This thesis is dedicated to the memory of my parents and sister, Bob, Chris and Iva. This is for them.

My grandparents have been my best friends, my counselors, and everything good in my life, and I wouldn't have been able to do this without their love and support. Rick and Jeannie have been my helpers in anything and everything I needed in life, and Bill has been the best uncle any nephew could ask for. I could never have hoped to have a more loving, supporting, and happy family.

Fritz H. Hieb
New Mexico Institute of Mining and Technology
May, 2017

DEVELOPMENT OF A DIGITAL IMAGE CORRELATION AND PARTICLE IMAGE VELOCIMETRY SOFTWARE PACKAGE DESIGNED FOR UNIVERSITY MECHANICAL ENGINEERING APPLICATIONS

by

Fritz H. Hieb

Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Science in Mechanical Engineering

New Mexico Institute of Mining and Technology
Socorro, New Mexico
May, 2017

ABSTRACT

Digital image correlation (DIC) and particle image velocimetry (PIV) are non-destructive digital measurement techniques. High-speed, high definition cameras capture detailed images of particle movement in a target area. The images are subsequently fed into software to analyze the particle movements via correlation algorithms, then output the motion characteristics of the particles in the overall target region. DIC is geared primarily toward solid material mechanics, analyzing and describing deformation and strain characteristics in a range of applications, from small-scale material samples to large scale structures such as bridge and building supports. PIV deals in the realm of fluid mechanics, characterizing the flow patterns and turbulence components of gas and liquid systems.

The correlation software component is a limiting factor due to the cost and availability of viable free programs. Currently, many options for DIC and PIV professional software exist on the market that perform high-quality, high-definition, reliable analysis. Yet these industry-scale programs are very expensive, and out of reach to a typical university engineering student, as well as many departmental budgets. Currently, free options in existence are limited, requiring access to third-party software or the knowledge of code compilation, and rarely do they exhibit cross-platform portability.

The goal of this project is to develop a software suite for engineering students and instructors that combines both PIV and DIC into a single user-friendly, graphical interface-based software package that serves basic engineering needs, and has a low barrier of entry. The user of this program will be provided with the tools needed to perform a moderately simplified, speedy correlation analysis. The raw data output of the analysis will then be available for outside study that serves to increase the learning component of the project, providing instructors latitude on which components of the process they are interested in teaching.

This developed program uses subset-based, zero-mean direct cross-correlation for DIC. The algorithm applies an integer-based search scheme developed here to find the nearest correlating coordinates, then offers the option to apply a subpixel interpolation scheme that resolves the particle movement accuracy to subpixel dimensions. The user interface displays the results of the search in the form of a contour interval heat-map, and offers the data for export. For particle image velocimetry, Fourier transform phase correlation is used as the primary correlation mechanism. Subpixel components are available using the direct cross-correlation mechanism. Visual feedback is provided in the form of on-screen, adjustable velocity vectors, with raw data output available.

Material-deformation and particle-movement experiments performed for this

project provided raw input images to test the software. Results from three professional and well-cited programs provided baseline data to compare and verify the results of this program. The developed DIC algorithm performed quickly and accurately compared to the baseline, up to the point of significant sample deformation. At this point, the algorithm had difficulty locating tolerable correlation matches only in the areas of high deformation, resulting in a moderate break-down in the accuracy of the results in the significantly strained regions of the sample. In the PIV data comparison, there was variability between the results from the two baseline programs and this one. Each program produced slightly different results, though the overall trend of the data was the same. In particle motion applications, it is apparent that the differences in proprietary algorithms, as well as the general volatility of fluid particle movement, causes higher variability in the results than in DIC.

Keywords: Digital image correlation, particle image velocimetry, cross-correlation, PIV, DIC, phase transform

ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Michael Hargather for accepting me as his student, and for taking on this project. I enjoyed his advisement and instruction throughout this process immensely.

Dr. Jamie Kimberly brought extensive knowledge to the endeavor, and I appreciate all of his time and efforts to make the project successful.

Dr. Kent Dybvig provided excellent and thoughtful feedback in the final stages of the project, and also brought a deep family connection and a personal component that I truly valued, and I thank him for that.

Additionally, I received help from several faculty members, namely Dr. David Grow and Dr. Curtis O'Malley, they were a tremendous help in several major components of this project.

Lastly, none of this would be possible if my good friend Jason Lee had not convinced me to apply to the mechanical engineering graduate program at New Mexico Tech. I didn't think I had what it took, but he convinced me otherwise.

Thank you again to everybody.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Motivation

Engineering students are increasingly counted on to be familiar upon graduation with the most technologically advanced concepts and methods of analysis. Visual image correlation (VIC) is one such concept, and can be aptly described as a method of recording the translation and deformation of particle groupings throughout a sequence of images, or visuals. There are two nominal subsystems in VIC: digital image correlation (DIC), and particle image velocimetry (PIV). Both are computerized particle tracking systems; however the terminology that has developed separates solid material mechanics (DIC) from fluid particle tracking (PIV). PIV stricly targets the velocities of particle motion, with the goal being to develop a map of vectors that show the flow and movement of fluid in an area of interest. As an example, the air flowing over a wing can be analyzed in a wind tunnel by "seeding" the airflow with smoke and using differential illumination to develop a model of the wing aerodynamics [Arnott et al., 2003]. DIC is concerned with the general deformation of a solid sample through time, which grants information on material properties and structural responses. An important real-world direction DIC is taking is in application to structural health monitoring, in the form of bridge health and safety. In a simple application, a camera can be focused on a bridge support to ascertain its structural integrity [Yoneyama et al., 2006]. DIC is increasingly being looked at as a cost-effective, thorough, and non-destructive method of tracking the structural health of bridge supports and roadways.

To utilize either PIV or DIC in an experiment or application, there are only three basic requirements:

- Image or video capture device
- Computer (laptop or workstation)
- DIC/PIV analysis software

The economics of the first two of these requirements have become increasingly attractive over the years. Digital camera technology and computing power has arrived at the point where the devices are cheap enough, and small enough, to

be obtainable for previously cost-prohibitive application concepts. Depending on the goals of the project, a simple setup of a high-resolution mobile phone and laptop computer may be sufficient.

However, a major obstacle exists to the third requirement: there is a limited availability of inexpensive (or free) useable analysis software that enjoys a shallow learning curve and does not require the use of third-party monetized software. A typical DIC or PIV program operates under a simple precept: analyze a set of user-inputted images to obtain the particle motion in the images through a progression of time. However, the implementation of this relatively simple concept can be expensive to develop, due to the complexity of the many different shapes and forms the subtleties of particle motion exhibit. There are many different mathematical treatments of particle motion, and developing them into a meaningful program can be a costly endeavor.

Accordingly, the professional software that exists is very expensive and out of reach for students and some university engineering departmental budgets. A paid license for an industry-level PIV or DIC software suite reaches reliably into the tens of thousands of dollars.

There are, however, several free or open source programs that are available for both PIV and DIC. For PIV, the most well-documented options are OpenPIV [Openpiv, 2017], MatPIV [Oslo, 2017], and PIVlab [Thielicke and Stamhuis, 2014a]. MatPIV and PIVlab are both well-cited programs that have been used successfully in outside applications and research projects. However, each is a Matlab application, which means that the user most own a copy of the Matlab software package, including the Image Processing Toolbox add-on, in order to use [Mathworks]. While Matlab is available at reduced cost to students and academia in general, the restriction to this operating environment is not ideal. OpenCV is an open-source program that is moderately updated and maintained. However, it is only available as a Matlab app, or as Python or C++ code that must be compiled or scripted within suitable environments. These options are not favorable to a novice user or for quick classroom demonstration. In the DIC realm, not many free applications exist. Ncorr, developed as part of a Master's thesis, is a Matlab application that provides basic DIC functionality.

The lack of options for students to acquire cheap, easy access to DIC and PIV software is the motivation behind this thesis. This project aims to provide a basic software application that provides *both* DIC and PIV functionality. At this juncture, we are unsure if it can be provided and maintained as a free or open-source application, or if there will be a small cost associated with the use or licensing of this product.

1.2 Project goals

There are two fundamental objectives of this project: provide the user with the tools to understand how the process works via program feedback, and provide the user with accurate raw data to analyze in an extracurricular activity.

Program feedback The user of DIC and PIV software necessarily learns quite a bit about the intricacies of each of the techniques, because in order to obtain good results from any of these, a working knowledge of the process must either be had beforehand or obtained during operation. In order to learn during operation, feedback must be available to the user in the form of data and/or visuals. In DIC, colored contours on the images are necessary to inform the user of the success of various necessary input parameters. In PIV, velocity vectors are provided. Additionally, the success rate of various aspects of the program must be made available. DIC and PIV are influenced greatly by a number of factors in the analysis process, and that influence should be indicated succinctly and accurately in real-time output.

Raw data While there is some information that can be gained directly from the user interface, much more data and detailed analysis can be had by an outside evaluation of the raw data. Many PIV and DIC software packages provide these analysis tools; however, this program is not meant to be a one-stop shop. The raw data is provided in a comma-separated value format (CSV), with attached read-me files. Data are often difficult to work with outside of the source program, due to indecipherable headers or just a general lack of information on the formatting. For this program, are provided clear and detailed instructions are provided that do not suffer for lack of clarity.

The program is developed in a manner such that (for the end user) it does not :

1. Require prior knowledge of computer programming
2. Require any third-party paid software (such as Matlab)
3. Use command-line or scripting environments

The program also:

1. Provides step-by-step instructions to the user on-screen
2. Provides easy access to help files
3. Has only the most necessary options available (for simplicity)
4. Is user error-resistant
5. Runs cross-platform (Windows, Mac, Linux)

1.3 Choice of programming language

The reason why Matlab has been used almost exclusively to create the available free PIV and DIC applications is because the programming knowledge required is limited; many engineering students have at very least a basic knowledge and experience with the Matlab language and environment. Another important reason is that Matlab provides all of the required user interface creation, image processing, and image displaying tools that take many programmer hours to develop. And thirdly, no matter what operating system the app is run on, since it runs in the Matlab scripting environment, the user would see no difference and the developer would spend no extra time developing cross-platform ports.

These features of Matlab were all taken into consideration when choosing the programming language, because the more pre-written routines and libraries that could be used, the better the program would turn out within the limited time frame of development. In the end, the Java programming language was chosen in which to develop the software. For an algorithm-heavy program, an object-oriented language such as Java is frequently not the optimal solution. However, due to Java being a "write once, run anywhere" language, the requirement of true cross-platform portability is met without adding extra development hours. Java requires the Java Runtime Environment (JRE), however on many machines this software is already installed and running and if not, is a simple update.

Java also provides the graphical user interface (GUI) features required to make a user-friendly program. Creating a GUI in Java is a simple, painless and high-level process, freeing up more time for behind-the-scenes algorithm and functional development. Additionally, open-source libraries are available for rendering different image formats, and extracting still frames from video.

1.4 Fundamentals of PIV and DIC

In a broad sense, the two methods are similar, as they are both particle tracking concepts. However, due to the very different nature of solids and fluids, the approaches to search and tracking use different mathematical concepts.

1.4.1 Principles of DIC

The essence of subset-based DIC lies in the tracking of a particle (or group of particles) within a region of interest (ROI) from one location to another, throughout a succession of images. The ROI is divided into sections based on the density of point data desired, resulting in a virtual interrogation grid. At each grid intersection, the center of a square virtual subset, also known as the kernel, is placed in the reference image ($P(x_0, y_0)$), shown in Fig. 1.1. The subset, of size (2M - 1) x (2M - 1), where M is the subset width, is shifted around the deformed image in

an effort to acquire the centroid of the location of a similar grouping of particles ($P'(x'_0, y'_0)$ as in the original location by centering the subset on neighboring pixels and applying a correlation function. The search results in a field of correlation values, ranging from -1 (perfectly negative correlation) to 1 (perfect correlation), where the peak magnitude of the field is the pixel location of the maximum correlation. The matching mechanism relies on a set of similarity correlation criterion applied to the grayscale values that fall within the subset boundaries, using the following definition for zero-normalized sum of square differences (ZNSSD) from Zhou et al. [2012] and Pan and Li [2011]:

$$C_{ZNSSD}(\mathbf{p}) = \frac{\sum_{x=-M}^{M}\sum_{y=-M}^{M}[f(x,y) - f_m][g(x',y') - g_m]}{\sqrt{\sum_{x=-M}^{M}\sum_{y=-M}^{M}[f(x,y) - f_m]^2}\sqrt{\sum_{x=-M}^{M}\sum_{y=-M}^{M}[g(x',y') - g_m]^2}} \tag{1.1}$$

where: $f(x,y)$ is the grayscale intensity at coordinates x, y in the reference (undeformed) subset, and $g(x',y')$ is the grayscale intensity at coordinates x', y' in the deformed subset. The mean intensity value for the reference subset is defined as: $f_m = 1/(2M+1)^2 \sum_{x=-M}^{M}\sum_{y=-M}^{M}[f(x,y)]$, and the deformed subset mean intensity is: $g_m = 1/(2M+1)^2 \sum_{x=-M}^{M}\sum_{y=-M}^{M}[g(x',y')]$. $\mathbf{P}$ is the deformation vector.

C_{znssd} from Eq. 1.1 is computationally intensive, compared to other cross-correlation formulations, however offers the advantage of being independent to shifts and offsets in light intensity and contrast between the reference and deformed image subsets [Athreyas et al., 2014]. The summations in the denominator are the variances of the reference and deformed subsets, making the coefficient independent of grayscale intensity shifts.

With enough successful kernel correlations, a full-field map of displacements in the interrogated area can be obtained, which may then be post-processed into a full-field strain map. To gain desired spatial resolution, the size of the grid step (width of the virtual grid cells) is set at a value much smaller than the kernel width. A common value in DIC processing is a 75% overlap of the kernel.

Tracking the subset in a sample undergoing deformation can be problematic. As the sample deforms, the subset deforms as well, which leads to a drop in the confidence of the calculated correlation value. This loss in confidence is referred to as decorrelation effect, as the surface pattern becomes mathematically unrecognizable from the original. To deal with this, some DIC algorithms deform the grayscale values in the reference subset virtually using interpolation and affine transformation mechanisms, in an effort to more closely match the subset in the deformed image. This process can prolong the time until failure of the correlation algorithm, and also provides additional deformation details such as rotation, shear and skew. Due to the limited scope of this project, the DIC algorithm created here tracks only the square kernel in the deformed image without using virtual deformation matching means.

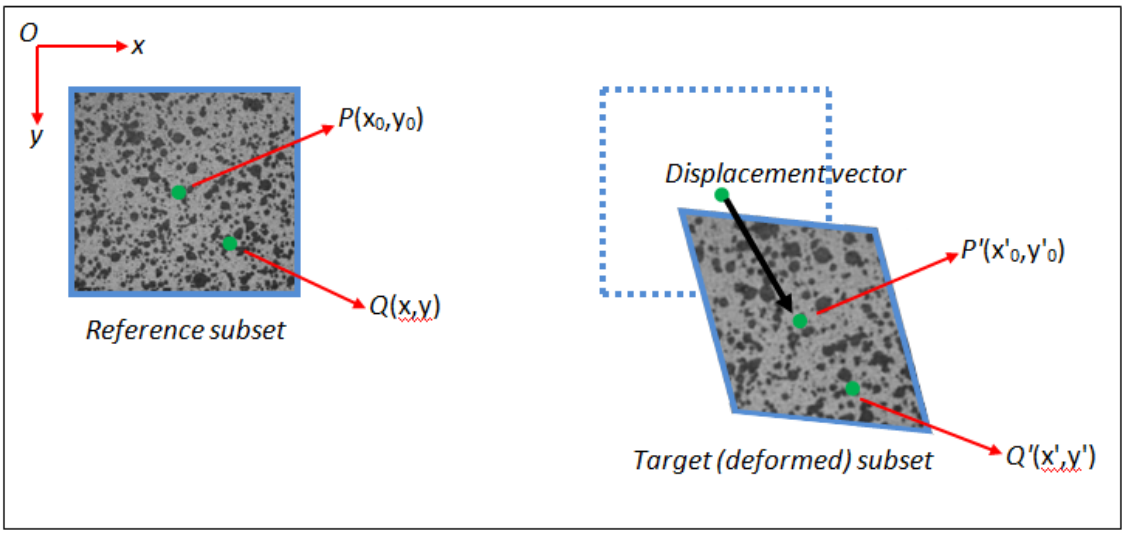


Figure 1.1: Principle of subset-based digital image correlation where the subset is tracked to its corresponding location in the deformed image, resulting in a deformation vector.

1.4.2 DIC experiment requirements

There are two elements of a successful DIC experiment and analysis:

- Equipment
- Sample preparation

Equipment The range of equipment is diverse; depending on the material being analyzed and the scope of the desired results, the equipment can range from a simple web-cam directed at a structural support, all the way to a highly intricate multi-camera (high definition) setup using a hydraulic materials testing machine. There is no "correct" way to perform DIC; the accuracy and precision of the equipment is driven by the accuracy and precision desired in the results.

