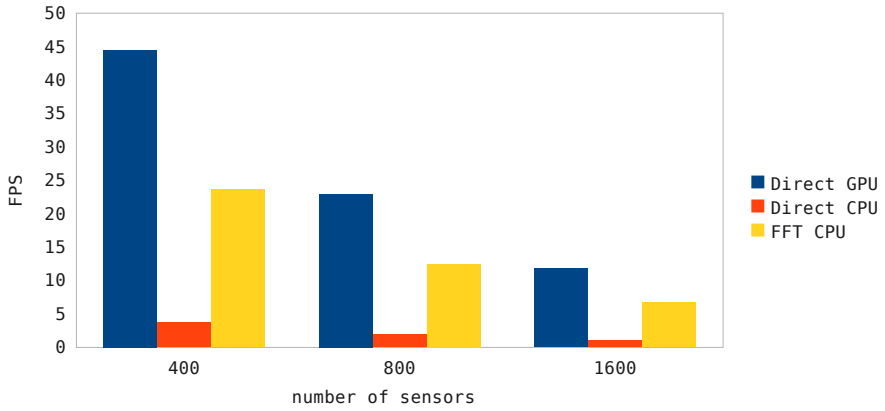

Figure 3.6: Performance of EPIV with different sensor density.

Figure 3.6 shows the performance of EPIV as the number of sensors are increased. A linear relationship between the number of sensors and FPS is apparent in this figure. If the number of sensors is doubled the FPS is roughly cut in half. At 400 sensors the FFT CPU and GPU methods are capable of processing in real time. Once the number of sensors is increased to 800, the GPU alone is still capable of processing at least 15 FPS. At 1600 sensors none of the methods are capable of processing at least 15 FPS. The GPU has additional overhead as the number of sensors increases because the results computed on the GPU have to be transferred back to the CPU. Figure 3.6 does not seem to indicate a scaling problem with the GPU so any overhead incurred by the memory transfer must be minimal.

3.4.3 Correlation Methods

Figure 3.7 shows the performance of the three different correlation methods. As expected the performance of the FFT CPU technique is about the same for all correlation methods. Results from the GPU also change very little for the different methods despite performing one additional operation in the inner loop for the correlation coefficient method.

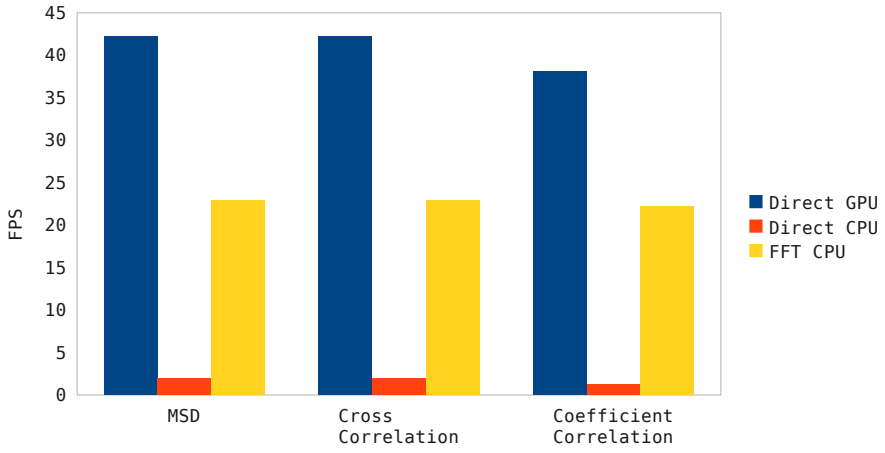

Figure 3.7: Performance of Different Correlation Methods.

Performance of the GPU is not suffering much from the additional operations because the hardware has some free arithmetic logic units. These units are able to process the additional operations while the GPU waits for memory accesses. Results from the direct CPU method show a significant difference in performance among the three methods. Additional operations required for the correlation coefficient method make it process at roughly two-thirds the performance of the cross correlation method. Performance of the direct CPU method could be improved by using the sum area table generated for the FFT method. Using a sum area table should bring the performance of the CPU direct method for correlation coefficient up to the performance level of cross correlation and minimum square difference.

3.4.4 Processor Count

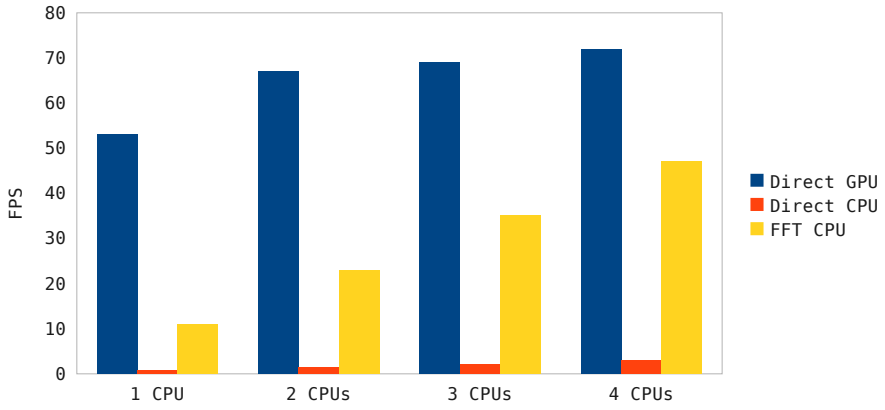

Figure 3.8: CPU Scaling.

EPIV CPU scalability is show in Figure 3.8. Both the FFT CPU method and the Direct CPU method appear to scale linearly with the number of CPU cores as expected. Note that the speed of the reader process does impose a hard limit to the scalability of EPIV. Performance of the direct GPU method also improves as additional cores are added. While the GPU itself does not change the ability of the CPU to feed the GPU data and wait on results is affected by the number of available CPU cores. With a single core processor a significant amount of time is spent reading in video and displaying output. The GPU may finish quickly but be forced to sit idle while additional frames are read in and processed. Notice that GPU performance only increases a small amount when the number of cores is increased to four. With three processor cores, the system has ample resources to read, write, and process results as fast as the GPU completes them.

The standard CUDA synchronization method involves a busy wait while the GPU is processing data. A spin lock is not ideal for systems with limited CPU resources. In EPIV the CUDA algorithm is configured to use a type of waiting mechanism called cudaDeviceBlockingSync which employs a synchronization primitive to wait of the device to finish. Using cudaDeviceBlockingSync could cause additional latency, but on systems with limited processing resources, cudaDeviceBlockingSync improves performance. EPIV performance increase is on the order of 20% when using cudaDiviceBlockingSync with one CPU and did not noticeably decrease the performance when running with the full four cores.

3.5 Real-Time Measurements

EPIV processing speed was also measured on a mobile computer platform designed to collect and process data in the field. The mobile platform was initially designed to collect video from two sensors and save it uncompressed to a hard drive. Later this video is processed and used for PIV measurements. Because of the significant size of the video the device cannot operate for long periods of time even with the largest storage devices available. To address the storage issue EPIV was extended to support reading video directly from firewire sensors using the dc1394 library. Because processed PIV results take up much less storage than video files, the mobile computer device is capable of much longer deployment.

The mobile computer system is based on the NVIDIA ION platform. The processor is an Intel Atom with two hyper-threaded cores running at 1.6GHz. The GPU is a NVIDIA ION stream processor with 16 cores and 256MB of memory. These and additional computer components were mounted into the custom computer case shown in Figure 3.9. Fedora 14 was installed as the operating system and EPIV was copied and recompiled on the portable system.

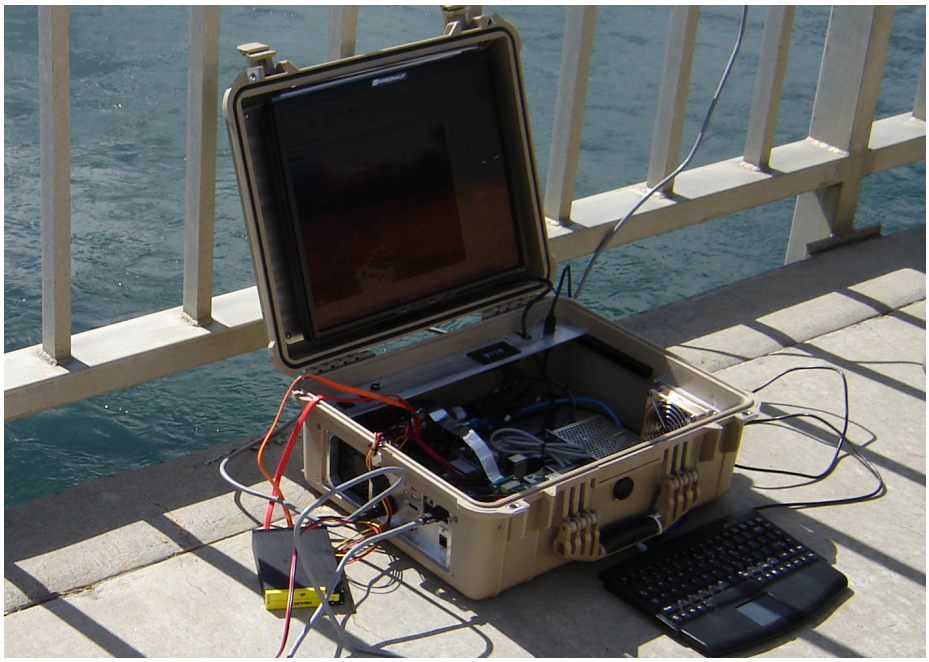

Figure 3.9: Target Mobile Platform for EPIV.

Additional speed measurements were made using a reference window size of 39x39 and a correlation surface size of 19x19. These measurements compare the real-time processing capability of the Atom system with the Xeon system. Because the firewire camera used in the experiments operate at 7.5 frames per second the mobile system must be capable of processing at least 7.5 frames per second. Processing speed was tested by increasing the number of sensors or measurement points until the processing rate fell below 7.5 frames per second. On the mobile Atom system when using a CPU or GPU based correlation up to about 64 measurement points may be computed in real time and on the Xeon system in excess of 4800 measurement points may be processed in real time when using GPU based correlation. When the size of the PIV windows are reduced by roughly half the Atom system using the ION GPU is capable of processing about 1500 measurements in real time while the Xeon system is capable of processing in excess of 47,000 measurements in real-time.