Sample preparation In DIC, the preparation and production of the sample surface can be a rather intricate and finicky process, while being decidedly low-tech. The surface pattern should be high-contrast, non-repetitive, isotropic, and coarse. A repetitive and anisotropic field may give false correlation matches where the pattern is repeated in multiple locations. If the contrast is too low, not enough information is provided to develop a correlation peak that is sufficiently outstanding from the surrounding area; there could be multiple peaks of similar heights,

resulting in error. And lastly, the scale of the pattern should be coarse, yet not too coarse. In the case of large-diameter (too-coarse) speckles, the subset kernel size must necessarily be of equal or larger dimension of the largest "average" speckle size. Otherwise, the kernel will have areas of extremely low contrast where it is completely encapsulated by the speckle. This results in a loss of spatial resolution, where small deformations will either not be accurately tracked, or unobserved completely. If the speckle pattern is too fine, then camera resolution becomes an issue. Aliasing in DIC occurs when the resolution of the camera is not high enough to accurately capture the pattern, negatively influencing the analysis accuracy. In general, the smallest mean speckle diameter that may be accurately recorded and analyzed is three (3) pixels [lecompte et al., 2005]. However, when such a small mean speckle size is present, the kernel size should be smaller as well (to capture smaller movements). This causes the subsets to suffer from a lack of uniqueness, which causes problems during correlation. A mean speckle size in the range of 5 - 10 pixels has been found to produce good results [Lecompte et al., 2007].

To produce an effective speckle pattern, the most common technique involves applying flat black paint to a flat white background using a sputter spray technique. White paint may be applied to black, however white paint is less opaque so the underlying black components may show through, negatively affecting the pattern. Flat paints are superior for DIC due to the lower intensity of light reflection. A lower albedo reduces the variation in perceived intensity by the camera as the sample deforms.

1.4.3 Principles of PIV

In a PIV experiment, images are taken as back-to-back pairs, typically labeled A-B fashion. However unlike DIC, instead of comparing an entire image set of deformed images against the first (reference) image, only B is compared to A for each pair of images. The motion for each desired point index is recorded, and the data at each point in each image pair are typically averaged together at the end to create a velocity vector grid. Other methods, such as using a median velocity, may be used.

In particle image velocimetry, a grouping of fluid particles can be thought of in much the same manner as a grouping of speckle patterns painted on the surface of a specimen for DIC analysis. However, fluid particle flows introduce factors that limit the effectiveness of direct subset cross-correlation, and especially limit its speed. Fluids exhibit much more random movement than a solid undergoing deformation, as the particles in many cases do not adhere to one other. Additionally, potential out-of-plane motion of the particles causes the intensity of the light falling on the particles to transition between images. Due to these factors, subset correlation fails often fails. When correlation fails, the search algorithm must make a choice: either continue the search, or give up and move to the next grid location. The problem arises that for a particular subset location, the correlation

may never find another good enough match in the deformed image. A continued search makes the program highly inefficient and slow, yet if the algorithm gives up too quickly, potential information may be lost.

To combat this, a different type of correlation is used that analyzes the images in the frequency domain. As in DIC, a window (analogous to subset) size and grid step are chosen. For PIV, the grid step is typically in the range of half the value of the window size, and the window size should be large enough to encapsulate any particle movements in the ROI. However instead of shifting the window around the image looking for a peak correlation values, a single operation, called phase correlation, is performed on the two virtual windows, which are both centered on the grid indexes in the A and B images. From this operation, the location of the most matching pixel in image B is detected. While seemingly simpler, the computational cost of performing a phase correlation is magnitudes higher than a simple direct cross-correlation, so isn't used in DIC.

1.4.4 PIV experiment requirements

Particle image velocimetry experiments are more open-ended than digital image correlation. The fluid may be a gas, or liquid anywhere on the viscosity scale. The only requirement is that the fluid be seeded with a particle that diffuses and proliferates throughout the region of interest.

The camera requirements are mandated by the speed of movement of the fluid. In a highly viscous flow, the image pairs do not necessarily have to be taken within an instant of each other. However, when capturing a fast-moving gas or liquid, the motion of the particles relative to each other is high, so a faster camera shutter must be used. These are all subjective qualities of each individual experiment, where judgments must be made by the experimenter.

CHAPTER 2

SEARCH ALGORITHMS

This chapter details the search algorithms used in the program:

1. Integer-pixel search algorithm developed here for digital image correlation (DIC).
2. Integer-pixel phase correlation for particle image velocimetry (PIV).
3. Implementations of sub-pixel accuracy.
4. Multi-threading of the search process

DIC and PIV have been found to require fundamentally opposite approaches, even though both methods are generally, particle tracking implementations. When tracking a speckle image as it deforms on a material, the observer can count on a general orderliness of particle movement, due to the principles of continuous strain. Because the material is of a solid nature, it can be entirely expected and even predicted that the tracking pattern will not undertake many, if any, random or disorganized movements. Therefore a search scheme can look to anticipate future particle movement to inform the algorithm of the possible next move it should make, and therefore substantially increase the search efficiency.

In PIV however, the images are expected to exhibit no predictive movement, as fluids move in a less orderly and more random fashion. Therefore the correlation algorithm can not predict general pattern movement, resulting in longer search intervals to acquire cross-correlation. An algorithm that excels at DIC cannot be said to do the same for PIV; however, as implemented here, the differences between the two can be negotiated to form one algorithmic backbone that combines some of the shared general principles, applied in different manners.

2.1 Integer-pixel search routine for DIC

The first step in a digital image correlation procedure is to identify the highest correlated pixel location. There are several published approaches to this, yet most

work focuses on sub-pixel search techniques. The effectiveness of the integer-pixel search process is of fundamental importance, because any sub-pixel measurements made thereafter are duly affected by its accuracy.

The elementary complication of a pixel-search routine is due to the sheer amount of pixels that must be assessed in a high-resolution image. For example, using an image of resolution 1024×640 pixels, there are then $655,360$ eligible locations that the highest-correlated pixel could have relocated to. In a brute-force search scheme, every subset is searched at each pixel coordinate, for a total of 4.29×10^{11} calculations, which would take many hours for a single image. Obviously, a useable system cannot be developed with this method.

2.1.1 Alternative approaches

Several solutions to this problem have been developed; Zhang et al. [2006] developed an approach using an affine transform parameter calculated through the identification of several points in the undeformed and deformed images. Once the affine function is calculated, this parameter is then used as an initial guess for the points around it, in a seeding fashion. The drawback to this method is that each point must be manually located, because the points must be of distinct grayscale differentiation from the surrounding pixels in order to calculate an accurate affine transform. In this application however, user abstraction (less parameter input) is of fundamental importance, so this approach was not considered further.

Another popular approach is to use a quad-tree splitting process [Sousa et al., 2011]. This technique recursively splits the image into four (4) sub-images, splitting each sub-image over and over again until one of the following conditions is met:

1. Displacement calculated from one level to another does not follow smooth behavior.
2. Displacement calculated from one level to another is the same.
3. Maximum number of splitting levels is reached.

The benefits of this approach are that a bare minimum of coarse-search correlation calculations are performed, making the routine efficient. However it was unclear how the algorithm performs in regions of interest in a non-uniform geometry, which are are more common than not in a real-world DIC application, so was not considered further for implementation.

While most search routines establish a coarse displacement map before pursuing sub-pixel search, Pan [2009] developed a "reliability-guided" process that calculates the sub-pixel location without need for an integer-pixel guess. The process is as follows:

1. Manually locate a seed point in an area of low deformation.
2. Calculate the correlation coefficient for four (4) neighboring points.
3. Insert the points into an ascending correlation coefficient and coordinates queue.
4. Take the point from the top of queue, then calculate four (4) neighboring points.
5. Repeat steps 2-4 until all pixels are processed.

In the end this approach was not used for various implementation reasons, however credit is due to the authors for providing a cornerstone to the coefficient-guided search algorithm developed for this application.

2.1.2 Adaptive search scheme development for DIC

As stated previously, the two keys to a practical, effective coarse-pixel correlation search are speed and accuracy. For this application, speed is prized above accuracy because an in-classroom system is the ultimate objective, where the need the maximum possible accuracy is of less importance. However, speed is a tricky thing to accomplish effectively in DIC due to constant changes in sample geometry from testing deformation. The scheme must also not completely sacrifice accuracy for the sake of speed, so the balance between the two is always in fluctuation as the system develops. The following is an accounting of some of the problems that arise, and must be solved, when developing a search scheme for this application:

- There should require no input from the user on the magnitude of deformation. The algorithm therefore will have no information on the maximum distance to search from the current reference subset.
- The direction of force applied or general particle motion shall not be a required input, so the algorithm cannot "cheat" and check only in a known direction, which leaves every image pixel as a possibility.
- The speed of deformation will be unknown. Therefore between each image the reference subset may have jumped from zero pixels to n pixels, with no constant between images.
- The user may input any images from the set, so without careful selection there may result a large discontinuity in any deformation patterns developed during DIC testing.

To solve the above complications (and more not listed), the following will develop a search scheme that is both fast, reliable, and adaptable. The overarching purpose of each concept can trace back to a single fundamental ambition: to decrease the number of pixel subsets to which the correlation equation is applied.

0.12	0.99	0.01	0.08	0.15	0.37	0.14	0.51	0.38	0.4
0.08	0.8	0.07	0.14	0.16	0.63	0.55	0.57	0.24	0.41
0.04	0.01	0.18	0.2	0.22	0.54	0.56	0.73	0.7	0.47
0.85	0.47	0.39	0.31	0.03	0.6	0.62	0.69	0.71	0.28
0.46	0.53	0.55	0.42	0.39	0.51	0.28	0.15	0.52	0.04
0.47	0.54	0.76	0.83	0.60	0.42	0.49	0.26	0.33	0.25
0.33	0.94	0.82	0.89	0.63	0.48	0.3	0.32	0.39	0.36
0.79	0.91	0.99	0.97	0.91	0.39	0.31	0.38	0.25	0.22
0.61	0.84	0.90	0.87	0.68	0.35	0.37	0.34	0.16	0.53
0.71	0.68	.43	0.77	0.84	0.36	0.43	0.5	0.27	0.59

Figure 2.1: A section of image correlation coefficient values using a 21 $\times$ 21 pixel subset. Coefficients increase with proximity to the maximum correlated value.

Coefficient-guided pattern A useful feature of correlation coefficients is that the closer a subset is to the highest correlated value, the neighboring pixels also exhibit relatively high correlation coefficients, shown in Fig. 2.1. This trait can be taken advantage of to efficiently "guide" the search process in the direction of highest correlation, which reduces the number of required correlation calculations.

In image processing, pixels are always aligned in a grid, so there are nominally two patterns to apply interpolations, searches, etc. An 8-connected pattern accounts for all eight pixels surrounding the target pixel, and a 4-connected pattern uses the four pixels to the north, south, east and west directions. The 4-connected region, henceforth called the mask, is used in the following proceedings.

The mask is applied to each 4-connected pixel, until a minimum search correlation value (CV) is found in any of the five (5) underlying subset centroids, greater than or equal to cv_{search}. Once located, the mask does not to continue to move in a "random" pattern, because the cv_{search} value indicates that the path of highest correlation may have been found. The next subset center is set as the pixel of highest correlation value, found in one of the four branches of the mask. This procedure is visualized in Fig. 2.2, and results in the search pursuing a non-straight path of ever-increasing correlation. After cv_{search} has been reached, it is no longer considered the threshold of competency, and the next correlation values found are then compared to the previous CV, cv_{last}. The process terminates when the path no longer produces any correlations greater than cv_{last}.

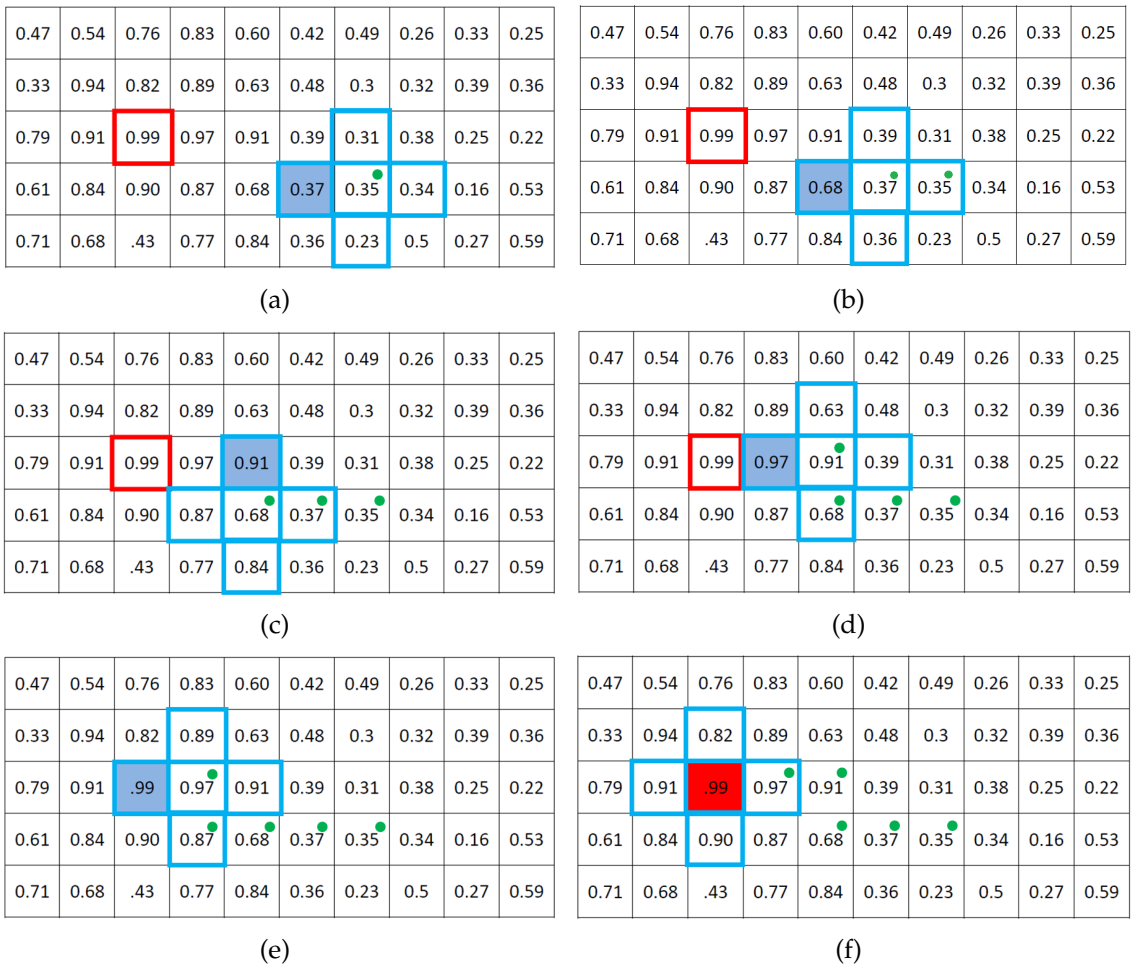


Figure 2.2: In this example of coefficient-guided correlation, coefficients are pre-calculated. The series of moves depicted is the path taken for the search scheme ((a) beginning, (f) ending) to locate the highest coefficient coordinate. The shaded area indicates the highest correlation found under the 4-connected mask, and dots indicate the path taken.

It is apparent that this process is susceptible to false matches. As seen in the top right quadrant of Fig. 2.1, there exist other areas in the image that resemble the pattern, and therefore contain enough comparable grayscale data to produce a coefficient that tricks the process into terminating at a false value. This issue is resolved by comparing the terminal coefficient to a super coefficient, cv_{min}, and only allowing the process to terminate if this threshold is reached.