The low performance of the GPU on the Atom is caused by a number of factors. The ION GPU only includes 16 cores while the Tesla GPU include 240 cores. In addition, the ION GPU is a CUDA generation 1.1 device and contains less registers than the Tesla device. The lack of registers caused the correlation algorithm to only utilize 67% of the GPU. The ION architecture by itself is not very useful for real time PIV measurements but may be improved by adding a discrete CUDA GPU. With the addition of a discrete GPU to the ION platform should be capable of processing a much larger number of sensors in real-time.

3.6 Conclusion

The performance overall of the GPU correlation module is impressive. There are a number of reasons why direct correlation may be preferred over the FFT method [20] and the GPU module is capable of computing direct correlation at speeds that rival and exceed the FFT method for window sizes smaller than 45x45. Previously correlation methods, such as absolute difference, that do not have an FFT approximation have been ignored because of poor performance. The direct method is much more flexible and these methods could be implemented by changing only a few lines of code. The GPU algorithm could also be used for field projects where computational power is limited. Any system with a PCI Express bus and a sufficient power supply could easily be converted into a real-time PIV platform.

When running PIV measurements on a high end computer system, the CPU alone is capable of processing a large number of measurements in real time without the assistance of a GPU, but even on these systems the GPU is capable of improving the processing speed by 2-4x. On portable system that do not include high performance processors the addition of a GPU has the potential to increase the number of sensors that may be processed in real time by as much as 10-20x.

A new version of EPIV was developed using what was learned from the proceeding sections. The new version of EPIV defaults to the direct correlation method on the GPU unless the reference window exceeds 16,000 pixels. If a GPU does not exist or if the reference window size is too large, the CPU based FFT method is used to compute correlation. The GUI included with EPIV was also changed to remove many of the options determined to be ineffective. The new version of EPIV has a smaller memory footprint and is capable of computing real-time PIV measurements directly from a video sensor.

Chapter 4

CORRELATION CONFIDENCE

4.1 Introduction

PIV experiments typically use cross correlation to generate a velocity measurement for the area around a virtual sensor. Cross correlation methods are well researched and numerous techniques are available to improve both accuracy and processing speed [16, 4]. Cross correlation does have a number of problems including a sensitivity to noise and changes in uniformity. Some of these problems are difficult or slow to correct. For real-time PIV systems the best method to address possible correlation problems is often to simply not record velocity measurements where the results of cross correlation are in question. In this chapter a number of different measurement confidence scores are developed. These methods are designed to identify correlation surfaces that are not likely to produce an accurate measurement of velocity.

A correlation surface may be presented graphically using a number of different methods including height maps or as grayscale images. In a grayscale image representation of a cross correlation surface the higher the value of the pixel the higher the correlation score. A black pixel represents a low degree of correlation and a white pixel represents a high degree of correlation. An ideal correlation surface contains very few pixels with a high correlation (white) but a large number of pixels with a low degree of correlation (black). Figure 4.1 shows the correlation surface produced from a strong and unique correlation. All the correlation surfaces shown in this section are grayscale images generated by multiplying each correlation score as computed by the normalized correlation coefficient method by 255. All the correlation surface images were generated using the EPIV software discussed in Chapter Three.

The remainder of this chapter is divided into four sections. Section 4.2 presents a number of different common correlation problems and possible methods to correct them. Section 4.3 presents a number of different scoring methods and modifications used to identify correlation surfaces that may produce incorrect measurements. Section 4.4 discusses image preprocessing methods that have been useful in specific situations. Section 4.5 concludes the chapter with general observations about the filtering and preprocessing methods.

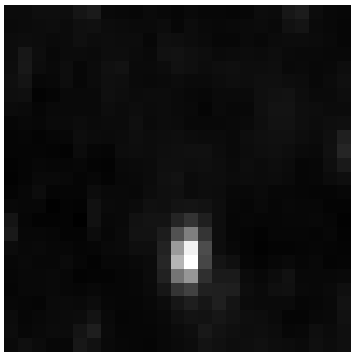

Figure 4.1: Correlation surface with a strong and unique peak.

4.2 Background

When used with video from field experiments, PIV often produces a large number of erroneous vectors. These erroneous vectors may be caused by a number of different problems but typically result in a correlation surface that contains no unique well defined peak, or a correlation surface that contains many peaks [9, 21]. Erroneous vectors may also be produced when fixed features such as reflections are present in the images. Fixed features often cause a line of high correlation scores along the fixed feature. In both of these cases the actual best peak may not produce an accurate velocity measurement. These and other causes of correlation failure are discussed in below.

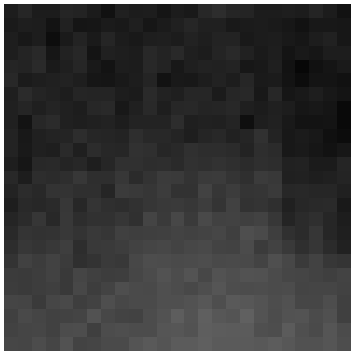

Figure 4.2: Correlation surface generated by introducing out of boundary motion.

Out of boundary particle motion occurs when features in the reference window at time t move too far to still be inside the search window at time $t + \delta t$. The correlation surface produced from out of bounds motion is often similar to Figure 4.2. Out of bounds motion may be eliminated by decreasing δt or by increasing the size of the search area. Decreasing δt is the preferred method to fix out of bounds motion because increasing the size of the search space also increases computation time and increases the probability of noise in the

correlation surface. A good method for setting the size of the correlation surface involves estimating the maximum amount of motion in each direction and using the formula

$$S_x = r_x p \delta t \tag{4.1}$$

$$S_y = r_y p \delta t \tag{4.2}$$

where S_x, S_y is the size of the correlation surface, r_x, r_y is the expected maximum velocity in the x and y direction, p is a scaling ratio to convert pixels space to world space, and δt is the change in time between image captures.

Out of plane motion may cause features to disappear resulting in a weak correlation peak. In 2D laboratory PIV experiments, a single plane is illuminated and only particles in that plane are visible. Particles that pass through the plane may appear in one frame and disappear in the next. Out of plane particles do not always cause correlation to fail. A small number of particles with out of plane motion may reduce the peak height but not to the point where noise begins to influence the result. Out of plane motion is reduced by increasing the size of the light sheet.

Sometimes particles within a reference window do not move in a uniform direction causing an in-plane velocity gradient [6]. A correlation failure may occur due to in-plane velocity gradients because the layout of particles in the reference window has changed in the search window. In-plane velocity gradients may be corrected for by reducing the size of the reference window or by a technique called window deformation [14]. The window deformation technique changes the shape of the reference window to correct for velocity gradients. The performance overhead required to perform window deformation operations would not be practical for a real-time system.

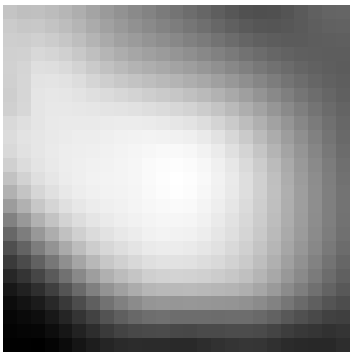

Figure 4.3: Correlation surface with no unique peak.

Peak locking is another cause of inaccuracies in PIV. Velocity is computed from the location of the peak within the correlation surface. Without any interpolation, the peak will

always be located exactly at a pixel position. The actual peak is almost never located exactly at a pixel coordinate but is instead somewhere between pixel locations. One method to address peak locking is to interpolate the location of the peak by using the correlation score of elements around the peak. Interpolation improves the accuracy of the measurements but does not completely solve the problem of peak locking [7]. Another method called multisampling is often used to reduce the error introduced from peak locking in addition to increasing spatial resolution. Multisampling computes correlation a number of times using different correlation windows sizes to improve the accuracy and spatial resolution of the peak location. Multisampling causes a significant performance loss and may not be appropriate for real time measurements.

4.3 Related Work

A number of correlation based and post processing based methods have been proposed to address erroneous vectors. One method uses a system to detect the number of particles and expands the reference window size until it contains a significant number of particles [26]. Another method combines a number of nearby correlation surfaces to remove noise [9]. Post processing methods that locate outlier measurements by looking at neighboring vectors have also been successfully used to improve PIV accuracy [13].

4.4 Correlation Confidence Scoring Methods

Many of the methods used to improve the accuracy of laboratory PIV experiments are not useful in riverine PIV because of low particle density. In PIV river experiments the most common cause of a correlation error is due to a lack of unique features in the reference window. The correlation surface shown in Figure 4.3 has many high peaks because the set of features in the reference window is not unique within the search window. Sometimes increasing the size of the reference window helps to produce a unique peak but at the cost of a decrease in spatial resolution and an increase in computation time. When practical, adding additional particles to the flow is helpful, but even with additional seed particles, correlation will often fail to produce accurate results at some virtual sensor locations. A number correlation confidence scoring methods are developed in this section. These methods could be used to filter out vectors generated from possibly inaccurate correlation surfaces.

EPIV includes a number of methods for scoring the quality of the correlation surface. The most simple method uses the height of the peak. A high peak indicates that there is a high degree of correlation. A low peak height indicates that there could be a problem with the correlation surface that may include out of boundary motion, out of plane motion, or a

velocity gradient. The peak height method is not useful for identifying a correlation surface where all the scores are very high. Such surfaces are common in PIV computations where the number of particles in the fluid is limited.

Another method used in the Matlab PIV software called MPIV [18] and also implemented in EPIV uses a peak uniqueness measurement to score a correlation surface. The peak uniqueness method is referred to here as uscore. Uscore is based on the noise floor of the correlation surface and how far above this the peak is. First the highest correlation score is located by finding the maximum correlation in the correlation surface using a linear sub-pixel interpolation method. Next the mean and standard deviation of the entire correlation surface are computed. These calculations are combined using

$$uscore = \frac{max(c) - mean(c)}{stdev(c)} \tag{4.3}$$

where *max* is the interpolated peak, *mean* is the mean of the correlation surface, $stdev$ is the standard deviation of the correlation surface, and c is the correlation surface. The uniqueness score identifies a correlation surface produced when there is little or no particles in the lag and lead images.