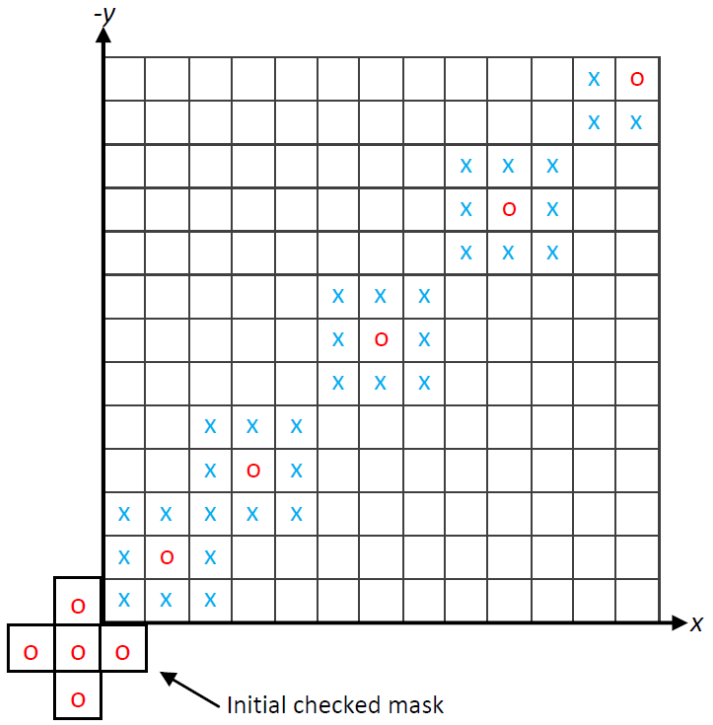
This design of max-coefficient walking allows the algorithm to be judicious in the calculations it makes. Instead of checking point-by-point in a generalized region, the search is self-guided while also requiring no intervention from the user.

Minimizing computations by implicit coefficient calculations As stated, the driving factor in processor time is the calculation of C_{zncc}. Any mechanism that minimizes this computation will have a positive effect on efficiency. This can be neatly achieved by using the supposition stated in the previous section, that correlation increases towards the local maximum.

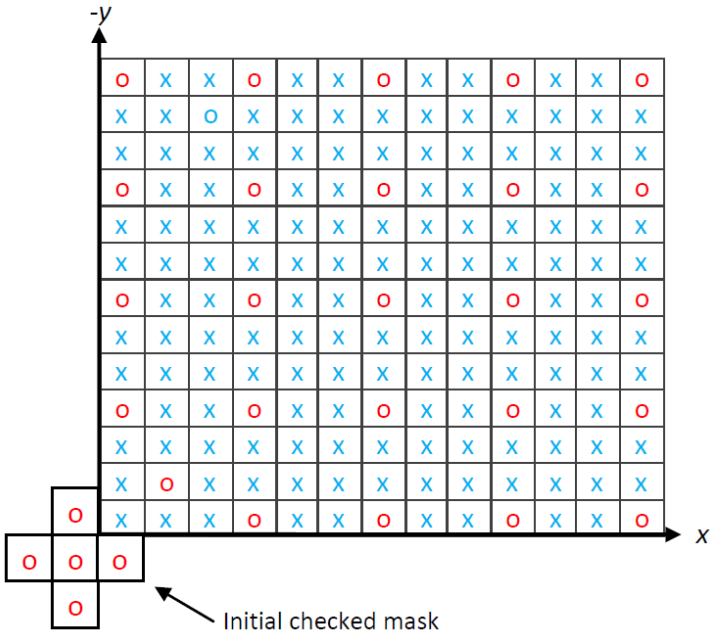
Regard 2.3 (a). Assuming that a high correlation will be adjacent to another high correlation, it is apparent that one needs to check only one out of every nine (1/9) pixels. Each coordinate surrounding the explicitly calculated pixel is implicitly checked due to the fact that if the calculated C_{zncc} is low, the surrounding pixels are presumed to exhibit low correlations as well, and therefore do not need to be calculated at all. This results in the algorithm being justified in stepping three (3) pixels at a time in each direction instead of one (1). The full search field for a 13×13 window is shown in Fig. 2.3, where only every third coordinate is calculated (the first point is shifted a step from the initial checked mask). The necessary calculations therefore undergo a sizeable reduction from $13 \cdot 13 = 169$, to 25.

Through gathered data, this approach produces good results, shown in Chapter 4. However, there can arise circumstances where the sample may have undergone deformation of large magnitude, and therefore the correlation coefficients will all be accordingly lowered. Even though the principle of increasing correlation still holds in these situations, the coefficients may be low enough that they do not meet certain correlation thresholds, and therefore do not do a good job of implicitly checking their neighbors. Provisions are made in the algorithm for this, outlined in upcoming sections.

Adaptation using directional recursion The last major feature of the search algorithm uses an adaptive scheme to narrow the geometric search window. The idea of "cheating" the deformation and knowing in which direction to check for the highest correlation can have significant increases on efficiency; however, as minimum user input is desired, there is no directional information ahead of time.



(a)



(b)

Figure 2.3: Map of coordinates showing explicit calculations ("o") and implicit calculations ("x"). (a): points showing 8-connected implicit checking; (b): full quadrant search map.

The algorithm uses a recursive scheme to check preceding displacement data to predict in what distance and direction the next maximum correlated pixel will be located, achieved in three steps:

1. Obtain the preceding reference subset coordinate displacements in the (x, y) directions from the prior image data set.
2. Calculate the value of the slope of the displacements.
3. Input the previous (x, y) data and slope into a function to determine which quadrant to use.

In step three, the quadrant check regions are seen in Fig. 2.5. The entire map can be split into nine quadrants that extensively cover any direction displacement may take. The justification for splitting the quadrants in such a fashion can be observed in Fig. 2.4. As often happens in lab tensile tests, displacement vectors often fall close to the major axes. If the algorithm were to have only four quadrants to choose from, it is apparent that fluctuations in the deformation direction would cause the search to miss the correlation completely. The displacement vector will then always fall into two of the regions, and the algorithm chooses the region that contains the greatest equal distribution of points on either side of the vector. When there is no previous displacement data, either in the first image in the set or when zero or low deformation has taken place, the ninth region *null* is used. This region contains every point in the image, so by definition is non-directional.

Secondary efficiency and accuracy mechanisms The preceding three sections describe the major underpinnings of the search algorithm and account for most of its success. There are several other ways that efficiency and accuracy are increased, resulting in minor yet important contributions:

1. **Adjust maximum search distance** The maximum search distance is set with a default value, as well as user-adjustable. The value can be varied up to 25% of the maximum image dimension. In the case that the algorithm fails to correlate a subset, it will increase the search distance in 2 increments: $distance \cdot 2$ and $distance \cdot 4$, resulting in step increases of 50% and 100%, respectively. If one of these increases succeeds in finding a correlation, it is assumed the deformation is speeding up and that this will hold true for the proceeding images, so the search distance is held at this value for the rest of the analysis. Otherwise, it is reversed to the original value. This adaptive adjustment makes sure that the search is covering relatively as little ground as possible, increasing overall speed.

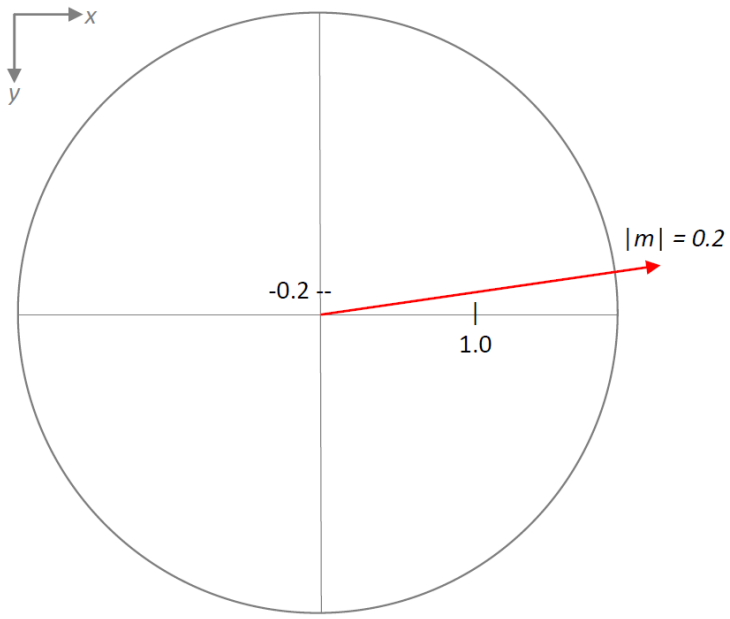
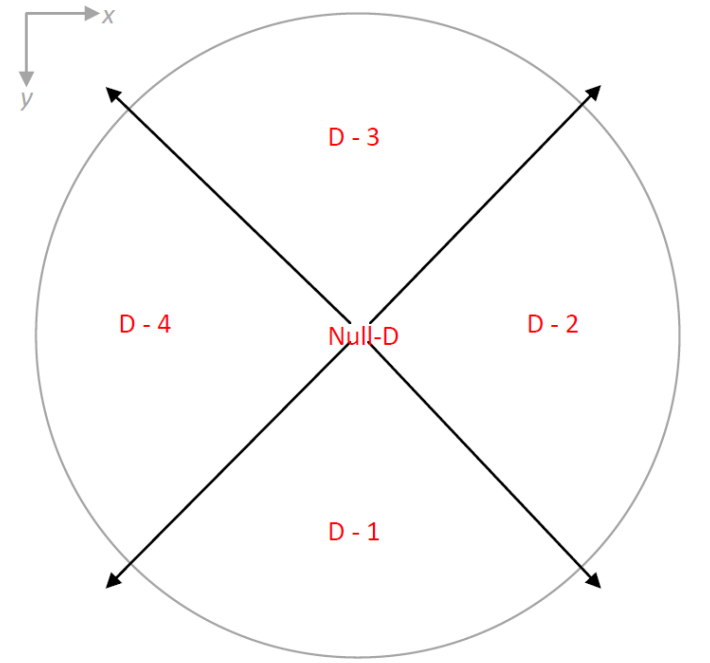


Figure 2.4: Arbitrary displacement vector showing pixel displacement slope and location among the four standard quadrants.

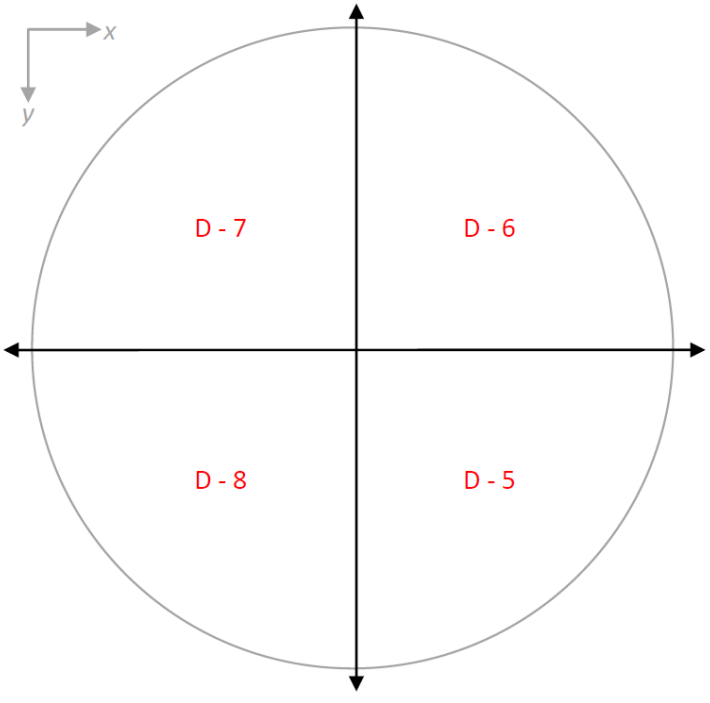
2. **Decrease step size** The step size of three (3) using implicit calculations is effective in a vast majority of cases. However, as mentioned previously, there may be times when the deformation characteristics result in an overall low pattern of C_{zncc} maximums, so this step size may not be adequate. If failure to locate a coefficient above the threshold occurs, the search restarts with a reduction of step size by one (1), so that accuracy is made sure not to suffer at the expense of efficiency gains.

3. **Predict future location** The prediction of future location gives very modest gains in efficiency, and is not something that is heavily relied on. Yet in cases of large discontinuities in subset displacement between images, predicting the displacement magnitude can reduce the search time considerably, so is employed as a first measure. To do this, the algorithm peeks back at previous displacements for that index, and begins building the search list by placing those values first in the queue. Therefore, a somewhat inaccurate prediction is made that has the possibility of helping efficiency, yet does not negatively affect accuracy in any manner. The reason this cannot be relied on further is because it is up to the discretion of the user which images to submit for analysis, so a large displacement may be followed by small, with no way of knowing this has occurred.

Incremental reference image update Measuring large deformation can be difficult using correlation functions, due to something called decorrelation effect.



(a)



(b)

Figure 2.5: (a) Search quadrants centering on the primary axes (90°), including null (all quadrants); (b) Search quadrants centering on secondary axes (45°).

during deformation, the specimen not only translates the speckle pattern, but deforms it as well, as seen clearly in Figs. 2.6 (a) and (b). The speckle pattern, notably on the right side next to the circular cutout, has undergone considerable transformation. This is what is known as decorrelation effect.

To combat this, the reference image can be moved to a new location closer in the image sequence to the highly deformed image. After this is done, instead of correlating to the original image (which the deformed image may no longer resemble), the new reference target is an image which more closely represents the new geometry. This is a simple process, only involving keeping a record of the image number that the reference was updated to. Post-processing then takes care of backing out the total magnitude of the displacement for images analyzed after the update. This can be done any number of times, providing an effective way to manage decorrelation effect, and still successfully analyze images that have undergone serious transformations.

However, there is warrant to perform reference image updating as few times as possible due to error propagation. In post-processing, the displacement magnitudes are calculated by adding the distance the subset traveled, which without updating would be the original location. However, if updating has occurred, any error in displacement calculation seen up to and including the previous image is added to the next image. With repeated updates, propagation of error results.

Therefore, this algorithm strives to take every measure possible and exhausts all other methods before recording a bad correlation incidence. Defaulted into the application, the minimum number of bad correlations allowed to transpire is 0.15% of total pixels in the index before updating. This can also be modified by the user to allow more or less updating, depending on the needs. Ideally, this number should suffice when all other methods of search are utilized.

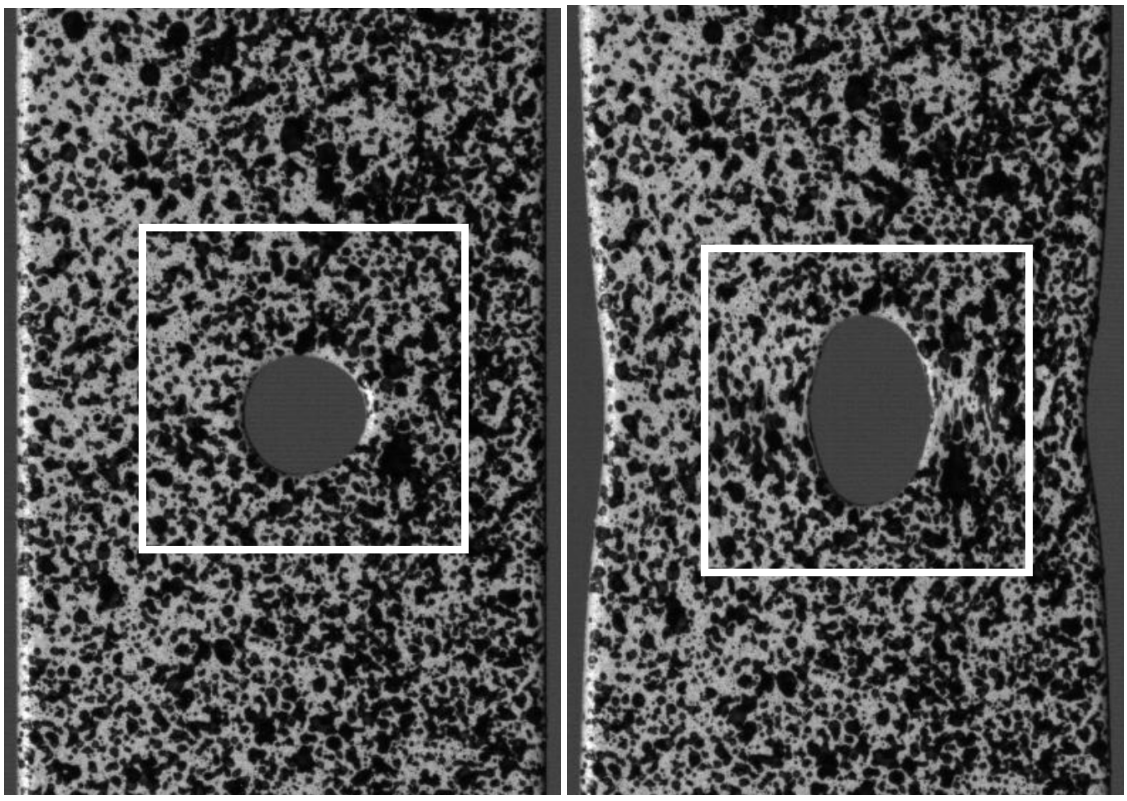
2.1.3 Algorithm processes

This section will elucidate the major steps taken and processes used in the correlation algorithm. There are two major processes that form the foundation of the algorithm:

1. Regular search
2. Exhaustive search

Each of these is comprised of a number of minor processes, with the main structure of the algorithm as follows:

1. Bring in a new image.
2. Obtain subset to analyze from index.



(a) (b)

Figure 2.6: Decorrelation effect showing speckle pattern deformation; (a) is reference image, and (b) is deformed image. The correlation may fail to produce viable correlation values when the pattern deforms.