A high uscore does not indicate a high degree of correlation. Under some circumstances a correlation surface may generate a very high uscore but contain a very weak correlation peak. Uscore may be modified by combining it with the peak correlation score. When the NCC method is used to compute the correlation surface the uniqueness score may be weighted by the peak correlation score using the formula

$$wscore = \frac{max(c) - mean(c)}{stdev(c)} * max(c) \tag{4.4}$$

where c is the correlation surface. A good cutoff for wscore is typically in the range of $[2,3]$ but may be different depending on the application. Vectors produced from correlation surfaces with scores lower than the cutoff could be discarded to remove possibly incorrect measurements.

An additional modification of the wscore method was also found to be effective at identifying inaccurate vectors generated from high intensity surfaces within the lag and lead images. The modified method uses the wscore and computes an additional measurement similar to the wscore but with the mean value replaced by standard deviation. The final score of the correlation surface is taken as the minimum of the two scores. The code for this method is given below.

$$mean\_score = \frac{max(c) - mean(c)}{mean(c)} * max(c) \tag{4.5}$$

$$confidence\_score = min(wscore, mean\_score) \tag{4.6}$$

A final modification to the wscore method adds an additional power component to the difference between the $max(c)$ and $mean(c)$ components. The additional parameter may be used to increase or decrease the correlation score depending on the level of contrast within the correlation surface. By increasing the power of the difference between the peak and the surface mean, correlation surfaces that include high contrast are given a higher score. The power of the difference may also be decreased to decrease the score of correlation surfaces with high contrast. If an image contains a number of different zones with different levels of contrast, this method may be used to eliminate measurements for undesired zones. The formula for the weighted difference score is given below.

$$dscore = \frac{(max(c) - mean(c))^n}{stdev(c)} * max(c) \tag{4.7}$$

A filtering method based on a maximum velocity cutoff score was also useful in some experiments. If a correlation surface produces a measurement that is known to be too large it can be considered inaccurate and discarded. A simple modification of the maximum velocity filter was found to be useful. If a correlation peak is determined to be located along the edge of the correlation surface it is considered to be inaccurate. A correlation along the edge may indicate that the actual peak is further from the edge as is the case with out of bounds motion and therefore the velocity measurement is not correct.

In experiments where the flow is in a semi steady state a number of vector fields may be averaged together to produce one velocity field representing a longer span of time. Taking the time average will help to reduce the influence of noisy vectors but may also lead to lower overall velocity measurements. If vectors considered to be erroneous are removed using one of the methods above, taking the time average will help to produce a more complete vector field without reducing the magnitude of velocity measurements.

The methods discussed above were used to determine how many incorrect vectors could be effectively identified and removed. Approximately 143 virtual sensors were placed in a specific part of a video where the direction of motion is known to be towards the south. The images used for this measurement were extracted from the first 200 frames of a video captured of the Wolf River on May 26, 2010 starting about 10:00AM. Averaging each sensor over the 200 frames resulted in 127.634 or 89% of the sensors producing a measurement in the south direction and 10.26 or 7.6% of the sensors producing a measurement in the

north direction. Enabling the maximum velocity filter and setting the wscore filter to 2.0 caused the number of south vectors to decrease to 101.317 and the number of north vectors to decrease to 0.52. Setting the wscore filter to 3.0 caused the number of south vectors to decrease to 91.0 and the number of north vectors to decrease to 0.322. Assuming the number of incorrect vectors in the south direction is equal to the number of incorrect vectors in the north direction, an estimate of the total number of incorrect vectors removed when wscore is set to 2.0 is $2 * (10.26 - 0.52)$ or 19.48, and an estimate of the total number of correct vectors removed is $127.634 - 101.317 - 19.48$ or 6.837.

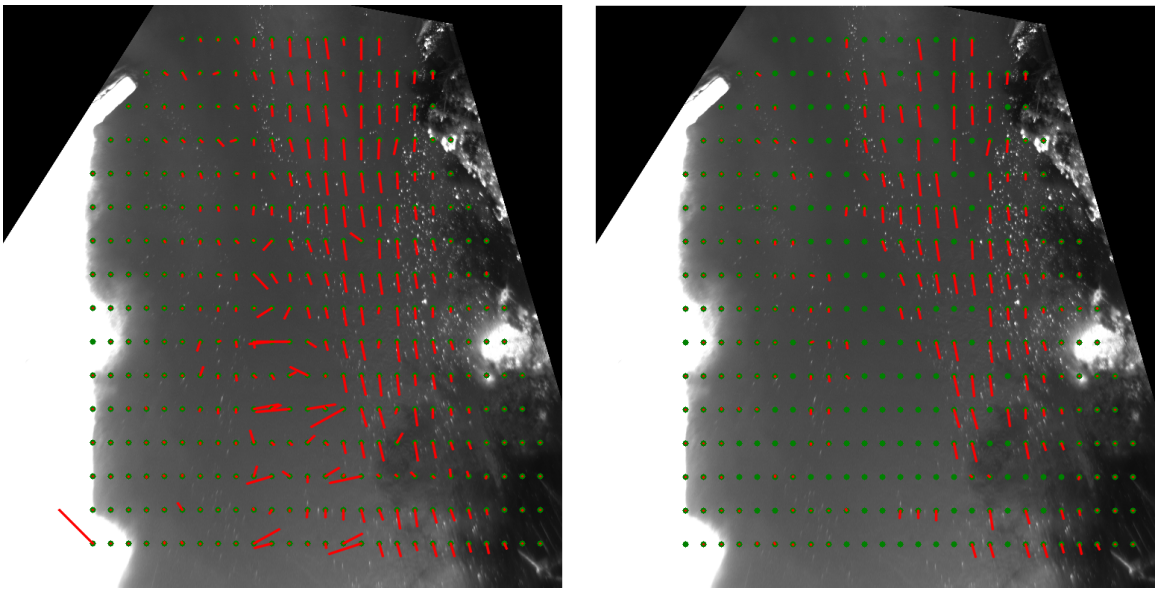

Figure 4.4: Filtering by wscore Disabled (left) and Enabled (right).

4.5 Image Processing Methods

In some experiments image processing helped to improve the quality of the measurements generate by PIV. One experiment contained video that included strong shadows in the crossflow direction. The shadows caused the correlation surface to report a measurement close to zero when there was clearly motion visible in the video. The methods discussed below helped to produce accurate velocity measurements even with the shadows present in the video. These methods also helped to correct measurements around other strong stationary features such as fence posts.

The strength of the shadow or stationary feature within an image causes high correlation scores in a peak range parallel to the feature. An image processing method was developed to remove the common elements in the lag image and correlate the results. The processing

method takes the lag frame and subtracts the lead frame from it.

$$lag[x][y] = lag[x][y] - lead[x][y] \tag{4.8}$$

Any part of the image that is the same in the lag and lead frames is removed and only the parts that are unique to each image remain. The new image generated from the difference between the lag and lead images is used provide intensity information for the reference window. Normalized cross correlation is performed on the the difference image and the original lead image. Because the strong stationary shadows and features are removed the resulting correlation measurements are more accurate as long as there is sufficient motion in the images. In sections without motion the modified lag image does not contain texture and therefore cannot be used to produce correlation scores.

A modified version of the method discussed above was developed and may be more accurate in specific experiments and especially in zones with small or no motion. The modified methods takes the difference between the lag frame and half the lead frame using the expression below

$$lag[x][y] = lag[x][y] - lead[x][y]/2 \tag{4.9}$$

$$lead[x][y] = lead[x][y] - lag[x][y]/2 \tag{4.10}$$

Removing half of the lead image from the lag images effectively reduces the intensity of pixels common in both images. Zones with zero motion are correctly detected with this method because intensity measurements common to both frames are reduced but not eliminated. The same method may be applied to the lead image as shown in the equation above. Additionally the value 2 could be increased or decreased to preserve more or less of the common intensity measurements.

A third frame (called scout in the equations below) captured at double the δt of the lead and lag frames may be used in place of subtracting the lag frame.

$$lag[x][y] = lag[x][y] - lead[x][y]/2 \tag{4.11}$$

$$lead[x][y] = lead[x][y] - scout[x][y]/2 \tag{4.12}$$

Subtracting the lag frame from the lead frame may cause high correlation scores in the opposite direction of the motion and may in some cases cause incorrect measurements. If the motion is mostly steady taking a frame at a point in time after the lead frame may improve the quality of the measurements.

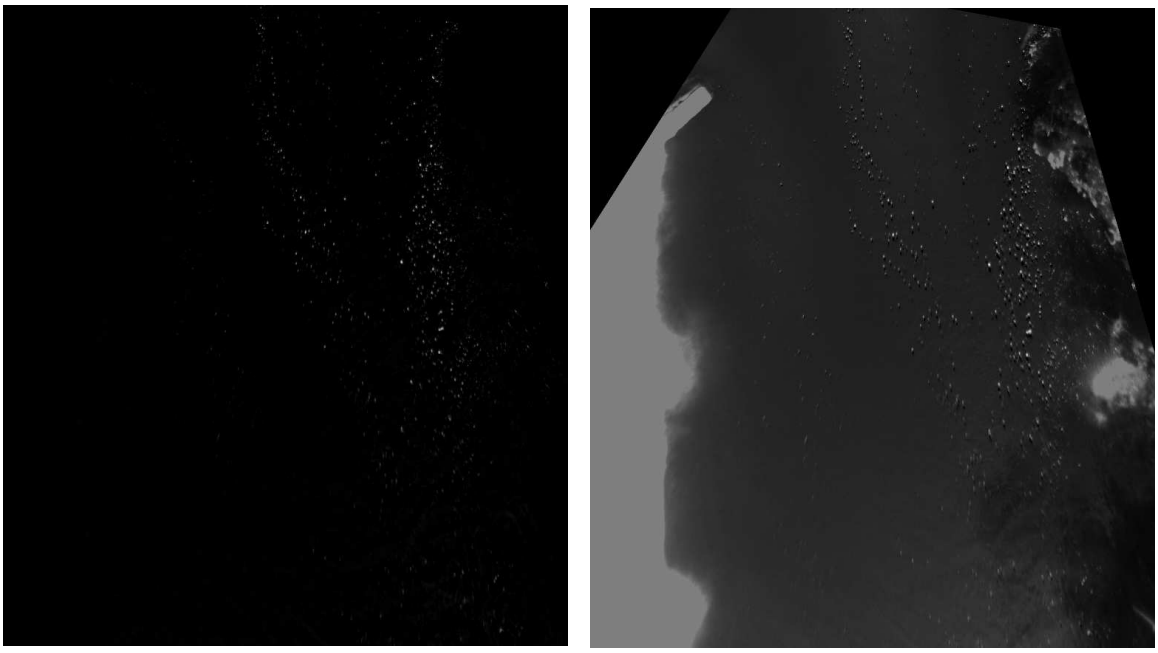

Figure 4.5: Difference (left) and Weighted Difference (right) Methods on the Image from Figure 4.4

4.6 Conclusion

Assigning a level of confidence to a correlation surface helped to improve both the accuracy of single and time average PIV measurements. The correlation confidence scoring methods increased accuracy of standard PIV computations by successfully identifying and removing most of the measurements that were know to be incorrect. Time average PIV calculations were improved by the confidence score because random vectors generated by noise and other factors that appear in each individual PIV calculation were rejected and not included in the average velocity measurements.

There are a few outstanding issues with the filtering methods discussed in this section. In some cases velocity measurements that are accurate have a low confidence score and are removed by the filter. In other cases a measurement that is clearly inaccurate is given a high correlation score. One situation occurs when the correlation surface has a small number of well defined peaks. Scoring methods based on the difference between the peak and the mean will give this a high measurement, but the peak may not be correct because there are a number of close to equal alternatives that could also be correct. In addition to not always being accurate these filtering methods require a large amount of manual adjustment and trial and error to determine the best method and parameters.

Image processing techniques are effective at removing shadows and other objects com-

mon to both the lag and lead frames. In videos where a shadow or other object is causing problems for PIV these image processing m ethods have effectively turned inaccurate measurements into accurate measurements. There are some problems with the image processing methods. In images where there are no common objects that interfere with correlation better results are usually obtained with unmodified images. Removing the stationary objects also causes the PIV measurements to be incapable of generating a high correlation with zero motion.

Chapter 5

FIELD EXPERIMENTS

5.1 Introduction

Three experiments are discussed in this chapter and referred to respectively as Cable Bridge Experiment, Hotel Experiment, and NPS Beach Experiment. The purpose of these field experiments was to test methods for video collection and to measure the accuracy and capability of the EPIV system discussed in Chapter Three. These results will be used to test if EPIV is capable of taking accurate field measurements in real time. The remainder of this chapter is divided into four sections. Section 5.2 titled "Related Work" reviews other work that uses LSPIV or PIV to measure velocity using images collected in the field. Section 5.3 titled "Equipment" describes the equipment and methods used for the experiments. Section 5.4 gives the results for the experiment conducted at Cable Bridge. Section 5.5 gives the results for the experiment conducted at the Hotel location. Finally section 5.6 gives the results of the experiment conducted at NPS Beach.

5.2 Related Work

PIV methods have been successfully used in a number of field experiments related to ocean flow measurements. One example is a system designed to calculate near shore ocean currents [12]. Video is rectified to the ground plane using a photogrammetric camera model similar to 2.11 and 2.12. Correlation is measured using a minimum difference scoring method similar to MAD. Noisy vectors are detected using a post processing method that takes the mean of eight neighbor vectors to look for outliers [21]. If an outlier vector is detected it is replaced with a weighted mean value computed from the neighbor vectors. PIV results were found to be consistent with current meters and different flow measurement methods. PIV methods have also been applied to underwater ocean measurements. A method of using laser illumination and standard PIV techniques is used to measure turbulent flows in the costal bottom boundary layer of the ocean [23].

PIV methods have also been applied to measuring river surface flow. One example is a system designed to measure river surface velocity using a video sensor with no artificial lighting [11]. The system consists of a truck mounted camera set up for collecting images

and a laptop PIV program to process the results. The rectification method uses a plane to plane transformation technique [17]. Correlation is measured using a traditional CCOEFF method. The flows are considered to be in a steady state so a time average is used to improve the velocity measurements. Multiple partially overlapping images are combined to increase the amount of area that may be measured by the PIV system. Results from the PIV measurements are said to agree with other reference measurements. Measurements are only computed once every two minutes.

Methods to seed large surface flows have been developed and successfully used in a number of experiments. One such PIV experiment takes measurements of a large surface flow using many of the techniques common in small laboratory experiments [29]. The technique employs many of the tools used in a lab PIV experiment including seed particles and constant lighting. A correlation filtering method called the peak ratio factor is used to reject vectors that are likely incorrect. The method uses the difference between the highest and lowest peak divided by the difference between the second highest and lowest peak. Laser-Doppler-Velocimetry measurements were taken to verify the vectors generated by PIV.

Similar work using infra red video devices and PIV is currently being conducted by the COHSTREX group [15]. The COHSTREX group uses a barge equipped with a thermal camera and a number of different sensors to study river flow structures. A thermal camera mounted to an aircraft is also used to collect images of the river. Results from this group show that coherent thermal structures do exist and that PIV may be used to measure these structures.

5.3 Equipment

Equipment used in these experiments included a Dell Precision T7500 computer, a Point Grey Research Grasshopper GRAS-14S5M-C near infrared sensor, a Point Grey Research Grasshopper GRAS-20S4C-C color sensor, and a FLIR SC6700 thermal sensor. The near infrared sensor was equipped with a 9mm Fujinon HF9HA-1 lens and a 9mm Opteka HD2 infrared filter. The color sensor was also equipped with a 9mm Fujinon HF9HA-1 lens. Both the color and near infrared devices were equipped with a polarizing filter. The thermal sensor was equipped with a 25mm lens. Camera settings including shutter, exposure, and gain were adjusted manually to improve the quality of surface features recorded to video. The video captured from these devices was later processed using the Dell Precision T7500 computer and an NVIDIA Tesla C1060 GPU.

To capture video in the field, a mobile computer system was designed and developed. The

mobile computer system includes an Intel Atom processor and an on-board NVIDIA GPU. Video sensors were connected to the mobile computer using FireWire. For experiments where the video sensors must be placed a significant distance from the computer a FireWire to Cat 5 converter was used. As video was collected it was stored directly to the hard drive of the mobile device as uncompressed frames. Due to the size of the video files a number of high capacity hard drives were connected to the mobile computer. These video files were later compressed with H.264, stored to a file server, and processed by EPIV.

Table 5.1: *Camera and PIV Experiment Parameters*.

	Cable Bridge (26)	Hotel	NPS Beach
Angular FOV	57.3	22.3	42.2
Azimuth	-173.6	15.5	28.8
Tilt	55.8	76.8	81.8
Roll	-3.6	0.5	-1.5
North	3374571.222	5393997.769	5394502.469
East	281669.1227	550995.343	555600.492
Elevation	16.248	545.784	543.982
Reference Window	1.2x1.2m	3.5x3.5m	3.5x3.5m
Correlation Surface	0.6x0.6m	3.0x2.5m	2.75x2.0m
Correlation Method	CCOEFF	CCOEFF	CCOEFF
Pixel Resolution	0.030 m/p	0.10 m/p	0.10 m/p
Averaging Time	800 frames	800 frames	800 frames

All PIV measurements were computed using EPIV. EPIV requires input that includes a video file, camera position parameters, camera orientation parameters, control points, image registration area, reference window, correlation surface size, sensor location, and filter settings. The position of the camera was determined by field survey equipment. Camera orientation was computed by manually adjusting the orientation parameters until the points correctly line up with associated image features. The image registration area was determined by using satellite images to get a rough estimate of the ground location and then fine tuning the results by manually adjusting the registration parameters. Reference window size was initially set to a low starting value and increased until the visual observation of the surface flow stopped improving. Correlation surface size was initially set to a high value and then reduced until vector speed began to be affected. The location of sensors were configured using a uniform grid that typically included enough space between sensors so that vectors do not overlap on the display. Filter settings were adjusted manually by observing the number of vectors that appear to be inaccurate and increasing the filter settings until these were

removed. Final setting used in the three experiments are given in Table 5.1

5.4 Cable Bridge Experiment

Cable Bridge Experiment was conducted on the Wolf River in Mississippi. The Wolf River is a small river located in South West Mississippi that originates in southern Lamar County and flows in a south to southeast direction before emptying into the Bay of St. Louis. Both the depth and discharge of the Wolf River very widely depending on rainfall conditions. Video of the river flow was taken at a location just south of Cable Bridge Road near Landon Mississippi.

Figure 5.1: Wolf River Experiment Location.

A Point Grey Research Grasshopper NIR camera equipped with a 7.5mm lens was positioned on the bridge crossing the Wolf River. The approximate location of the camera is given in UTM zone 16R coordinates in Table 5.1. The camera was secured to the bridge using a camera mount and ratchet straps. A portable Intel Atom based system located off the bridge was connected to the cameras using a FireWire to Cat 5 repeater. Video was collected at various times during the morning and afternoon on May 24, 25, and 26 of 2010 and stored by the Atom system as uncompressed AVI files. The camera equipment was removed at night and repositioned the next day according to markings made on the bridge.

Processing was not done on the Atom system itself. Instead the video files were copied to a file server and later processed by the system described in Chapter 3. Configurations and Results are discussed in four subsections: Image Registration, PIV configuration, Results, and Validation.

5.4.1 Image Capture and Registration

Images of the Wolf River were captured by a NIR video camera. The NIR device was configured to collect one image every 7.5 seconds and store these images to an uncompressed video stream on the hard drive of a nearby computer. The NIR camera was equipped with a polarizer that was manually adjusted to remove some of the glare visible on the water surface. Other external camera settings on the NIR device such as aperture and focus were manually adjusted to maximize the number of features visible on the water surface. Internal camera settings including shutter speed were set manually using camera control software.

A number of different objects were surveyed and used as control points for image registration. The position of these control points in UTM were entered into the EPIV camera configuration module. EPIV displays these world coordinate points as green dots in the image space. Camera orientation numbers were adjusted until the green dots representing the world coordinates of the control points matched closely with the position in the image of the objects used as control points. Camera orientation parameters are given in Table 5.1 and the world control points projected to the image in Figure 5.2.

The registered image area used for the Wolf River experiment ranged from UTM 281656 E, 3374538 N to 281680 E, 3374564 N or approximately 650 square meters. Each pixel in the remapped image represented approximately one-thirtieth of a meter for a total resolution of 750x780 pixels or 25x26 meters. A single registered frame from video taken on May 25 and May 26 is shown in Figure 5.3. The same registration area that was used for the May 26 video was also used for the May 25 video.