3. Begin regular search.
4. If regular search is successful, loop back to step two; else, enter exhaustive search.
5. Re-enter regular search with either failure or success messages.
6. Continue in regular search according to message content.
7. Loop back to step two.
8. All coordinates are analyzed, restart process anew at step one with a new image.

Regular search process Regular search uses an identical method on two different circumstances. The first is that the subset centroid is a new value from the reference list, and the second being that regular search has been re-entered from exhaustive search, and has a new subset centroid to analyze. Either way, the techniques of coefficient-guided search explained previously are used, and the process visualized in the flowchart in Fig. 2.7. The steps are summarized as follows:

1. Input previous centroid coordinates where the maximum correlation occurred, as well as the data for the subset in the reference image.
2. Build the regular search list, comprised of the four (4) subset centroids underlying the arms of the 4-connected mask.
3. Calculate each C_{zncc} and test against a) CV_{search} and b) the previously calculated maximum, if it exists.
4. If the new value is higher than the old value, relocate the mask to center on the new value, and loop back to step two.
5. If the correlation has not increased, or else has increased yet does not exceed CV_{min}, exit regular search into exhaustive search.
6. Read message from exhaustive search, which has two outcomes:
   (a) The exhaustive search scheme has located a value exceeding $CV_{confirm}$, therefore loop back to step two using this new subset centroid.
   (b) No suitable correlation has been found; increment the bad correlation counter.
7. Exit to main function.

The main function performs the task of tracking the bad correlation count and identifying whether it is time to update the reference image. It should be noted that regular search only allows exhaustive search to be entered a single time. Otherwise, a perpetual loop would form if no good correlation was found. In general, regular search does the job without needing to enter exhaustive search if the deformation is somewhere in the neighborhood of 0-4 pixels between images, meaning that decorrelation effect is not pronounced and the algorithm can self-guide to the absolute peak.

Exhaustive search process Once the exhaustive search routine is entered, it stays there until one of two following things happens: either $CV_{confirm}$ is exceeded, or the search list is exhausted without success. $CV_{confirm}$ is a value higher than CV_{search} and lower than CV_{min}, and serves to guard against false positives from local correlation peaks. Visualized in Fig. 2.8, the procedure is as follows:

1. Input to exhaustive search is the same as input to regular search: centroid coordinates of the maximum correlation of the corresponding subset in the last image. Search direction is also calculated here, as explained in previous sections.
2. The search list is then built using directional, distance, and step size inputs, and is comprised of the (x, y) coordinates of each subset centroid that should be correlated. These points are variable and depend on how many iterations the routine has gone through (how many failures for that subset have occurred).
3. The first point in the search list is correlated singularly (instead of as a 4-connected mask), and removed from the queue. If the point is above CV_{search}, the 4-connected mask is then applied (similarly to the regular process) in an effort to reach $CV_{confirm}$. If this threshold is reached, the process immediately exits to the regular routine, using this subset coordinate as the new input.
4. If the indicated threshold is not reached, this means a false positive has been successfully overcome, and all the previous calculations and movements are ignored. The algorithm then moves to the next point in the list by returning to step three.
5. If this point has been reached, then one of three things will happen before re-entering step two, depending on the iteration, and whether a search distance increase is available:
   (a) Step size is decreased
   (b) Maximum search distance is increased
   (c) Search direction is set to *null*

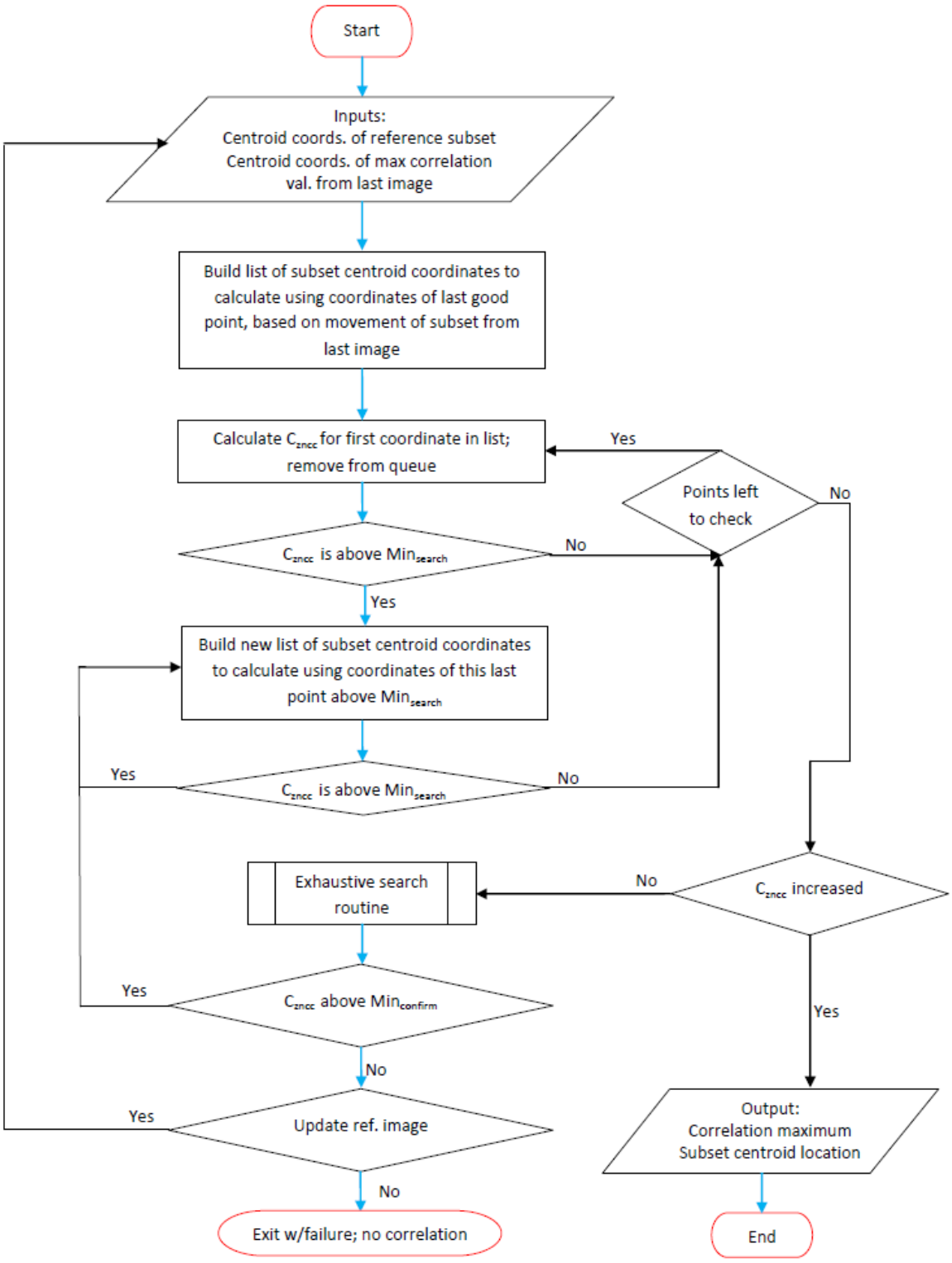


Figure 2.7: Flowchart of broad-scheme correlation search process.

6. At this point, the routine has failed to find a subset coordinate with $CV > CV_{confirm}$. Exhaustive search is exited to regular search with failure message.

With these processes, an integer-pixel location showing good correlation has been reached, and the next step, if desired, is to estimate the pixel location more precisely to sub-pixel coordinates.

2.2 Integer-pixel search routine for PIV

The PIV program uses a purely mathematical, frequency-domain-based approach for image registration, due to the random and out-of-plane motion of fluid particles which causes subset-based search routines to be inefficient.

2.2.1 Phase correlation

Phase correlation is a method of obtaining the relative translation between images by using the discrete Fourier transform (DFT) to represent grayscale image pixel information in the frequency-domain. Relating this to image correlation for PIV analysis, the images are segmented into a grid of virtual Fourier windows, which are their own discrete images. Phase correlation is performed on each window to determine its displacement between images. After performing the correlation on all grid indexes, a velocity field in pixels/image pair results.

Mathematical basis The fast Fourier transform (FFT) is used in phase correlation due to its increased computational efficiency over the standard DFT. Phase correlation is perfomed by the following steps, from Druckmullerova [2011]:

1. Calculate the FFT for the reference and deformed window.
2. Calculate the cross-power spectral density by multiplying the complex conjugate of the deformed FFT by the reference window FFT (element-wise).
3. Normalize the result element-wise.
4. Apply the inverse FFT (iFFT) to the normalized product.
5. Determine the location of the peak value in the iFFT grid using the real part of the results.

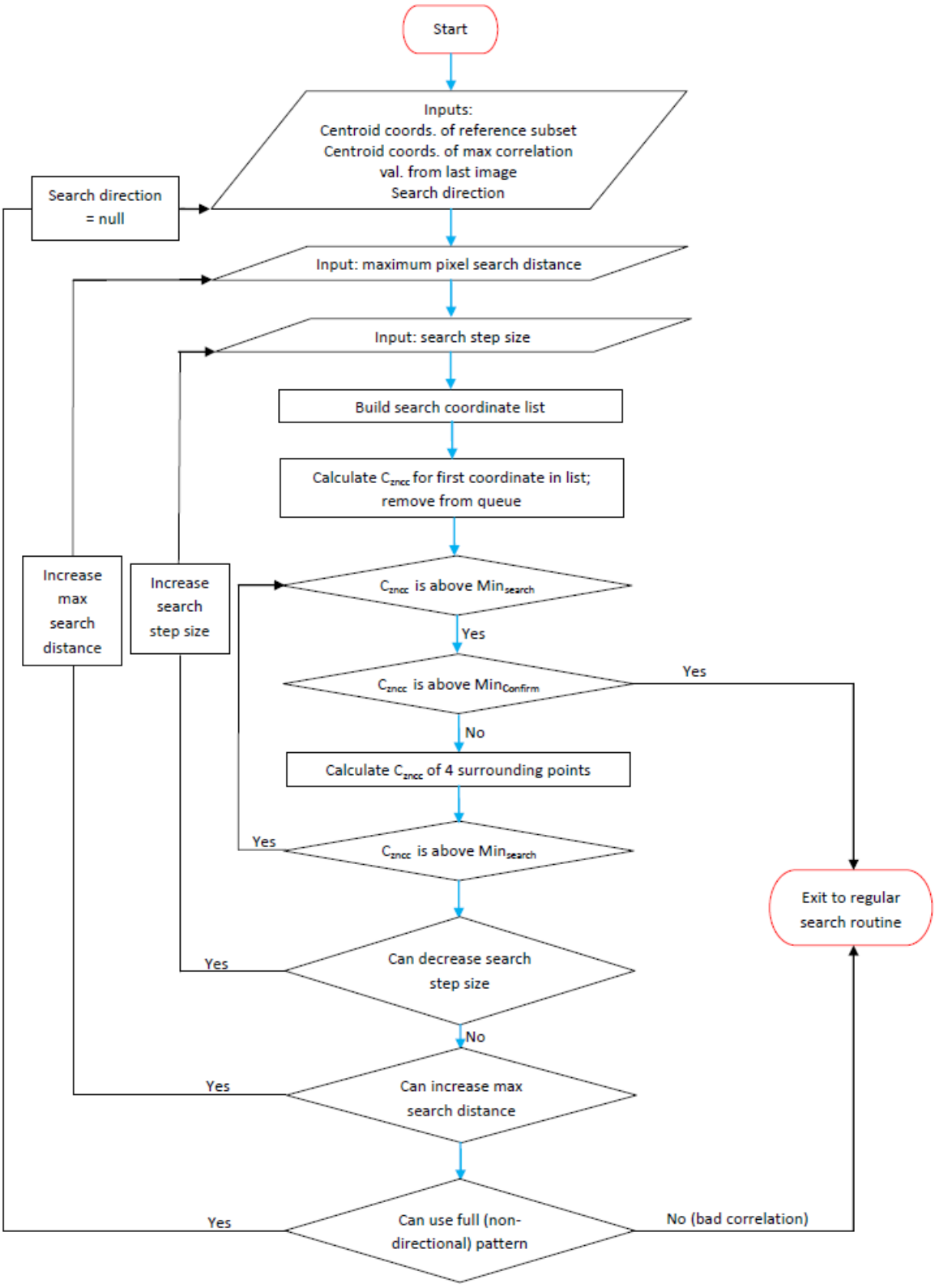


Figure 2.8: Flowchart of exhaustive search scheme correlation process.

The location of the peak in the iFFT is the relative motion of the deformed window to the reference window. The complete phase correlation calculation:

$$\mathscr{G}_0 = \mathscr{F}(g_0)\,, \mathscr{G}_1 = \quad \mathscr{F}(g_1) \tag{2.1}$$

$$p(x,y) = \mathscr{F}^{-1}\frac{\mathscr{G}_0 \otimes \mathscr{G}_1^*}{|\mathscr{G}_0 \otimes \mathscr{G}_1^*|} \tag{2.2}$$

where $\mathscr{G}_0$ and $\mathscr{G}_1$ are the FFTs of the reference window and deformed window, respectively, $\mathscr{G}_1^*$ is the complex conjugate of the deformed window FFT, and $p(x,y)$ are the pixel coordinates of the point in the deformed image showing peak correlation.

The process of phase correlation is simple, negating the need for a subset-based search algorithm. However, the computational cost of a single FFT calculation is significantly higher than a zero-mean cross-correlation calculation. The processor time to run a phase correlation for a single window is about triple the time it takes to calculate a cross-correlation. Therefore, since the motion of particles in DIC is relatively consistent and predictable, it is much less efficient and time-consuming to use the phase correlation approach. In PIV, the opposite is true and speed gains using the frequency-domain approach are substantial.

To verify the phase correlation result for each Fourier window location, a single cross-correlation calculation is performed to establish a confidence coefficient. If the coefficient is too low (user-adjustable minimum coefficient), no translation is recorded for that window index. In PIV analysis, the incidence of poorly correlated points is high, especially in the background regions.

2.3 Multi-threading of the search process

The operation of both phase correlation and subset-based correlation starts with an indexing of all available points within the region of interest. The result is a multi-row index of the row/ column image coordinates of every subset centroid in which all pixels in the subset fall completely within the ROI. Each entry in the index is independent of all other entries, meaning that the program may commence search operations on neighboring entries in the index without interference.

To operate a simultaneous search, the index is split into a number of sections, which are entered into their own computational threads in the program. Each deformed image is entered simultaneously, where the index-splitting operation is performed. The threads work independently to correlate all the points in each assigned section of index, then wait for the other threads to complete before exiting and reporting the results to the main thread. The number of sections into which the index is split depends on the number of cores available in the processor; the maximum number of threads is equal to the number of available cores,

# Threads	% Faster
1	—
2	56.3
3	91.1
4	100.36

Table 2.1: The program makes significant speed improvements by multi-threading the correlation process.

and the user also has the option to reduce the amount cores used. Table 2.3 shows a series of digital image correlation runs quantifying the speed increase gained by multi-threading the application. The number of runs for each thread combination is 10, with the results averaged together and proportionally compared against the baseline of a single thread. Doubling the number of threads (from one) increases the speed by approximately 1.5, and by using four (4) threads, the speed of the correlation is doubled.

2.4 Sub-pixel search routines

Integer-pixel search methods are the first step in visual image correlation. In a quick visual analysis, integer-pixel resolution may be a sufficient outcome. However, in teaching or research environments, sub-pixel accuracy may be a requirement for functions such as validating finite-element strain analysis or mapping the stress-strain curve in a sub-region of a sample. Integer resolution may not provide enough data to achieve a significant result, especially when stress is provided on a single axis and the sample is of uniform dimensions. The analysis may indicate zero secondary axial deformation, when in fact the deformation could have been anywhere on the order of zero to one pixel.

Therefore, the need arises to identify the location of the "true" movement of the subset, mapped to a location between pixels. In visual image correlation, typically two orders of deformation are examined: the first is zero-order movement, which is purely translational. Affine warp is not taken into consideration, as all higher order data is removed from consideration. The subset centroid, $(2M+1, 2M+1)$, is considered to translate from (x_0, y_0) to (x', y') by

$$x' = x_0 + u + \Delta u \tag{2.3}$$

$$y' = y_0 + v + \Delta v \tag{2.4}$$

where u, v are the integer-pixel translational components, and $\Delta x, \Delta y$ are the sub-pixel movement components. First-order deformation takes into consideration the displacement gradients,

$$x' = x + u + \Delta u + u_x \Delta x + u_y \Delta y \tag{2.5}$$
$$y' = y + v + \Delta v + v_x \Delta x + v_y \Delta y \tag{2.6}$$

where u_x, u_y, v_x, and v_y are the first-order displacement gradients of the subset. A more complicated second-order shape function can also be derived to further quantify the displacement [Lu and Cary, 2000], however that is not discussed here as it has no relevant purpose to this application.