5.4.2 PIV Configuration

For video captured on Aug. 26, PIV was configured to use a reference window of size 1.2x1.2m and a correlation surface of size 0.6x0.6m. These parameters were chosen mostly by trial and error. In general the size of the reference window should be larger than the correlation surface and the correlation surface should be large enough to find the maximum expected amount of motion. The camera was set to capture 7.5 frames every second and the image reader in EPIV was set to an offset of two frames making the effective frame rate of the video for PIV 3.75 fps.

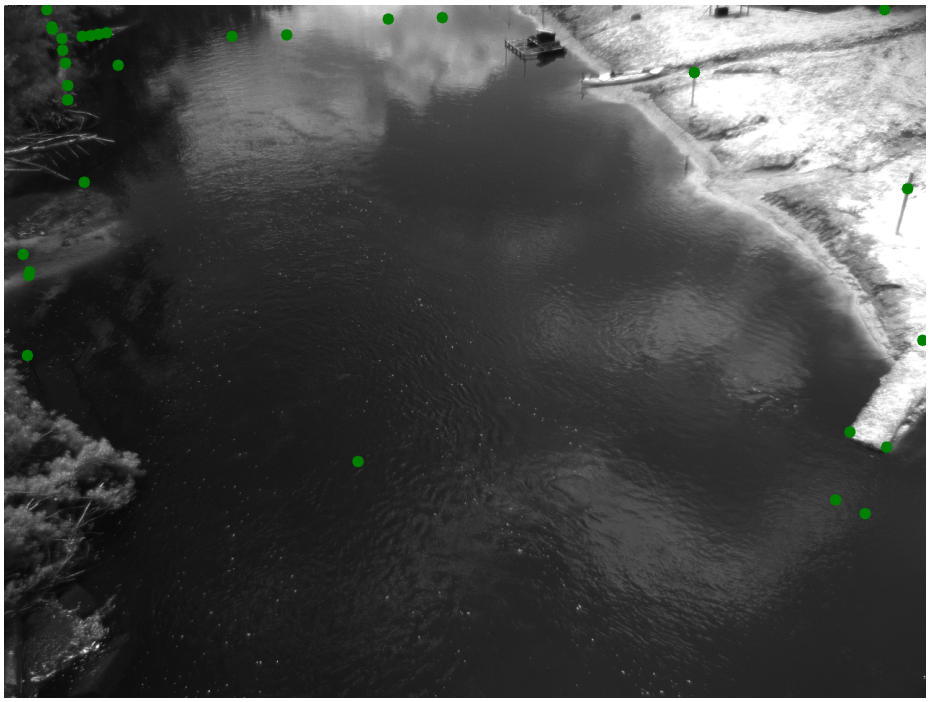

Figure 5.2: Cable Bridge Control Points.

With a correlation array of 0.6x0.6m or 21x21 pixels (windows sizes are rounded to the nearest odd integer), the maximum motion detectable in the x and y directions would be $floor(21/2) * 0.03 * 3.75$ or 1.125 m/s. The offset between the images used in EPIV is typically set by starting out at one and increasing the value until jitter motion in the vectors begins to stabilize.

Sensors were configured in a uniform grid arrangement with one sensor placed every 1.5 meters in the north to south direction and every 0.8 meters in the east to west direction. Some sensors in this grid fall outside the imaged area (black pixels in Figure 5.3) and others are too close to the edge of the image. These invalid sensors are automatically eliminated by EPIV. After elimination, a total of 368 valid sensor locations were detected. The position of these sensors is shown as green dots in Figure 5.3.

5.4.3 Results

Video of the Wolf River was collected on May 24, 25, and 26. Video collected on May 24 was not processed due to a problem with the camera settings that caused the recorded video to not have a constant frame rate. Lighting was problematic in these measurements due to shadows from trees and clouds. Glare from the water surface was also an issue for video collected during the midday hours. Video collected during the morning hours tended to

produce the best PIV results. Particle distribution was also not ideal in these experiments. Parts of the water surface contain little or no particles to track, and some of these particles are not always stable and sometimes merge or disappear. Even with these issues good time average measurements were obtained on May 25 and 26.

Results from a time average of 800 frames collected in the morning of May 25 are given in Figure 5.3. Time average measurements rely on the wscore technique to eliminate inaccurate vectors. In these measurements the minimum acceptable wscore is set to 2.5. Any vector that scores less than 2.5 are not included in the time average results. No vector is reported for a sensor if it does not produce at least 20 vectors with wscore greater than 2.5. Each of the sensors with a red line extending from it produced at least 20 vectors that scored at least 2.5 or better during the 800 frames processed. Results from May 25 show a maximum surface velocity of about 60cm/s in the center east region of the river.

Results from computing the time average PIV of 800 frames collected in the morning of May 26 are given in Figure 5.3. A maximum velocity of about 90cm/s is reported. The increase in velocity from May 25 to May 26 is likely due to rainfall during the afternoon hours of May 25 that increased the water level in the river by at least 10cm. Notice that particle density and lighting were better on May 26 which resulted in more sensors reporting a velocity measurement.

Processing 800 frames of video collected at the Wolf River location took approximately 40 seconds or 20 frames per second on a single core system with a GPU processing correlation. Real time processing would have been possible with this experiment because the video collection rate was set to 7.5 frames per second. The source video was not compressed so no significant overhead from video decoding is introduced in the input module. The system used for these real-time tests is the benchmark system described in Chapter 3 with three of the processor cores disabled. Disabling all but one processor core simulates the performance that could be expected on a low end portable computer.

5.4.4 Analysis

Velocity measurements were recorded by a mobile surface ADCP device called a RiverRay. A number of transects were made using the RiverRay two of which overlap the area imaged by the camera. The RiverRay device uses a GPS and ADCP unit to sample and record water velocity and position. Positions sampled by the RiverRay in the two overlapping transects were used as sensor positions in EPIV. PIV measurements were computed for each of these sensors and the results were compared to the RiverRay measurements of the water current 25 cm below the water surface (Figure 5.4).

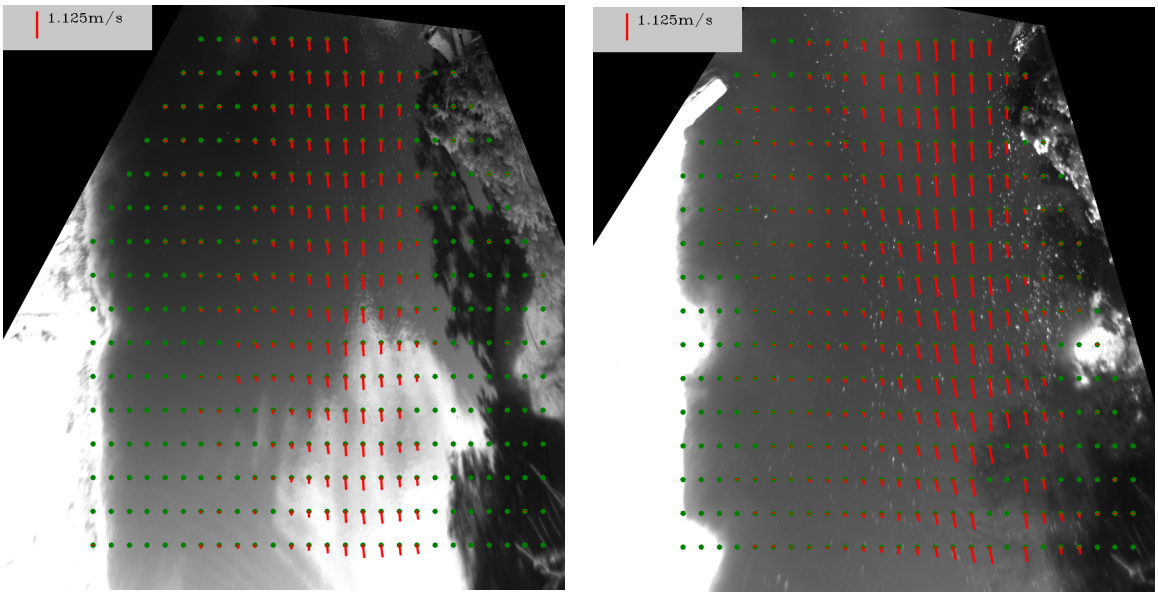

Figure 5.3: Cable Bridge PIV Results For May 25 (left) and May 26 (right).

Average difference between the vectors produced by these two devices is 0.157 for the first transect and 0.088 for the second transect. Inside the main channel the RiverRay produces a lower speed reading than does PIV. The difference in speed measurements in the main channel could be due to the difference in recording time. The RiverRay measurements and PIV video were collected about two hours apart. Due to thunderstorms on May 25 and May 26 the condition of the river was likely to change somewhat during the time-span between experiments. The difference between these measurements could also be caused by sampling depth. PIV only support surface current sampling and the RiverRay only support subsurface sampling at a minimum depth of 25 cm.

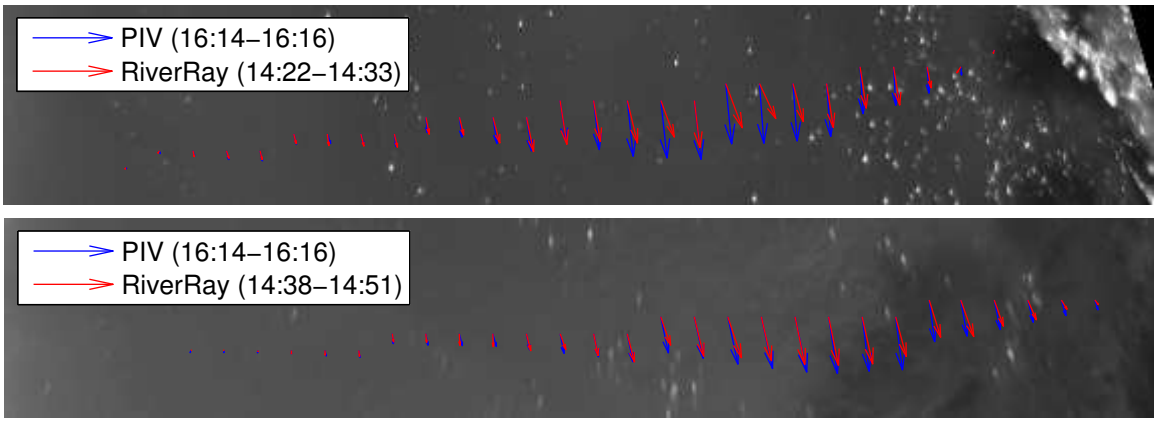

Figure 5.4: Cable Bridge PIV and RiverRay Vectors.

5.5 Kootenai Hotel Experiment

Hotel Experiment was based at the Kootenai Best Western Hotel that overlooks a section of the Kootenai River just east of the US Highway 95 bridge in Bonner's Ferry, Idaho. The Kootenai river is a major North American River that originates in British Columbia and flows south into Montana. In Montana, the river flow changes to a northwestern direction and continues through Boundary County, Idaho and into British Columbia. The width of the Kootenai River at the Hotel Experiment location is in excess of 200 meters and only about 100 meters of the river starting from the south bank is recorded onto video.

Cameras were positioned at the Kootenai Best Western Hotel in rooms 401 and 405. Exact UTM coordinates for the camera location are given in Table 5.1. Video was collected starting on Aug. 8 and ended on Aug. 18. Most recordings were set to collect 2 minutes of video and stop. Additional longer recordings of 5-20 minutes were also collected. The Kootenai River experiment used the same NIR camera used in the Wolf River experiment in addition to a Point Gray Research Grasshopper color camera and a FLIR SC6700 thermal camera. The NIR and color cameras were both equipped with a 9mm lens and polarizer filters, and the FLIR camera was equipped with a 25mm lens. The NIR camera was also equipped with a Near IR filter.

PIV results are similar on video collected from the different devices, and so only the results from one device are shown at each location. Each of the devices has some advantages and disadvantages. More surface texture is visible in the images collected by the FLIR but the field of view and resolution is more limited. Images from the color and NIR devices have to be collected during a time of the day when lighting is good. The FLIR device is much less sensitive to changes in lighting and may collect images at night or during the day. Results for the hotel experiment were computed using video from the FLIR device. The sequence of video used to generate the PIV results was collected on Aug. 16. Other video files from different times and different days all produce similar results indicating that the condition of the river did not change much for the duration of the experiment.

The Hotel location is a section of river with a mostly uniform flow direction concentrated in a main channel. The main channel is located between the southern bank of the river and the sand bar seen in Figure 5.5. One non-uniform flow pattern is visible at the eastern end of the central sand bar. From the edge of the sand bar, water appears to flow in a southerly direction into the main channel where it changes to an easterly direction. No significant concentration of surface particles are visible in this section of the river, but surface disturbances in the water are numerous in the main channel.

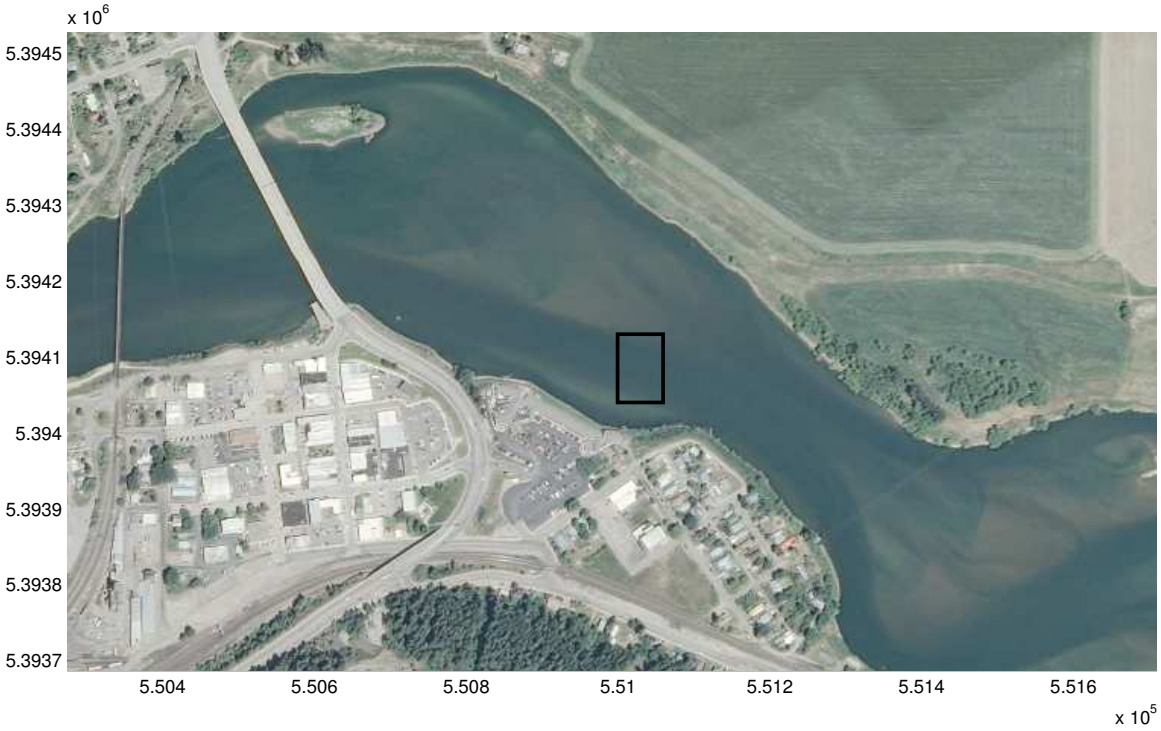

Figure 5.5: Hotel Experiment Location.

These surface disturbances seem to mirror the direction and speed of the water flow and are stable enough to use as features for PIV.

Lighting is problematic for the color and NIR devices at the Hotel location. The Hotel casts a shadow over the river during the morning hours that makes surface features in the water nearly invisible. During the evening hours glare from the sun is intense, obscuring many of the surface features. The best video recorded on the color and NIR devices was captured during the hours when the sun was mostly overhead. Lighting does not seem to be much of an issue for the FLIR device. FLIR video seems to record surface features during almost all hours of the day and night.

Both the NIR and color devices record underwater features such as sand and debris along the southern edge of the river. Where these underwater features are visible PIV tends to report an incorrect velocity of zero. After registering the image, the amount of river where underwater objects are visible is small but may still cause a number of vectors in the lower part of the image to report zero velocity. Images from the FLIR device do not show the underwater objects.

Overall the FLIR device did a better job at recording surface features and was chosen as the source of video for PIV at the hotel location. While the difference in velocity measurements between the FLIR, color, and NIR were limited, the FLIR does produce a

more complete vector field. The choice to use the FLIR device to produce results for the hotel location is primarily due to the underwater features present in the color and NIR video.

5.5.1 Image Registration

Figure 5.6: Control Points at Hotel.

Image collection using the FLIR device is a little different from collection using the NIR camera. The FLIR device works with a special software packages that controls the camera parameters. All these settings were initially left to the default values except for the capture rate which was set to 6.0 Hz, 7.5 Hz, or 16.0 Hz depending on the video. The frame rate setting was recorded in the file name and entered into EPIV. Some filter settings were adjusted near the end of the experiment but these adjustments did not appear to improve the results any.

Data collected by the FLIR camera is stored in the sfmov file using 14 bits of information for each pixel. Each pixel must be converted to an 8 bit integer because the PIV and display algorithms do not directly support 14 bit pixels. Currently the 14 bit number is converted into 8 bits by dropping the 6 least significant bits. As an alternative to dropping the 6 bits of information a manual range may be specified that maps all values in that range to the avaliable 8 bits. An alternative method is under development that attempts to automatically

determine a significant intensity range and map these values to 8 bits. The alternative method would preserve addition information in some of the bits dropped by the current conversion method.

Control features in the form of fence posts were used to determine the camera orientation parameters and to register the images collected from the FLIR camera at the hotel location. Fence posts were driven into a sand bar near the center of the river. Survey equipment was used to determine the UTM position of each fence post. Camera orientation parameters were adjusted manually until the projected position of the world coordinates of the fence posts matched the image position of the fence posts. The final alignment used for the hotel location experiment is shown in Figure 5.6, and the actual measurements are given in Table 5.1.

The registered image area for the hotel experiment ranged from UTM 551000 E, 5394040 N to 551060 E, 5394130 N, or a total of 5400 square meters. All points in the registered image are mapped to the elevation of the water that was measured at 531.824 meters. Each pixel in the registered image represents 8 meters for a total resolution of 480x720 pixels or 60x90 meters. Figure 5.7 shows the result of registering the video captured by the FLIR camera at the hotel location.

5.5.2 PIV Configuration

PIV was configured using the parameters given in Table 5.1. The correlation surface is set to 3.0x2.5 meters or 31x25 pixels. Video from the FLIR device was recorded at a rate of 16 frames per second and δt for the PIV computation was 11 frames or 0.6875 seconds. The maximum velocity that could be detected using these settings is 2.182 meters in the east and west direction and 1.75 meters in the north and south directions. The reference window is set to 35x35 pixels or 4.375x4.375 meters. Typically the window sizes are set by first determining the maximum amount of motion in the east and north directions and choosing a correlation surface size large enough to detect that amount of motion. The reference window is then set to be larger than the correlation surface.

The frame offset of 11 was determined by trial and error. With lower frame offsets the vector direction tends to oscillate a small amount to the north and south. As the offset is increased these oscillation disappear. At really small frame offsets, the oscillation is likely due to the sub-pixel interpolation. Train traffic near the hotel was heavy and causes visible motion in some of the video files which could be a source of some of the oscillation. Increasing the offset also eliminated some of the erroneous vectors.

Sensors were configured in a uniform grid pattern starting at 551000 E and 5394040

N UTM with a spacing of two meters in the north and east directions. Grid spacing is typically configured so that vectors do not overlap. In this experiment the maximum amount of motion in the east and north directions is just over two meters so the spacing between sensors is configured to be two meters. The total number of sensors after eliminating invalid locations were 594.