2.4.1 Correlation coefficient curve-fitting methods

The first subpixel techniques discussed are curve-fitting routines. The goal of a curve-fitting function is to find the point at which a continuous function is created that optimally fits correlation coefficient data. Using the points surrounding the best-correlated integer point and mathematically describing a surface containing these points, the best matching peak may be obtained by maximizing the function in the area corresponding to the real location. In essence, the goal is to shift the coordinates of the peak correlation to somewhere in between the centroids of the pixels, i.e. subpixel accuracy.

No single function can perfectly describe such a surface, so a number of different techniques and approximations can be used based on the application's speed and accuracy requirements. In parabolic and Gaussian fitting, the shape is assumed to fit two orthogonal curves that often approach a bell shape, so are defined using parabolic and Gaussian surfaces using information from four immediate surrounding points (four-connected approach). Bi-quadratic Laplace uses an iterative process to fit the surface to an eight-connected pattern of correlation data.

Each process comes with deficiencies as well as benefits for this application. In previous work it has been shown that the parabolic and Gaussian fitting methods may result in a high error if high shear strain is applied to the specimen [Debella-Gilo and Kaab, 2011, Nobach and Honkanen, 2005]. However, since each curve requires only the help of two neighboring correlation values (5 total correlation calculations), the methods are computationally efficient and therefore speedy. The Lagrange formulation requires more inputs and internal calculations so is slower, however the error bars are diminished.

Parabolic fitting In parabolic surface fitting, the Δx and Δy components are determined independently, using the assumption that the real correlation peak in each direction lies on a curve containing 3 points. The curve is defined using a one-dimensional quadratic function, with the peak located somewhere along

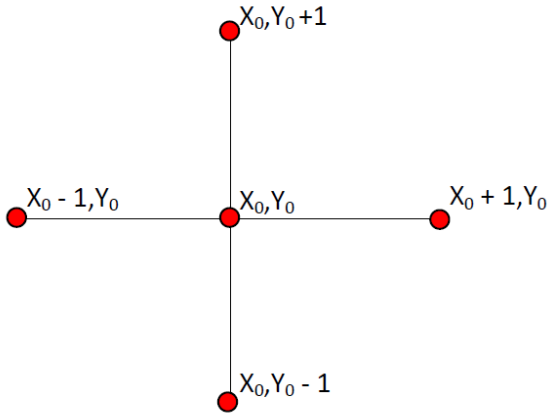


Figure 2.9: Four-connected placement of correlation points for Gaussian and parabolic surface fitting. The subpixel displacements in *x,y* are independently calculated along each dimension.

that curve. Using the 4-connected scheme depicted in Fig. 2.9, the two correlation coefficients of the neighbors in the x and y directions are computed, then fit independently with curves.

The center, best-matching pixel has two neighbors in either the x or y direction, $|x_0 - 1|$ and $|x_0 + 1|$, or $|y_0 - 1|$ and $|y_0 + 1|$. To find the sub-pixel distance $\Delta x, \Delta y$ in each direction, a parabolic curve is defined using the three points in the specified dimension that intersect them, and then computed to attain the maximum [Debella-Gilo and Kaab, 2011]:

$$\Delta X = \frac{\rho(X_0 - 1, Y_0) - \rho(X_0 + 1, Y_0)}{2\rho(X_0 - 1, Y_0) - 4\rho(X_0, Y_0) + 2\rho(X_0 + 1, Y_0)} \tag{2.7}$$

$$\Delta Y = \frac{\rho(X_0, Y_0 - 1) - \rho(X_0, Y_0 + 1)}{2\rho(X_0, Y_0 - 1) - 4\rho(X_0, Y_0) + 2\rho(X_0, Y_0 + 1)} \tag{2.8}$$

where ρ is the correlation coefficient at that index. The results are therefore separable, and are added to the integer (X_0, Y_0) coordinates to obtain the true location of the best correlated point.

Gaussian fitting The Gaussian surface is computed much like parabolic, except the surface is assumed to follow that of a 2D Gaussian function. The maximum location is found by fitting a second-order polynomial to the maximum sample

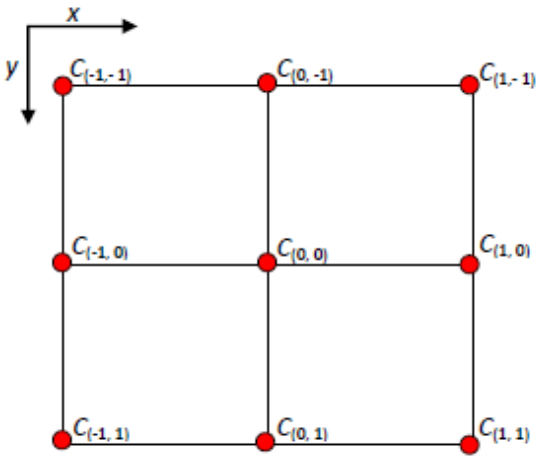


Figure 2.10: Index map of correlation points for the bi-quadratic Langrange formulation.

logarithm. The formulation uses a four-connected index, and each dimension is again orthogonal and separable [Debella-Gilo and Kaab, 2011]:

$$\Delta X = \frac{\ln(\rho(X_0 - 1, Y_0)) - \ln(\rho(X_0 + 1, Y_0))}{2\ln(\rho(X_0 - 1, Y_0)) - 4\ln(\rho(X_0, Y_0)) + 2\ln(\rho(X_0 + 1, Y_0))} \tag{2.9}$$

$$\Delta Y = \frac{\ln(\rho(X_0, Y_0 - 1)) - \ln(\rho(X_0, Y_0 + 1))}{2\ln(\rho(X_0, Y_0 - 1)) - 4\ln(\rho(X_0, Y_0)) + 2\ln(\rho(X_0, Y_0 + 1))} \tag{2.10}$$

Bi-quadratic Lagrange fitting The bi-quadratic Lagrange method fits an optimal surface to the best integer matching point and the eight surrounding points using their correlation values, shown in Fig. 2.10. Indexing is done according to the center pixel, so distances always fall in the range of [-1, 1]. An iterative process is used to approximate the point of maximum correlation at a subpixel location [Pan et al., 2012].

The Lagrange surface in bi-quadratic form is as follows:

$$\hat{C} = a_1x^2y^2 + a_2x^2y + a^3x^2 + a_4xy^2 + a_5xy + a6x + a_7y^2 + a_8y + a_9 \tag{2.11}$$

where x, y are the indexes of each correlated point. From this it is apparent that the system is of nine unknown coefficients that correspond to the surface function that passes through each discrete point. The nine linear equations can efficiently be evaluated in matrix form by the following procedure:

$$L\vec{a} = \vec{s} \tag{2.12}$$

where

$$L = \begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 1 \\ 1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 & -1 & 1 & 1 & -1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\vec{s} = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \\ s_8 \\ s_9 \end{bmatrix} = \begin{bmatrix} C_{(-1,-1)} \\ C_{(-1,\ 0)} \\ C_{(-1,\ 1)} \\ C_{(0,\ -1)} \\ C_{(0,\ \ 0)} \\ C_{(0,\ \ 1)} \\ C_{(1,\ -1)} \\ C_{(1,\ \ 0)} \\ C_{(1,\ \ 1)} \end{bmatrix}, \text{ and } \vec{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \\ a_9 \end{bmatrix}$$

To solve (2.12) and isolate the $\vec{a}$ vector using these identities, $\vec{s}$ must be multiplied with the inverse of L, as so:

$$\vec{a} = L^{-1}\vec{s} \tag{2.13}$$

Since L is always a 9×9 matrix using an 8-connected point pattern, it is square (full rank) , never singular, and therefore fully invertible. The inverse of L can be pre-assembled in code to save processing time, since the distances involved will always be integer pixel values in the region $[-1, 1]$:

$$L^{-1} = \begin{bmatrix} 0.25 & -0.5 & 0.25 & -0.5 & 1 & -0.5 & 0.25 & -0.5 & 0.25 \\ -0.25 & 0 & 0.25 & 0.5 & 0 & -0.5 & -0.25 & 0 & 0.25 \\ 0 & 0.5 & 0 & 0 & -1 & 0 & 0 & 0.5 & 0 \\ -0.25 & 0.5 & -0.25 & 0 & 0 & 0 & 0.25 & -0.5 & 0.25 \\ 0.25 & 0 & -0.25 & 0 & 0 & 0 & -0.25 & 0 & 0.25 \\ 0 & -0.5 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 & -1 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.5 & 0 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.14}$$

And multiplying the $\vec{s}$ vector by this inverse, $\vec{a}$ vector coefficients are now available:

$$\vec{a} = \begin{bmatrix} 0.25s_1 - 0.5s_2 + 0.25s_3 - 0.5s_4 + s_5 - 0.5s_6 + 0.25s_7 - 0.5s_8 + 0.25s_9 \\ -0.25s_1 + 0.25s_3 + 0.5s_4 - 0.5s_6 - 0.25s_7 + 0.25s_9 \\ 0.5s_2 - s_5 + 0.5s_8 \\ -0.25s_1 + 0.5s_2 - 0.25s_3 + 0.25s_7 - 0.5s_8 + 0.25s_9 \\ 0.25s_1 - 0.25s_3 - 0.25s_7 + 0.25s_9 \\ 0.5s_2 + 0.5s_8 \\ 0.5s_4 - s_5 + 0.5s_6 \\ -0.5s_4 + 0.5s_6 \\ s_5 \end{bmatrix} \tag{2.15}$$

recalling that s_n are correlation coefficients. To find the peak of the correlation surface, (x, y), for these 8-connected points, a zero-slope solution is sought where the surface has a zero (or close enough to it) derivative with respect to x and y. For this, a Newton iterative scheme (also known as the Newton-Rhapson method) is used to successively hone in on a sufficiently close approximation (within a specified tolerance) of the solution, as follows from Bruck et al. [1989]:

$$\vec{p}^{\,n+1} = \vec{p}^{\,n} - J^{-1}(\vec{p}^{\,n})F(\vec{p}^{\,n}) \tag{2.16}$$

where

$$\vec{p}^{\,n} = [x^n \; y^n]^T \tag{2.17}$$

are the x and y values of the guess at iteration n, and $\vec{p}^{\,n+1}$ is the solution for that iteration. The solution is considered converged when

$$|x^{n+1} - x| \leq \delta \text{ and } |y^{n+1} - y^n| \leq \delta, \tag{2.18}$$

where in this work the tolerance, $\delta = 0.05$, and note that $\vec{p}^{\,0} = [0 \; 0]^T$, implying the center pixel is the optimal starting solution. From (2.16), $F(\vec{p})$ is the "residual" function from Newton's scheme, the partial derivative with respect to x and y of (2.12):

$$F(\vec{p}) = \begin{bmatrix} \frac{\partial \hat{C}(x,y)}{\partial x} \\ \frac{\partial \hat{C}(x,y)}{\partial y} \end{bmatrix} = \begin{bmatrix} 2a_1xy^2 + 2a_2xy + 2a_3y^2 + a_4y^2 + a_5y + a_6 \\ 2a_1x^2y + a_2x^2 + 2a_4xy + a_5x + 2a_7y + a_8 \end{bmatrix} \tag{2.19}$$

Also from (2.16), J is the Jacobian matrix:

$$J(\vec{p}) = \begin{bmatrix} \frac{\partial^2 \hat{C}(x,y)}{\partial^2 x} & \frac{\partial^2 \hat{C}(x,y)}{\partial x \partial y} \\ \frac{\partial^2 \hat{C}(x,y)}{\partial y \partial x} & \frac{\partial^2 \hat{C}(x,y)}{\partial^2 y} \end{bmatrix} \tag{2.20}$$

$$= \begin{bmatrix} 2a_1y^2 + 2a_2y + 2a_3 & 4a_1xy + 2a_2x + 2a_4y + a_5 \\ 4a_1xy + 2a_2x + 2a_4y + a_5 & 2a_1x^2 + 2a_4x + 2a_7 \end{bmatrix} \tag{2.21}$$

for the sake of simplicity, let the elements of the Jacobian be:

$$J(\vec{p}) = \begin{bmatrix} J_1 & J_2 \\ J_3 & J_4 \end{bmatrix}, \tag{2.22}$$

then

$$J^{-1}(\vec{p}) = \begin{bmatrix} \frac{J_4}{J_1J_4 - J_2J_3} & \frac{-J_2}{J_1J_4 - J_2J_3} \\ \frac{-J_3}{J_1J_4 - J_2J_3} & \frac{J_1}{J_1J_4 - J_2J_3} \end{bmatrix} \tag{2.23}$$

Implementation of the bi-quadratic Lagrange formulation is computationally straightforward when using the preceding matrix identities; first the surrounding eight points are given correlation values, the coefficients of (2.11) are found using (2.15), then (2.16) is iterated until the solution is sufficiently approximated. The costly part of the process is calculating the eight surrounding coefficients, whereas plugging those values into the pre-formed matrices takes little time. The solution usually converges in 1-2 iterations.

2.4.2 Gradient-based methods

Non-iterative spatial gradient method This model makes two assumptions about the specimen and environment. The first is that the subset is undergoing solely rigid body translation, i.e. exhibits zero-order deformation characteristics, the same assumptions that hold for the surface fitting methods. The second is that the gray-level intensity for each pixel is constant, therefore the only thing that changes between the deformed and reference subset is the location of each pixel.

For this method,

$$f(x_i, y_i) = g(x_i', y_i') \tag{2.24}$$

where

$$x' = x + u + \Delta x, \quad y' = y + v + \Delta y \tag{2.25}$$

states that the subset in the original location, f, undergoes a rigid-body translation to g. x and y are the original starting locations of the subset centroid, u and v are the integer-pixel displacement components in the x and y directions respectively (found from a previous coarse search), and Δx and Δy are the sub-pixel displacement components in question [Pan et al., 2012].

Neglecting the higher-order terms of (2.24), a Taylor expansion yields

$$g(x_i', y_i') = g(x + u + \Delta x, y + v + \Delta y) \tag{2.26}$$
$$= g(x + u, y + v) + \Delta x \cdot g_x(x + u, y + v) + \Delta y \cdot g_y(x + u, y + v) \tag{2.27}$$

where g_x and g_y are the intensity gradients in the x and y directions. Gradients can be found a number of ways, primarily in image correlation by using a convolution operator. [Pan et al., 2012] suggest using a mask of

$$[1/12, -8/12, 0, 8/12, -1/12]$$

applied in the x and y directions, as it shortens the truncation error to $o(h^4)$, smaller than standard gradient operators such as the Sobel or Prewitt (note that the processing time however for decimal values is slightly higher than that for integer or shorts, and requires more memory).

In this method, solving for dx and dy requires that the sum of squared differences (SSD) correlation measure be maximized;

$$C_{SSD} = \sum_{i=-M}^{M} \sum_{j=-M}^{M} = (f(x_i, y_i) - g(x_i', y_i'))^2 \tag{2.28}$$

that is,

$$\frac{\partial C_{SSD}}{\partial(\Delta u)} = 0, \text{ and } \frac{\partial C_{SSD}}{\partial(\Delta v)} = 0$$

The boundaries, M, imply that the summations are taken over the entire subset. Inserting (2.27) into (2.28), the sub-pixel displacements are found by solving the system

$$\begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} =$$

$$\begin{bmatrix} \sum_{x=-M}^{M} \sum_{y=-M}^{M} (g_x \cdot g_y) & \sum_{x=-M}^{M} \sum_{y=-M}^{M} (g_y)^2 \\ \sum_{x=-M}^{M} \sum_{y=-M}^{M} (g_x)^2 & \sum_{x=-M}^{M} \sum_{y=-M}^{M} (g_x \cdot g_y) \end{bmatrix} \times \begin{bmatrix} \sum_{x=-M}^{M} \sum_{y=-M}^{M} [(f - g) \cdot g_x] \\ \sum_{x=-M}^{M} \sum_{y=-M}^{M} [(f - g) \cdot g_y] \end{bmatrix} \tag{2.29}$$

Rearranging and distributing these terms in a more code-friendly manner (each summation taken over the entire subset as before):

$$\Delta x = \frac{(\Sigma\Sigma(g_y)^2) \cdot (\Sigma\Sigma g_x(f-g)) - (\Sigma\Sigma g_x g_y) \cdot (\Sigma\Sigma g_y(f-g))}{(\Sigma\Sigma(g_x)^2) \cdot (\Sigma\Sigma(g_y)^2) - (\Sigma\Sigma g_x g_y)^2} \tag{2.30}$$

$$\Delta y = \frac{(\Sigma\Sigma(g_x)^2) \cdot (\Sigma\Sigma g_y(f-g)) - (\Sigma\Sigma g_x g_y) \cdot (\Sigma\Sigma g_x(f-g))}{(\Sigma\Sigma(g_x)^2) \cdot (\Sigma\Sigma(g_y)^2) - (\Sigma\Sigma g_x g_y)^2} \tag{2.31}$$

CHAPTER 3

PROGRAM OPERATION, OUTPUT AND APPLICATION

Several experiments were performed to present sample output from the program for both digital image correlation and particle image velocimetry. The images and data from the following experiments are also used to provide verification and validation of the program and algorithms in the following chapter.