5.5.3 Results

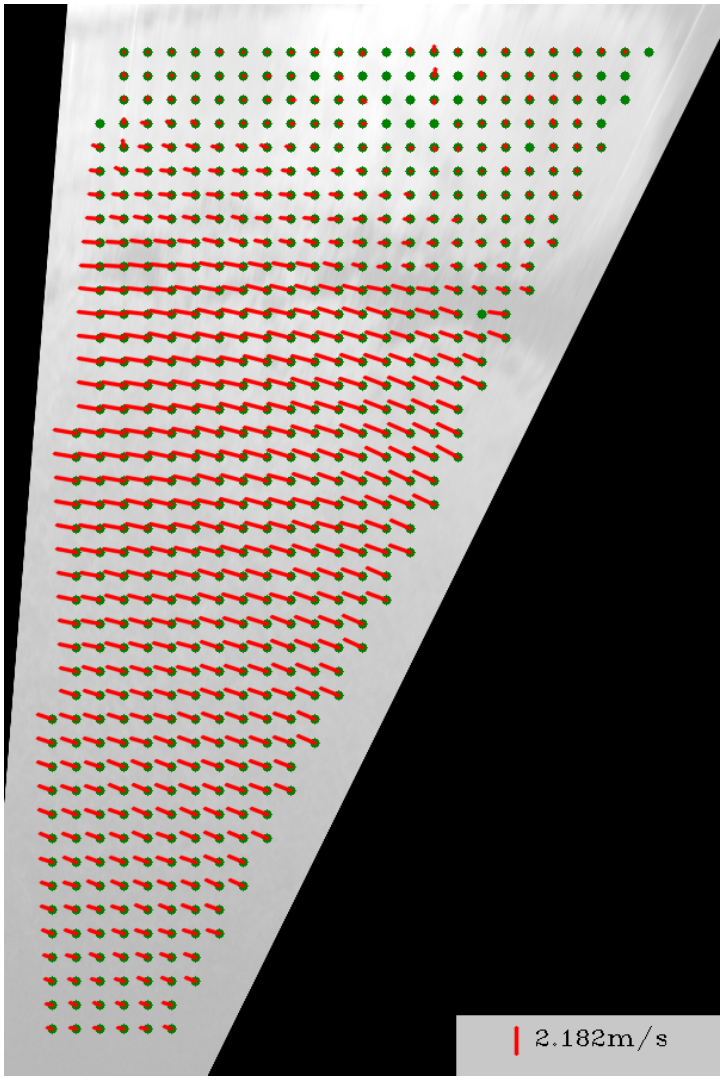

Figure 5.7: Hotel Experiment Results.

A number of video files were collected and processed during the time period from Aug. 8 to 18. The condition of the river remains mostly consistent during this period of time, but lighting conditions vary widely. Velocity readings over this period of time also remain

mostly consistent. Results from the time average of 800 frames collected at 16 frames per second on the morning of Aug. 16, 2010 are given in Figure 5.7. The flow at this location is mostly uniform in direction with velocity measurements of just over two meters per second in the center of the channel and slower measurements of less than one meter per second close to the southern shore and center sand bank. Video from the three the different capture devices was analyzed using EPIV with the best results typically from data generated by the FLIR device.

The FLIR device captures a lot of features in the water and produces a vector map that requires very little filtering of noisy vectors. Results from the other devices show the same velocity in most sections but the ground features visible in the color and NIR video cause a slower velocity reading for the southern most part of the river. Video from the FLIR device is more consistent and not as dependent on the lighting condition as the other two devices. All the results tend to agree that the velocity in the main channel is on average just over two meters per second.

The total time required to process 800 frames of video from at the hotel location took 46 seconds or 17.4 frames per second when using a single core processor with a GPU. Capture rate for the FLIR camera was set to 16 frames per second and could have been processed in real time with EPIV. Video recorded from this experiment was not compressed so performance when using input directly from a camera should be similar.

5.6 NPS Beach Experiment

NPS Beach Experiment was also located on the Kootenai River but at a position upstream from the Hotel Experiment. At this position the river is about 200 meters wide. The main channel of water flow is located between the center of the river and the northern bank with a slower channel between the center of the river and the southern bank. A small channel of water separates from the main river just west of NPS Beach. The two flow channels become separated by land for about 1.7km before rejoining. The flow of the river narrows and the direction shifts slightly more toward the south as the water travels west from NPS beach.

Both the NIR and Color cameras were positioned on the southern bank of the river at roughly 6 meters above the water level. Video from the NIR and Color devices was captured on Aug. 12 between the hours of 2050 and 2224 (GMT). Thirty minutes of video was recorded at a sampling rate of 7.5 frames per second for a total of 13500 frames. The FLIR device was positioned further upstream. Video from the FLIR was recorded on Aug. 12 between 2050 and 2224 (GMT) and on Aug. 18 between 1848 and 2010 (GMT).

Figure 5.8: NPS Beach Location.

On Aug. 12 the FLIR device was initially set to sample at a rate of 7.5 frames per second but was later changed to sample at a rate of 6 frames per second. On Aug. 18 the FLIR device was set to sample at a rate of 16 frames per second.

Lighting is an issue at the NPS beach location. On the video recorded from the FLIR device a strong glare on the surface of the water limits the amount of usable pixels in the video frames. The NIR device does not pick up many strong features in the water and inconsistencies in the lighting cause vertical lines in the mapped image that tend to cause correlation to fail in some regions. Lighting was most favorable for the color camera at NPS beach. The color camera recorded a large number of strong features in the river that were not affected by uneven lighting. None of the video collected at NPS beach records any significant underwater features but the fence posts used for control points caused some zones where PIV incorrectly reports a zero velocity.

5.6.1 Image Capture and Registration

The Color camera was configured using the same technique developed for the NIR device. First the external camera settings such as focus and aperture were adjusted manually to improve the number of features visible on the surface of the water. Next the polarizer was adjusted to remove any glare from the sun.

Figure 5.9: Control Points at NPS Beach.

Finally internal camera parameters such as shutter speed were adjusted to improve the image quality. EPIV does not currently support computing PIV on each channel of a color image, so images from the color camera are converted to grayscale before processing.

The technique used to generate a mapping from world to pixel space with the images collected at NPS beach involved determining the camera placement and the location of two control points. The color and NIR devices were placed on a tripod that was positioned directly over a point that had previously been surveyed. Position was taken directly from the survey point and the height of the cameras was computed from the height of the survey point plus the height of the tripod and camera mount. Field of view measurements for each camera were previously computed using video from the hotel location. The final three parameters, azimuth, tilt, and roll were determined by manual alignment of the mapped position of the control points to the first frame in each video. Control point alignment is shown in Figure 5.9. Green dots are used to show the position of the control points as computed by the mapping from world space to pixel space. Final measurements for azimuth, tilt, and roll are given in Table 5.1.

Image registration parameters for NPS beach location are given in Table 5.1. The color camera used at NPS beach collected video frames at a resolution of 1600x1200. These frames were registered to ground coordinates that ranged from 55600 to 555670 UTM East and from 5394510 to 5394610 UTM North resulting in a surface area of 70x100 meters.

Pixel size of the registered image was set to 0.0833 meters per pixel to produce a registered image of 840x1200 pixels. The registered image was generated at a height of 538.3552 NAVD88 meters which was the elevation of the water as measured at the NPS beach location.

5.6.2 PIV Configuration

PIV settings used for the NPS beach experiment are listed in Table 5.1. The correlation surface size was set to 2.75x2.0 meters or 27x21 pixels while the reference window was set to 3.5x3.5 meters. Video was recorded at 7.5 frames per second and the mapped conversion from world to pixel space was 0.1 meters per pixel. With δt set to 4 frames or 0.533 seconds the total amount of motion that may be detected using these settings is roughly 2.44 meters per second in the east or west direction and 1.875 meters per second in the north or south direction.

Sensors were placed in a grid pattern starting at 555600 UTM East and 5394500 UTM North and extending 120 meters in both directions. The spacing between sensors was set to 2.0 meters in both the East and North directions. Sensors that are not within the image or too close to an image edge are deactivated. The north most row of usable sensors were located at 5394608 UTM North and the east most row of usable sensors were positioned at 555668 UTM East. The south most row of usable sensors were located at 5394520 UTM North and the west most column of usable sensors were located at 555608 UTM East. A total of 902 sensors were determined to be usable.

5.6.3 Results

Results are given in Figure 5.10 and show velocity measurements that range from near 0 m/s along the shore line to as high as 1.5 m/s in the main channel. Within the main channel velocity measurements in the east part of the image tend to be slower than velocity measurements in the west part. The increased velocity in the west direction could be due to a narrowing of the main water channel. There are some regions of error in the vector field. Velocity measurements taken by sensors near fence posts tend to be too slow or in the wrong direction. Stable reflections visible on the water surface near to the shore may be causing the velocity measurements to be too slow in this region. Overall the results for this section of the river show a clear and mostly uniform flow pattern in a west direction that tends to agree with visual observation of the video and of the river.

The time required to process 400 frames of color video from NPS beach was 29 seconds or 13.8 frames per second on a single core processor and GPU.

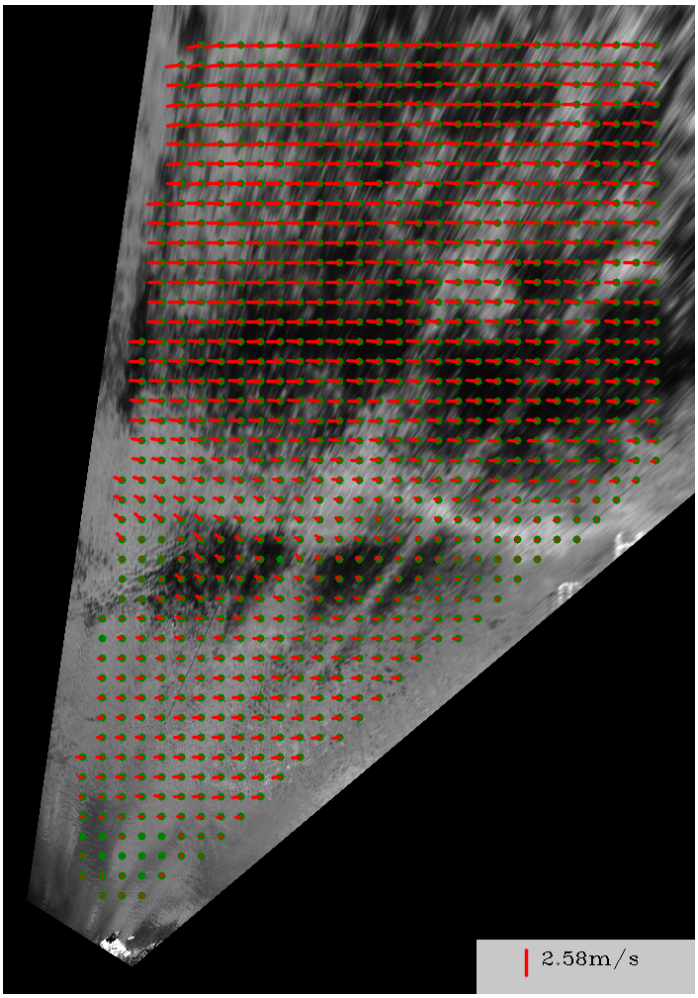

Figure 5.10: NPS Beach Results.

Video was captured at a rate of 7.5 frames per second so processing for this experiment could have been computed in real-time. The large number of sensors and the additional overhead from decoding color video make this experiment the most processor intensive. Even with these settings a GPU assisted EPIV is easily capable of processing frames in real-time.

5.6.4 Validation

Velocity measurements from an ADCP device were taken on a part of the river that was recorded by the color video camera. Results from the ADCP device give the velocity in the main channel to be from 1.3 to 1.5 m/s and in the slower regions to the south to be from 0.4 to 0.6 m/s. Measurements from the ADCP device agree with the PIV results that show a general acceleration in the water as it moves to the west. PIV results were in general a little

slow but mostly agree with the ADCP measurements in both direction and velocity.

5.7 Conclusion

Results from the three experiments discussed above show that the EPIV software is capable of producing accurate PIV measurements quickly. EPIV was capable of processing data from each of the experiments in real time with only a single core processor and a high end GPU. Results also show that EPIV measurements agree with ADCP measurements to within 15% difference. These experiments also show that EPIV is capable of recording velocity measurements based on different features. The Wolf River velocity measurements are based on the speed and path of foam particle on the surface of the water, and velocity measurements from the Kootenai river were based on the speed and path of surface disturbances. Velocity of these two different types of surface features were correctly measured by EPIV.

Chapter 6
DISCUSSION

A number of issues involving the processing performance and the accuracy of riverine PIV experiments were addressed. Sources of correlation error were analyzed and new scoring methods were developed and successfully used to remove many of the erroneous vectors common when PIV is used on natural flows. The processing performance of the direct and FFT methods to compute correlation were measured. A new GPU direct correlation algorithm was developed and measured to be as fast or faster than the CPU FFT method. Processing performance of different correlation methods was measured and the coefficient method was found to be only slightly slower than the other more simple methods due to the use of sum tables on the CPU and free processing resources on the GPU.

New PIV software called EPIV was developed specifically to process video of river flows under non-ideal conditions in real time. EPIV was successfully used to process the data collected by a color sensor, a NIR sensor, and an FLIR sensor. EPIV was able to run on a portable computer and on a high end system equipped with a GPU. Real time video processing was possible using EPIV with common PIV settings. Even a low end Intel Atom based system was powerful enough to compute a number of PIV measurements in real-time. The addition of a GPU to the Atom platform would provide a significant speed improvement in the range of 10-20x.

Video recorded at the three three different locations was processed by the EPIV software. For each experiment EPIV was used to generate a mapping from world space to image space and then to generate a registered image. The registered images were processed using PIV and a vector map was generated that showed the time average of the results. The Wolf River in Southern Mississippi was shown to have a current of roughly 0.6 m/s in the main channel during the day of Aug. 25, 2010 and roughly 0.9 m/s in the main channel during the day of Aug. 26, 2010. The second experiment was conducted on the Kootenai River in Northern Idaho and involved measurements taken over the course of a number of days at two different locations. At the hotel location the velocity in the main channel was steady and ranged from 1.9 to 2.1 m/s. At the second location called NPS beach the velocity measurements in the main channel ranged from 1.2 to 1.5 m/s. Results from each experiment compared favorably to measurements taken from other devices. Measurements of processing speed show that EPIV should be capable of producing PIV results in real time under most circumstances.

Erroneous vectors produced when an insufficient number of features were present in some of the data collected at these experiment locations was removed by a technique called wscore and a number of additions and modifications to wscore. Performance was improved to support real time processing by using a parallel software design and by offloading the expensive correlation task to the GPU. A time average method for steady state experiments that includes only vectors considered by wscore to be good was used to produce measurements that are not greatly influenced by inaccurate vectors generated by noise. Using these techniques, the measurements computed by EPIV tend to agree with measurements from other devices that record surface velocity to within 16% difference. Results from these experiments show that river surface velocity may be accurately measured in real time using desktop class computer hardware.

A number of improvements could be made to EPIV to make it more accurate and easy to use. A method could be developing to automate the generation of camera and position parameters. The current method of manually aligning the control points in the image is both slow and likely the source of some error. To further reduce mapping error, an image distortion correction method should be added to EPIV. Using information about the GPU, CPU, and problem size EPIV could be made to automatically determine the fastest processing platform and switch from GPU to CPU as needed. For lower power computer systems the CPU load could be further reduced by processing the correlation surfaces on the GPU. Post-processing methods commonly used in PIV experiments could be added to EPIV to further reduce the impact noisy vectors have on the final time average result.

BIBLIOGRAPHY

[1] R. J. Adrian. Twenty years of particles image velocimetry. *Experiments in Fluids*, 39:159–169, 2005.

[2] Abderrahmane Bennis. *Implementing a Highly Parameterized Digital PIV System On Reconfigurable Hardware*. PhD thesis, Northeastern University, 2010.

[3] A. Allen Bradley, Anton Kruger, Ehab A. Meselhe, and Marian V. I. Muste. Flow measurement in streams using video imagery. *Water Resources Research*, 38, 2002.

[4] Kai Briechlie and Uwe D. Hanebeck. Template matching using fast normalized cross correlation, 2001.

[5] F. C. Crow. Summed-area tables for texture mapping. *Computer Graphics*, 1984.

[6] J Duncan, T Bryce, H Thomsen, D Dabiri, J R Hove, and M Gharib. An extended study of a generalized digital particle image velocimetry (dpiv) processing technique. *Measurement Science and Technology*, 20, 2009.

[7] R.B. Fisher and D. K. Naidu. A comparison of algorithms for subpixel peak detection. In *Image technology, Advances in Image Processing, Multimedia and Machine Vision*, pages 385 –404. Springer-Verlag, 1996.

[8] Lichuan Gui, Wolfgang Merzkirch, and Ralph Lindken. An advanced mqd tracking algorithm for dpiv. In *Applications of Laser Techniques to fluid Mechanics*, Lisbon Portugal, 1998.

[9] Douglas P. Hart. The elimination of correlation errors in piv processing. In *9th Internaltional Symposium on Applications of Laser Techniques to Fluid Mechanics*, Lisbon Portugal, 1998.

[10] Douglas P. Hart. High-speed piv analysis using compressed image correlation. *Journal of Fluids Engineering*, 120:463–470, 1998.

[11] A. Hauet, M. Muste, and H-C. Ho. Digital mapping of riverine waterway hydrodynamic geomorphic features. *Earth Surface Processes and Landforms*, 34:242–252, 2009.

[12] K. T. Holland, J. A. Puleo, and T. N. Kooney. Quantification of swash flows using video-based particle image velocimetry. *Coastal Engineering*, 44:65–77, 2001.

[13] K. Todd Holland, Robert A. Holman, Thomas C. Lippmann, and John Stanley. Practical use of video imagery in nearshore oceanographic field studies. *The Journal of Oceanic Engineering*, 22:81–92, 1997.

[14] H. T. Huang and H. E. Fiedler. Deformed particle image pattern matching in particle image velocimetry. *Applied Scientific Research*, 51:179–183, 1993.

[15] Andrew T. Jessup, Robert L. Street, Stephen G. Monismith, and Alexander R. Horner-Devine. Remote sensing and modeling of coherent structures in river and estuarine flows. Accessed from http://www.onr.navy.mil/reports/FY09/pojessup.pdf, 2009.

[16] J. P. Lewis. Fast normalized cross-correlation, 1995.

[17] E. M. Mikhail, J. Bethel, and J. C. McGlone. *Introduction to Modern Photogrammetry*. John Wiley and Sons, Inc., 2001.

[18] Nobuhito Mori and Kuang-An Chang. Introduction to mpiv, 2003. http://sauron.civil.eng.osaka-cu.ac.jp/ mori.

[19] M. Piirto, H. Eloranta, P. Saarenrinne, and R. Karvinen. A comparative study of five different piv interrogation algorithms. *Experiments in Fluids*, 39:571–588, 2005.

[20] O. Pust. Piv: Direct cross-correlation compared with fft-based cross-correlation. *Proceedings of the 10th Internation Symposium on Applications of Laser Techniques to Fluid Mechanics*, 2000.

[21] M. Raffel, C. E. Willer, and J. Kompenhaus. *Particle Image Velocimetry: A Practical Guide*. Springer, 1998.

[22] T. Schiwietz. Gpu-piv. *Proceedings of the Vision, Modeling, and Visualization Conference*, 2004.

[23] W. A. M. Nimmo Smith, P Atsavapranee, J. Katz, and T. R. Osborn. Piv measurements in the bottom boundary layer of the coastal ocean. *Experiments in Fluids*, 33:962–971, 2002.

[24] L. Stromungsmechanik. Tracer particles and seeding for particle image velocimetry. *Measurement Science and Technology*, 8:1406–1416, 1997.

[25] Shuhei Tarashima, Manabu Tange, Satoshi Someya, and Koji Okamoto. Gpu accelerated direct cross-correlation piv with window deformation. *15th Int Symp on Applications of Laser Techniques to Fluid Mechanics*, 2010.

[26] R Theunissen, F Scarano, and M L Riethmuller. An adaptive sampling and windowing interrogation method in piv. *Measurement Science and Technology*, 18:275–287, 2007.

[27] P. Vennemann. *Particle image velocimetry for microscale blood flow measurement*. PhD thesis, Delft University of Technology, 2008.

[28] V. Venugopal, C. Patterson, and K. Shinpaugh. Accelerating particle image velocimetry using hybrid architectures. *Symposium on Application Accelerators in High Performance Computing*, 2009.

[29] V. Weitbrecht, G. Kuhn, and G.H. Jirka. Large scale piv-measurements at the surface of shallow water flows. *Flow Measurement and Instrumentation*, 13:237–245, 2002.

[30] J. Westerwheel. *Digital Particle Image Velocimetry*. PhD thesis, Delft University of Technology, 1993.

[31] J. Westerwheel. Theoretical analysis of the measurement precision in particle image velocimetry. *Experiments in Fluids*, 29:S003–S012, 2000.

[32] H. Yu, M. Lesser, G. Tadmor, and S. Siegel. Real-time particle image velocimetry for feedback loops using fpga implementation. *43rd AIAA Aerospace Sciences Meeting and Exhibit*, 2005.

[33] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:1330–1334, 2000.