3.1 Digital Image Correlation

Several experiments for digital image correlation validation were performed at New Mexico Tech using a materials testing machine provided by the Mechanical Engineering department. The test samples were machined in the department machine shop, and the following two experiments were chosen (arbitrarily) to present:

1. Low-carbon steel "dogbone" sample recorded with high-speed camera.
2. Aluminum "dogbone" sample recorded with an iPhone 6 camera

Each sample was prepared for imaging with a flat white background, followed by speckling with flat black spray paint, as described in Chapter 1.

3.1.1 Low-carbon steel dogbone test

Experimental setup For the following analysis, a dogbone sample appoximately 75mm x 25mm was machined from low-carbon steel, with 5 mm diameter necks and a 5 mm diameter center cutout. A Photron Fastcam Mini UX100 high-resolution, high-speed camera, coupled with a Nikon AF Micro Nikkor 60 f/2.8D lens were used to record images during the test. The sample was placed in an MTS Landmark servohydraulic materials tester (experimental setup seen in Fig. 3.1), and pulled from the anchor point at the bottom until separation occurred at a test rate of four (4) kiloNewtons per second (kN/sec), as seen in Fig. 3.2. 175 frames were selected from the image set up to the point where fracturing or separation appeared in the sample.

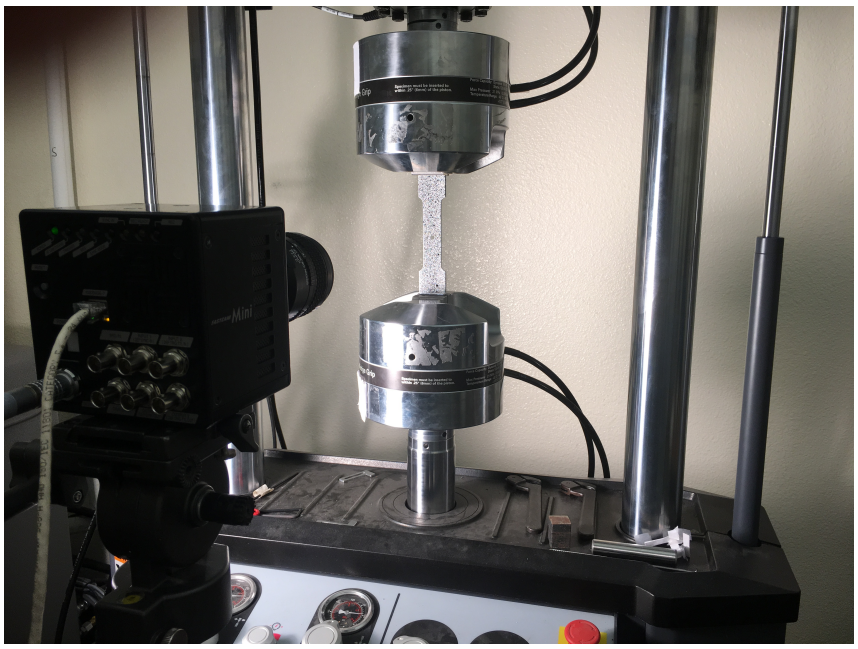
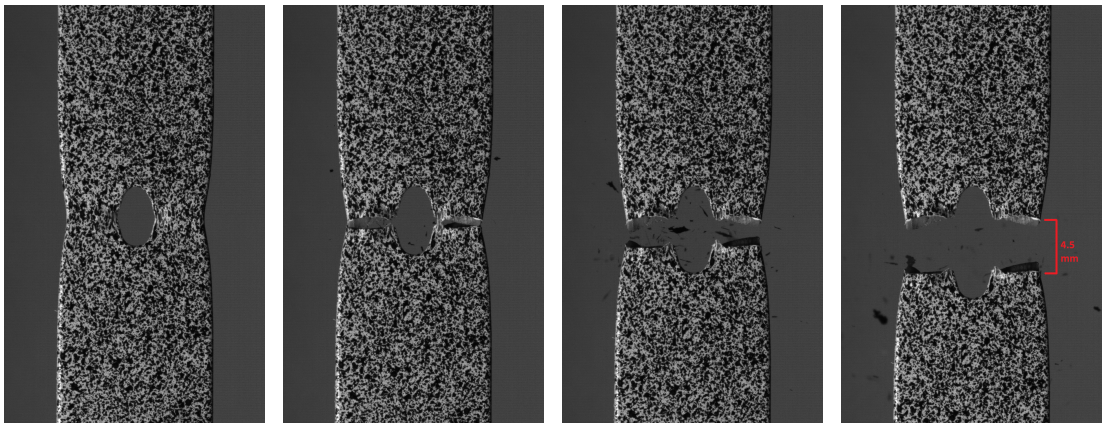


Figure 3.1: Experimental setup for digital image correlation analysis. The dog-bone sample is anchored from the bottom and top in the materials tester, with stress applied to the bottom as the anchor is pulled downward



(a) (b) (c) (d)

Figure 3.2: Fracture and separation sequence of test sample; time proceeds from image (a) - (d). (a) is the last image used in the experimental set.

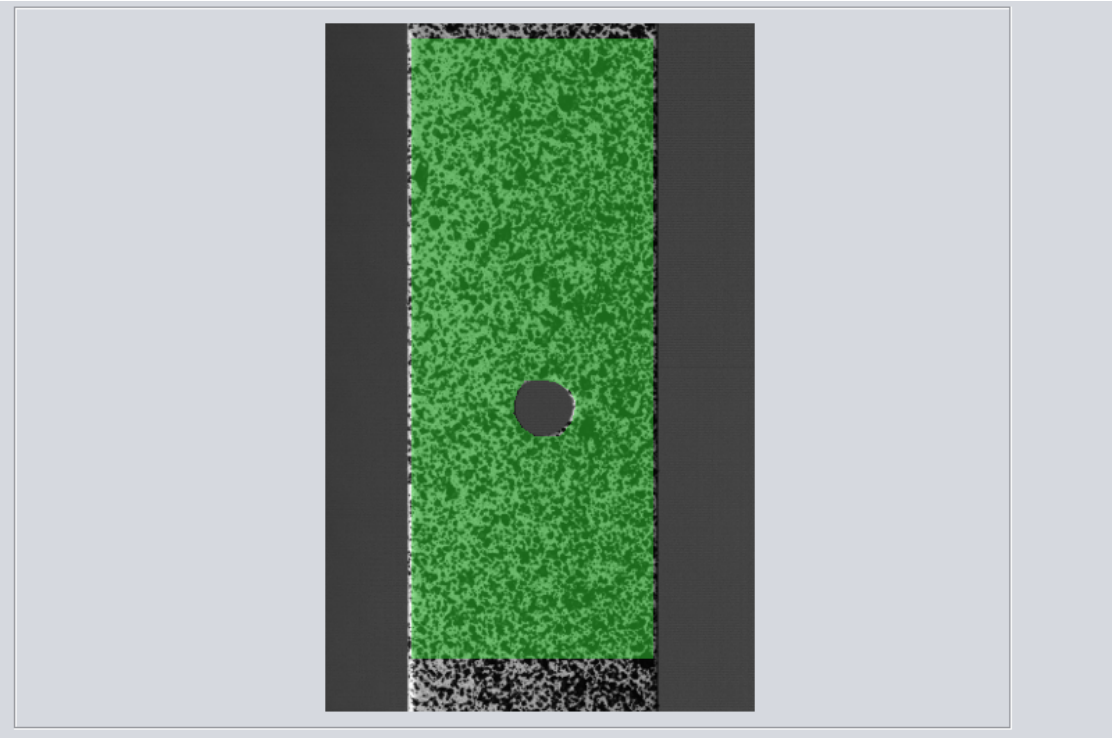


Figure 3.3: Region of interest selected for experimental analysis of mild-steel sample. The ROI is selected to include the parts of the sample not pulled out of frame by the stress test, as well as exclude background pixels.

Region of Interest In Fig. 3.3, the pixel mask (ROI) selected for this analysis is shown. The rectangular drawing tool provided by the program obtained the mask, with the center cutout clipped from the ROI using the free-form polygon selection tool. The lower coordinates of the selection are placed some distance from the bottom of the sample, as this type of sample-stretching event tends to drag the lower pixels out of frame.

Input parameters The kernel size chosen for this setup is 21, chosen after zooming on the sample in the image and selecting a representative speckle pattern. The step size is accordingly set at five (5), which gives a roughly 75% overlap of the kernel. The minimum correlation threshold was is the default value of 0.8, and image incrementing is selected with a bad correlation minimum count (the amount of correlation values below the correlation threshold) of 100. Parabolic subpixel resolution is also applied.

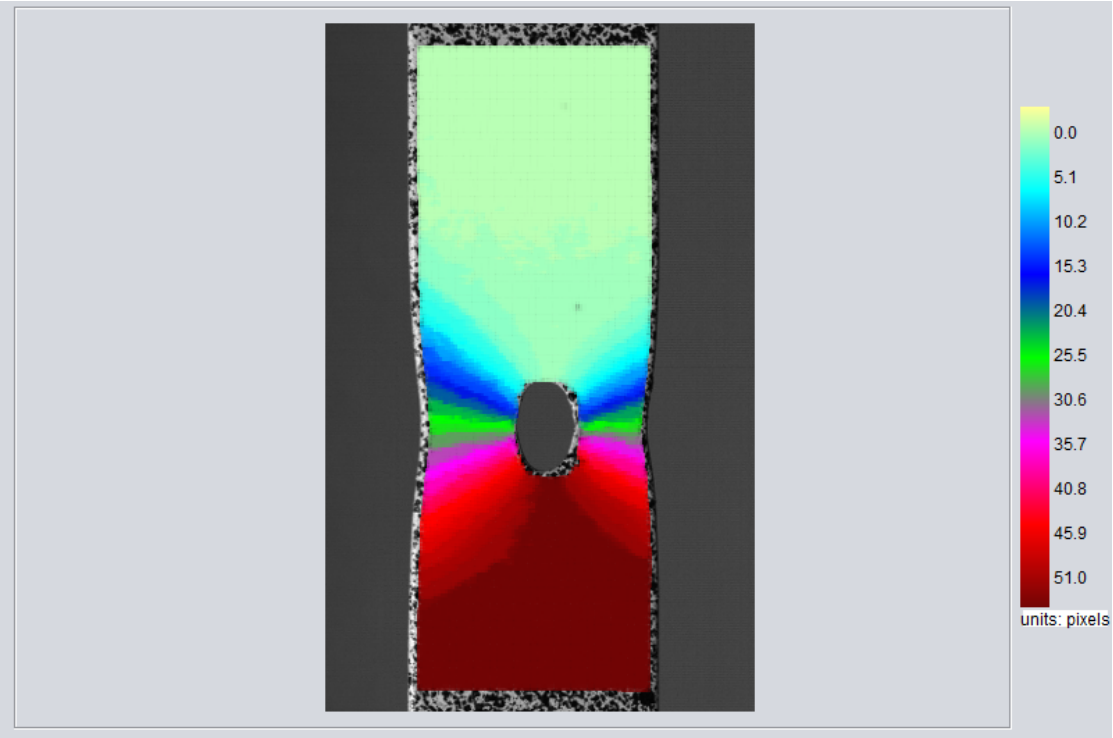
Output The images in Fig. 3.4 show the vertical and horizontal displacement of every fifth (equaling the step size) pixel in the selected ROI. Since a smooth image is desired to better view the deformation, a rectangle equal to the step size is filled with the desired color from the legend. From Fig. 3.4 (a), the deformation is greatest at the bottom of the ROI, and minimal at the top. This results from the MTS machine pulling vertically downwards from the bottom as the sample is held statically at the top. In the middle of the sample around the cutout, the deformation gradient is seen to steepen, as the stress on this area of the sample increases significantly as the steel is stretched apart.

In Fig. 3.4 (b), the upper-right quadrant of the output appears to be zero displacement. This is not strictly true if looking at subpixel detail, and is an artifact of the implementation of the legend in code, as well as a function of the choice to use incremental base image update in this analysis instance. The legend displays integer-level displacement values, and the result is that if the actual displacement is somewhere between 0 - 0.5 pixels, the legend will indicate zero displacement. For programming simplicity, the same integer contour map implementation is used when a subpixel analysis is performed.

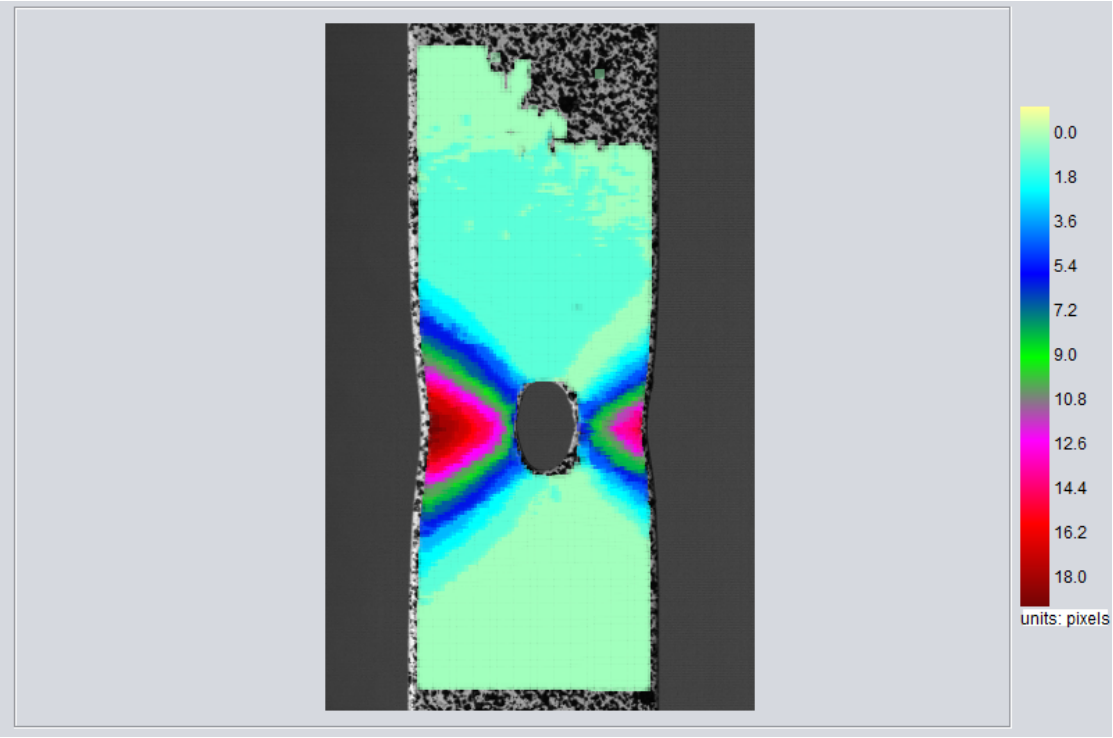
The choice of incremental base image update also contributes to the appearance of zero deformation in the x-axis in this particular sample. Fig. 3.5 shows a side-by-side of the same sample analyzed with all input parameters held constant except for incremental update. Fig. 3.5 (a), upper-right quadrant, shows no pixel displacement, however Fig. 3.5 (b) indicates that at least one (1) unit of displacement has occurred for the same region. The "true" value is somewhere between 0.5 - 1 pixel for that region (from subpixel analysis, which is available in the exported data). The reason for the discrepancy between the outputs is thus: since the region in question is of such low deformation, the amount of displacement from image to image is extremely low, somewhere less than 0.05 pixels in this sample. Therefore when an incremental base image update occurs, if the displacement at any pixel location between the last update and the current is less than 0.5, the reference location for that subset index is set back to zero.

3.1.2 Aluminum sample mobile phone test

Experimental setup From 3mm thick aluminum, a dogbone sample was machined at 95mm x 25mm, with 5mm necks and a 5mm center cutout. The MTS Landmark tensile testing machine was used again with a test rate of three (3) kN/Sec. An iPhone 6 recording in slow-motion video mode at 120 FPS was held in a custom rig, and a similar Fastcam Mini setup as used in the previous experiment was used for backup purposes, as seen in Fig. 3.6. The video frames were extracted using the tool included with this program, cut from the 0:14 - 0:17 second segment of the iPhone video. A total of 340 images were used for the analysis (three seconds at 120 FPS), selected for a segment of test before separation or visible fracturing of the sample occurred.

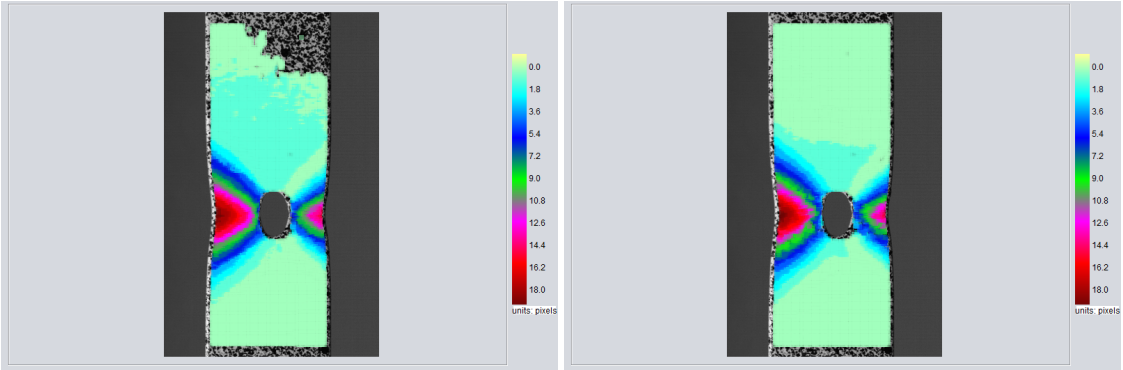


(a)



(b)

Figure 3.4: Contour plot of (a) vertical displacement in the y-axis, (b) horizontal displacement in the x-axis. The steepening of the gradient around the cutout indicates higher strain rate. Missing pixels in (b) represent the artifacts from incremental reference image updates.



(a) (b)

Figure 3.5: Contour maps of x-axis displacement for (a) analysis with incremental base image update, and (b) analysis without incremental update. The region in (a) showing zero displacement is an artifact of implementation.

Input parameters The cross-correlation kernel size chosen for this setup is 11, obtained using the zoom feature. Step size is set at two (2), giving a roughly 80% overlap. The minimum correlation threshold is set at the default value of 0.8, and image incrementing selected with a bad correlation minimum count of 100.

Output The output of the analysis is seen in Fig. 3.7, where both the horizontal and vertical displacement are shown. Since an iPhone is not able to zoom in with the same accuracy as a high-definition camera without sacrificing image quality, the view is wider than that of the Fastcam Mini. The tradeoff is a loss of definition, as the ROI for the iPhone video is 110 x 544 pixels, whereas the Fastcam records a frame where an ROI of similar placement has dimensions of 340 x 1720 pixels. The iPhone has a much lower barrier of entry and ease of use, however the loss of information resolution is significant. In this experiment, the iPhone was mounted off-axis, to make room for the larger Photron camera.

Results In Fig. 3.7 (b), v deformation shows that the maximum displacement before separation occurs is around 15 pixels, or approximately a third that of the low carbon steel sample. The stress risers in the middle can also be seen to show not nearly as steep a gradient. From Fig. 3.7 (a), there is almost no deformation in the x-axis before separation occurs. These characteristics are in line with expected results, as the Young's modulus of elasticity for aluminum is roughly a third that of carbon steel. Meaning, aluminum will exhibit much less strain than carbon steel before fracturing.

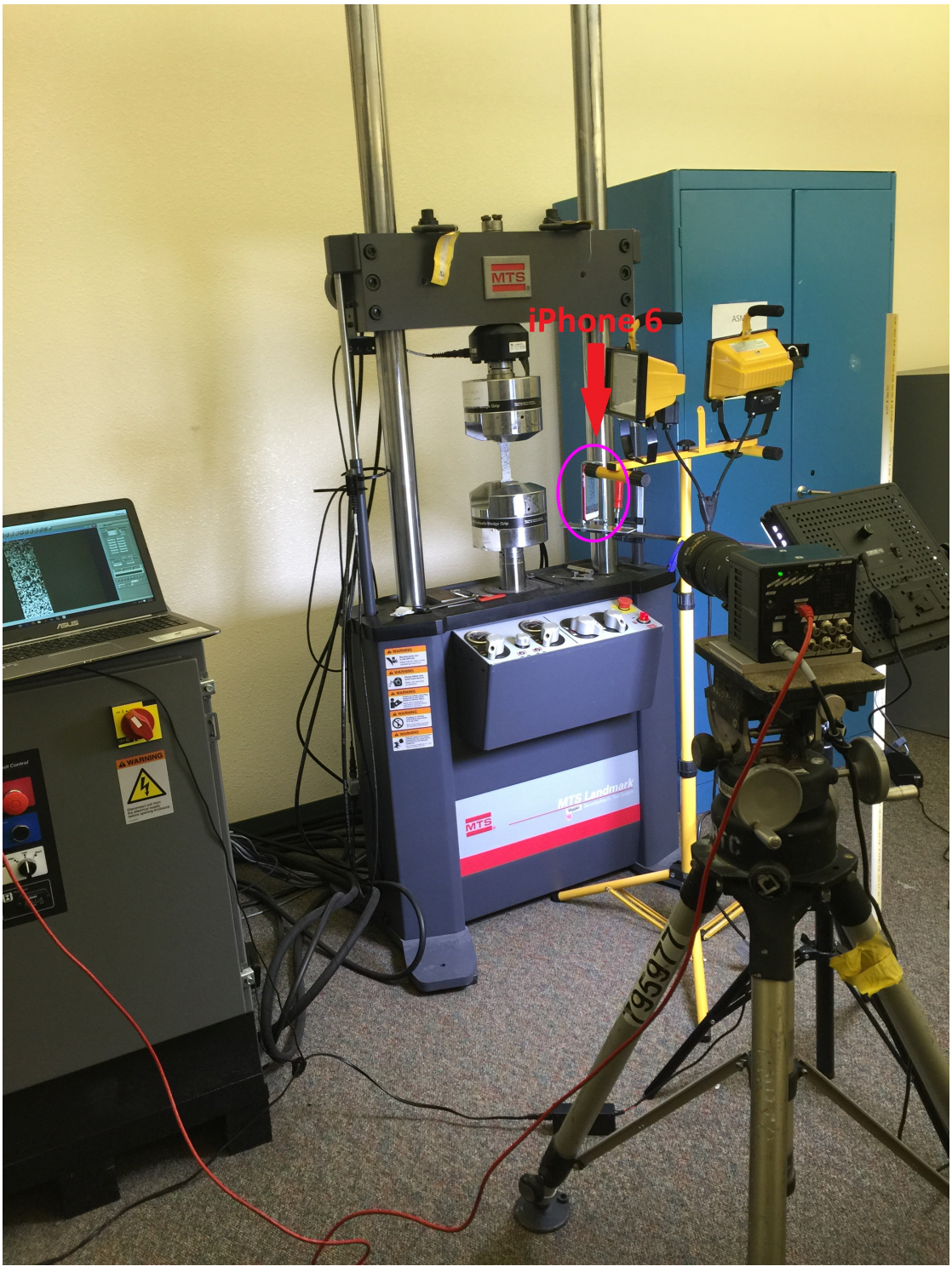
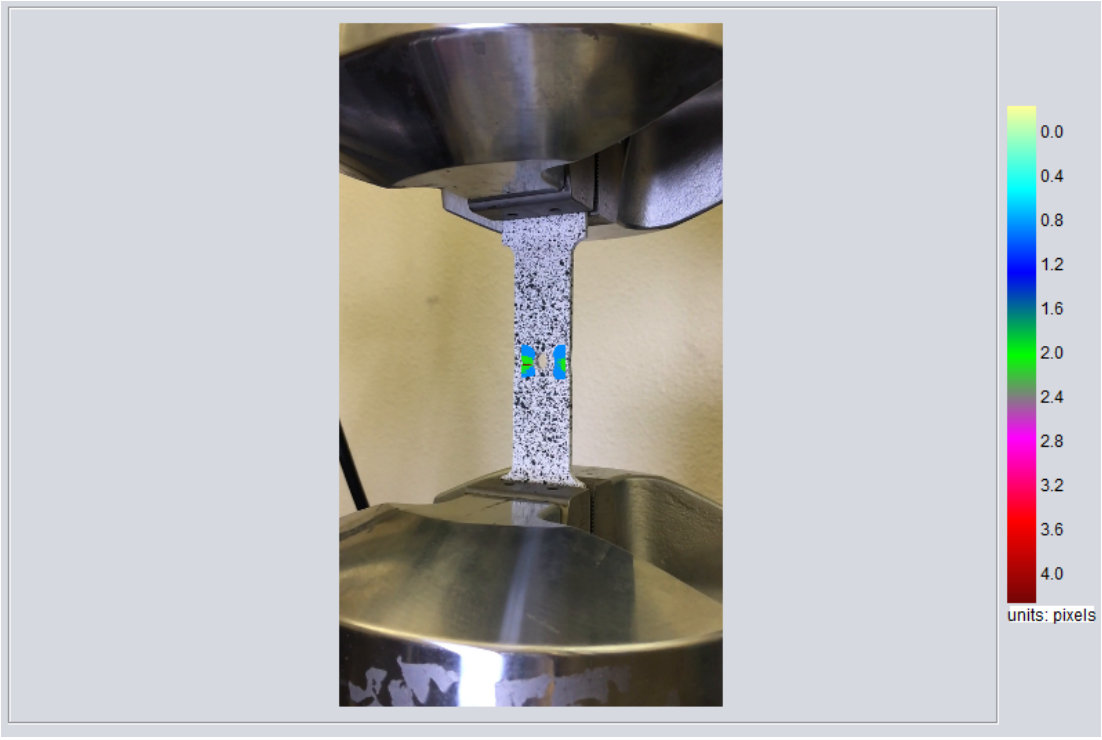
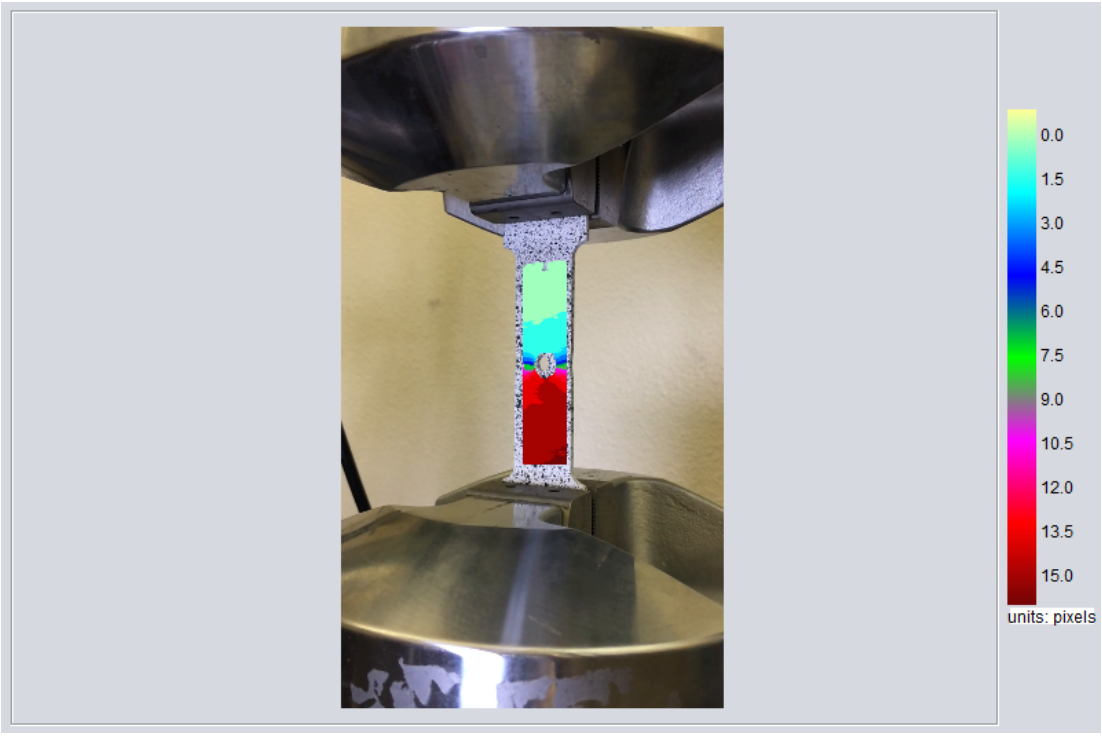


Figure 3.6: Experimental setup for iPhone test, using an aluminum speckle patterned sample. the multi-camera setup results in the iPhone set slightly off center-axis.



(a)



(b)

Figure 3.7: Aluminum digital image correlation test using an iPhone 6; (a) contour map of displacement in the x-axis, and (b) displacement in the y-axis. Compared to steel, the aluminum shows much less displacement before fracture.

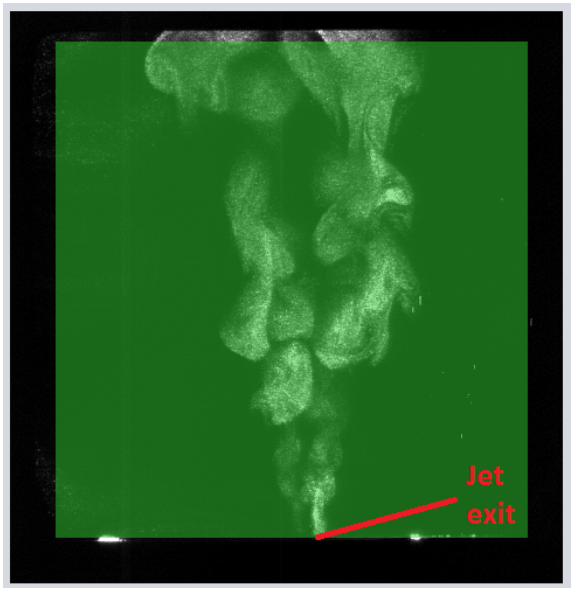


Figure 3.8: Region of interest selected for particle image velocimetry analysis.

3.2 Particle Image Velocimetry (PIV)

Experimental setup A cool-mist vaporizor is used as the test source. The vaporizer, placed in front of a laser light source and a TSI MP4 PIV camera, is situated such that the plane of the laser lay perpendicular to the camera lens. The resulting light "sheet" illuminated the water molecules emanating from the vaporizer in a single plane, enabling the irradiated molecules to be captured at a high rate of speed for acquiring PIV image pairs. 200 of the image pairs are selected for this analysis.

Region-of-Interest In Fig. 3.8, the region of interest selected is shown, using the first image in the image pair set as a backdrop. The ROI is 1150 x 1211 pixels tall, selected to capture distinguishing features of the fluid flow. After some experimentation with the images, the ROI was scaled back from the top so that the output would not contain vectors whose length resulted in the terminus outside the image boundaries (this choice was made for demonstration purposes, this has no effect on the output data or program).

Input parameters The analysis parameters for the PIV test:

- cross-correlation kernel size of 51

- Fourier window size of 82
- grid step of 30
- integer-level resolution

The analysis parameters were arrived at by a trial and error approach (common in PIV correlation analysis). With a smaller size Fourier window, the algorithm produces considerable noise, especially in the background regions. Images with fine-grained particles such as these also require a larger cross-correlation kernel when compared to a typical DIC sample with a coarse speckle patterning. A typical approach to PIV image analysis is to take a few runs to adjust input parameters to see how the data processing is affected, with the final choice of processing parameters identified that maximize resolution and accurate correlations.

Output Fig. 3.9 presents output of the program using the specified inputs, using the last image in the sequence as the background. During the progression of the program, the first image in each pair is shown with the corresponding vectors overlaid as each pair is analyzed (in this example, the user will see 200 images with overlying instantaneous velocity vectors, in pixels/image pair). For the last image upon program completion, the velocity for every interrogated subset index is averaged and plotted as the final "presentation" vector.

As is typical in velocimetry, the image shows many outlying data points that are not representative of the fluid flow, as edge zone, background regions, and especially turbulent areas can cause vectors that are obviously not in line with the rest of the results. Manual or statistical rejection of vectors may be used to suppress outliers. This program does not provide tools for manual vector suppression, and statistical rejection is left up to the user as a learning tool. However, for viewing purposes, the program applies a quick standard deviation analysis to reject any vectors over three deviations away from the mean, which does not effect the data available for export. This feature serves to dampen the most outlying noise in the data, so that the user can make adjustments to the input parameters needed after each run. If the user chooses to export the dataset for manipulation, confidence values that may help in statistically rejecting vectors is included.

Vector view adjustment If the length and scale of the output vectors is not convenient for interpretation (being either too long or short to be of visual value), the velocity vector adjustment slider may be used. A sequence of images for this experiment obtained using the adjustment tool is shown in Fig. 3.10. As the images progress from A-D, the vector forest grows thicker as the vectors begin to overlap, notably in regions of higher relative velocity. The benefit of this tool is apparent from this sequence, as it provides a "dynamic" visual perspective of the fluid flow. Often the ideal way to visualize fluid movement is to increase the vector scaling so that regions of higher velocity become denser, yielding greater contrast to regions of lower velocity.

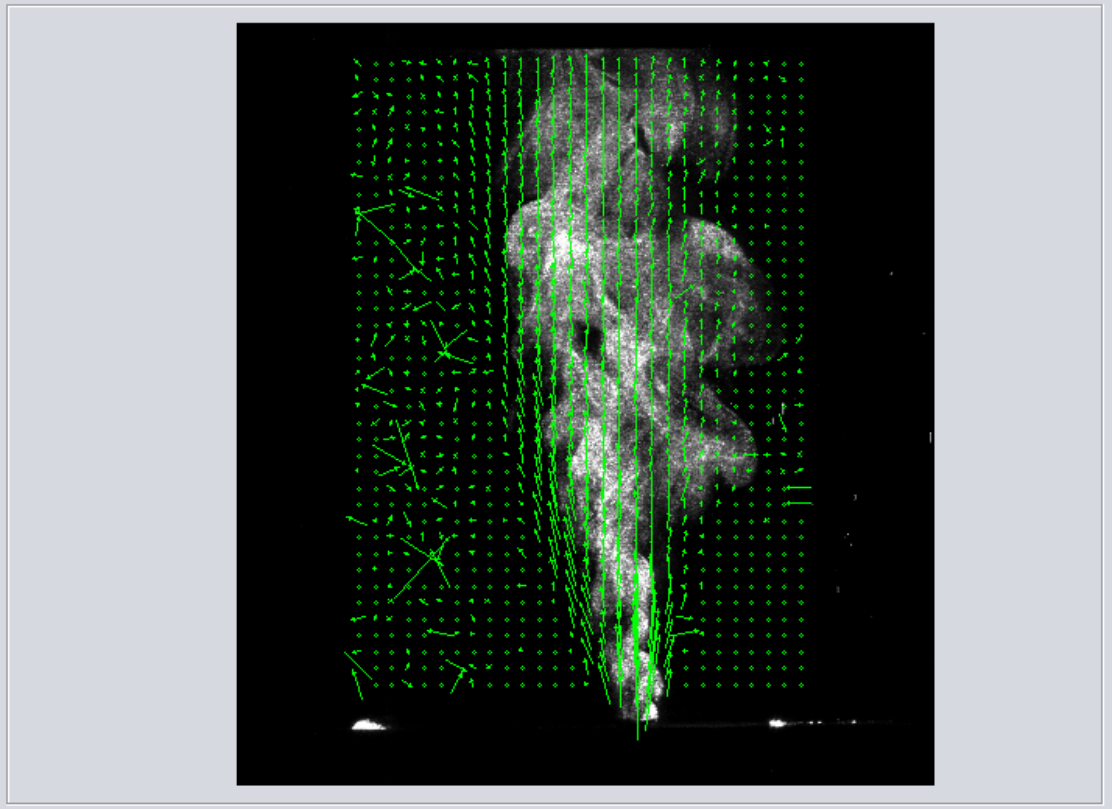
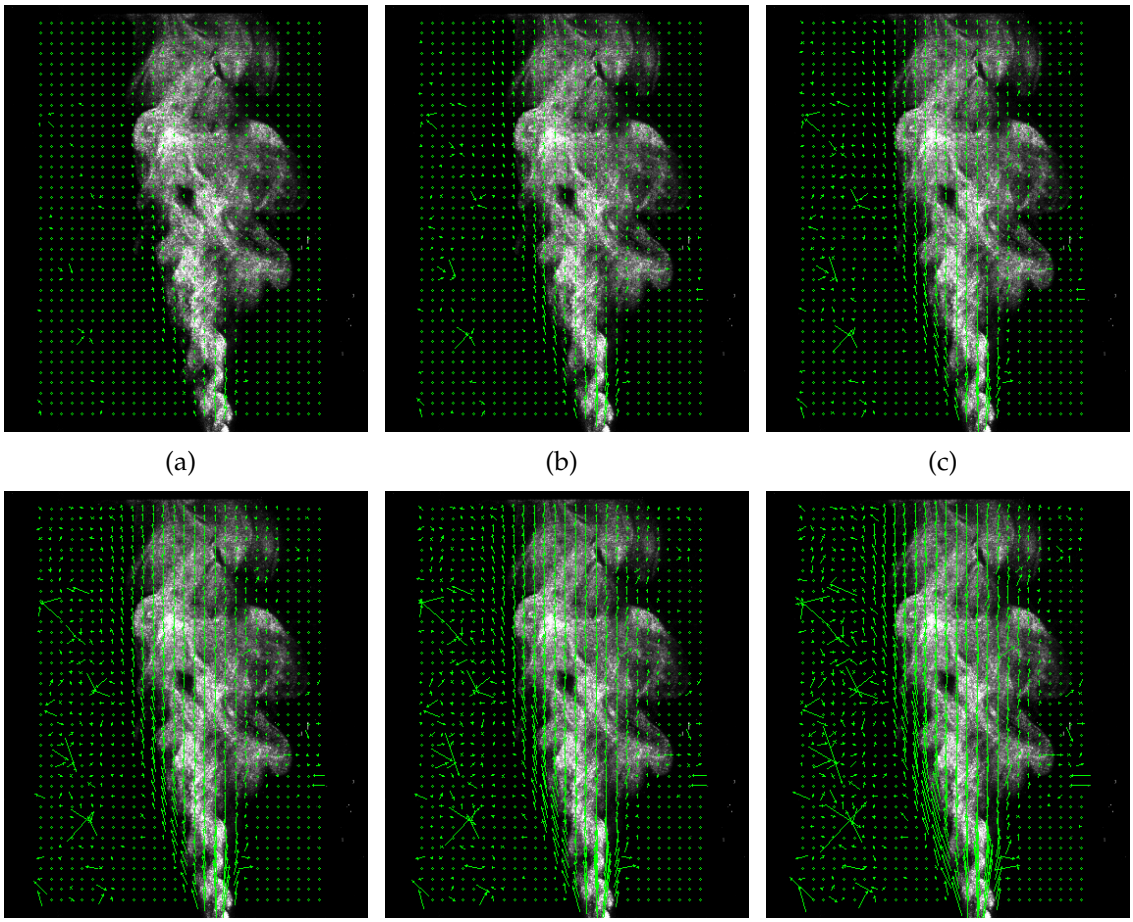


Figure 3.9: Program output for sample PIV analysis, with the last image in the input sequence as background. Highest velocities are seen at the source, with some outlying noise in the background regions.



(a) (b) (c)

(d) (e) (f)

Figure 3.10: Adjustment of vector lengths using the provided user interface tool. Images (a) - (f) show an increasing progression of velocity vector lengths on the same data set to improve the visualization.

CHAPTER 4

VERIFICATION AND VALIDATION

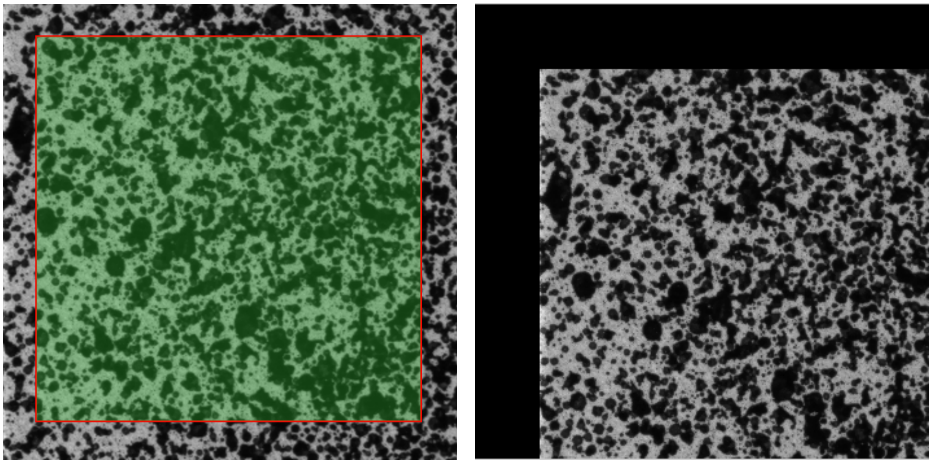
In the following chapter, the images obtained from the previous experiments are used to verify that the equations and algorithms used by this program are correct, and to validate that the experimental results are in line with expected values. For verification, sample images are treated in Matlab using image processing to deform them by a known quantity. The deformed samples are then analyzed with the PIV and DIC software, with the results compared to the expected values. For validation, three established software programs are chosen to provide baseline data for comparison with the output of this program.

4.1 Digital image correlation verification

Experimental setup To authenticate the DIC algorithm, a region of a steel dog-bone speckle image is digitally cropped to 350 x 350 pixels and subjected to pure translation in Matlab, seen in Fig. 4.1. The purpose of the experiment is twofold: to prove that with integer image translations the program tracks to 100% effectiveness, and to show that the algorithm can track large displacements between images. The kernel size is set to 21 pixels, using a step size of five (5) pixels, with a 0.8 correlation coefficient minimum (no reference image updating) for all runs.

Experimental results The image in Fig. 4.1 A was translated for 10 steps along each axis, as well as the diagonals, by the following magnitudes per step:

- 1 pixel
- 5 pixels
- 10 pixels
- 30 pixels



(a) (b)

Figure 4.1: a) Base speckle image used for DIC algorithm verification with ROI for calculations shown. b) Example result after translating (a) 50 pixels in the x-axis, and 50 pixels in the y-axis.

Data for each subset index are averaged for all 10 images, and the percent error from the known outcome calculated by:

$$\%_{error} = \frac{\#_{experimental} - \#_{known\ value}}{\#_{known\ value}} \cdot 100 \tag{4.1}$$

Results for the 10-pixel displacement are shown in Table 4.1. The error for every result is 0%, which indicates that for every subset location (every fifth pixel in the image), the algorithm successfully tracked to the corresponding location in the subsequent translated images.

For the transformations of 1, 5, and 30 pixel steps in all axes, the correlation success rate is also 100% (unshown). Therefore in strict subset translation, the algorithm has zero error in tracking and identifying the corresponding locations even when large displacements occur.

4.2 Particle image velocimetry verification

Experimental setup PIV algorithm integrity is verified in the following manner: a sample of image 350 x 350 pixels is sampled from the base image used in the preceding chapter, from Fig. 4.2 (a). The image is then translated 10 times with an acceleration of one (1) pixel per translation step, with a starting velocity of one (1) pixel/pair (after 10 translations the terminal velocity equaled 10 pixels/pair). Acceleration is added so that the algorithm sees movements other than at a constant velocity. The Fourier window is set at 64, with a grid step size of 15.

Step x-axis	Step y-axis	Total steps	Results x-axis	Results y-axis	% error x-axis	% error y-axis
10	0	10	50	0	0	0
10	-10	10	50	-50	0	0
0	-10	10	0	-50	0	0
-10	-10	10	-50	-50	0	0
-10	0	10	-50	0	0	0
-10	10	10	-50	50	0	0
0	10	10	0	50	0	0
10	10	10	50	50	0	0

Table 4.1: Results from DIC algorithm after translating Fig. 4.1 (a) by 10 pixels for 10 steps, resulting in 0% error for every calculation (all units in pixels). Results are averaged across every pixel in the image.

Results For the results displayed in Fig. 4.2 (b) - (f), at each subset index location all data is averaged together from each image pair, and scaled for viewing on-screen. The algorithm recorded a perfect tracking record, with each index's final average velocity equaling 5.5 pixels/pair.

4.3 Subpixel registration verification

Three methods of subpixel registration are available in this program: parabolic, Gaussian, and Lagrange. Each method is an interpolation process; parabolic and Gaussian rely on a 4-connected neighborhood, and Lagrange is based on an 8-connected region (detailed in Chapter 2). Using the following procedure, each registration process is experimentally verified to have the correct implementation in code and to quantify the error introduced by the interpolation process.

Experimental setup The base image in Fig. 4.1 (a) is digitally translated in Matlab by one pixel in the x and y directions, for a total of 10 steps. At each step the image is analyzed using all three subpixel registration methods, using a subset size of 21 pixels and a grid step of 1 pixel.

Results 90,000 subsets per translated image were analyzed, and the movement of each subset centroid averaged for the entire image in both the x and y dimensions. By using a translation step of one pixel (no interpolation required between translation points by the deforming method), the result for every subset can be expected to arrive at exactly one pixel of displacement in both u,v, so therefore an exact error calculation can be obtained. Results are shown in Table 4.2.

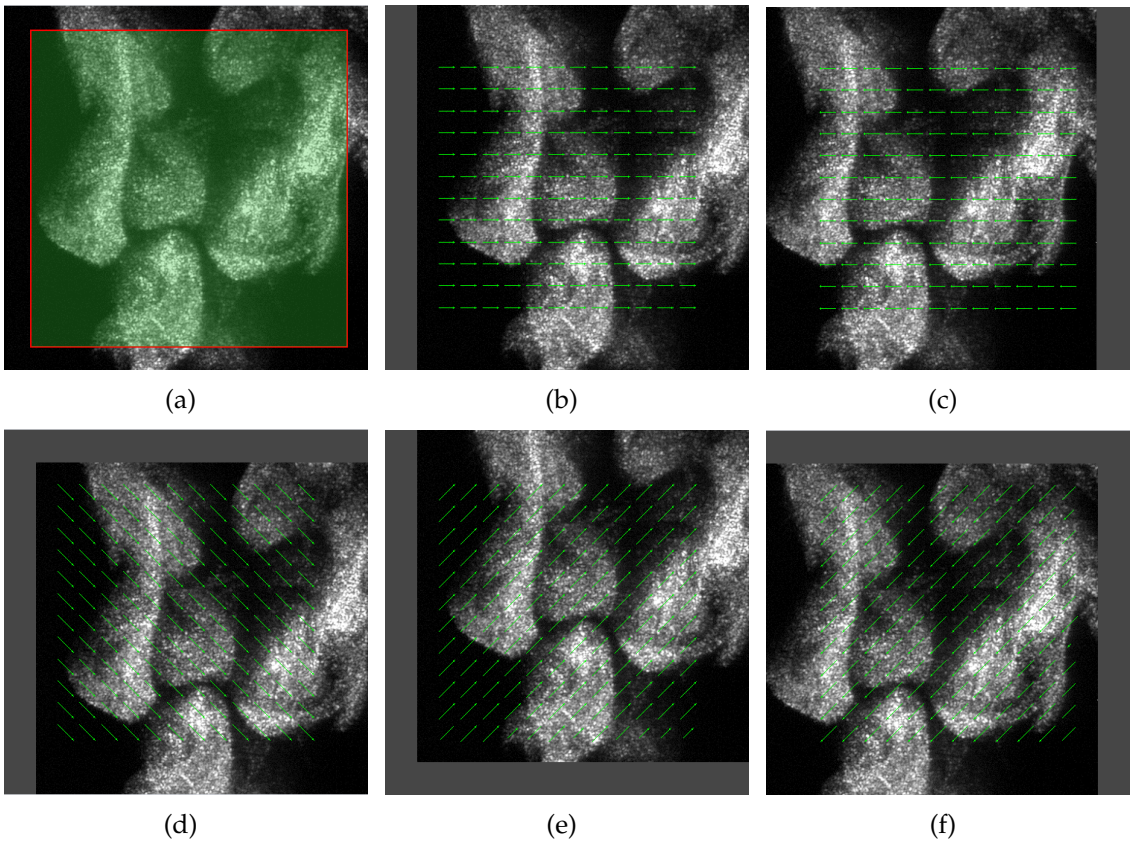


Figure 4.2: Results of PIV algorithm verification. The results are displayed as average velocity vectors for each subset over all image pairs. Image tracking results in a 100% correlation rate for purely translational velocities.

Subpixel method	x-axis % error (avg)	y-axis % error (avg)	**Total % error (avg)**
Parabolic	0.0082	0.0021	**0.0051**
Gaussian	0.0054	0.0028	**0.0041**
Lagrange	0.0021	0.0033	**0.0027**

Table 4.2: Percent error of the three subpixel registration methods, calculated via digital translation of Fig. 4.1 (a).

Discussion The error introduced by each registration process is small; for reference, 0.005% error is equal to a 0.00005 pixel difference between the expected and experimental value, or roughly 0.185 micrometers (μm) in this image sample. The image itself is approximately 13,000 x 13,000 μm (13 x 13 millimeters). The most accurate method is Lagrange, however the computational cost is extensive; the analysis takes an extra 10 seconds to complete (a 50% increase over Gaussian and parabolic).

4.4 Digital image correlation validation

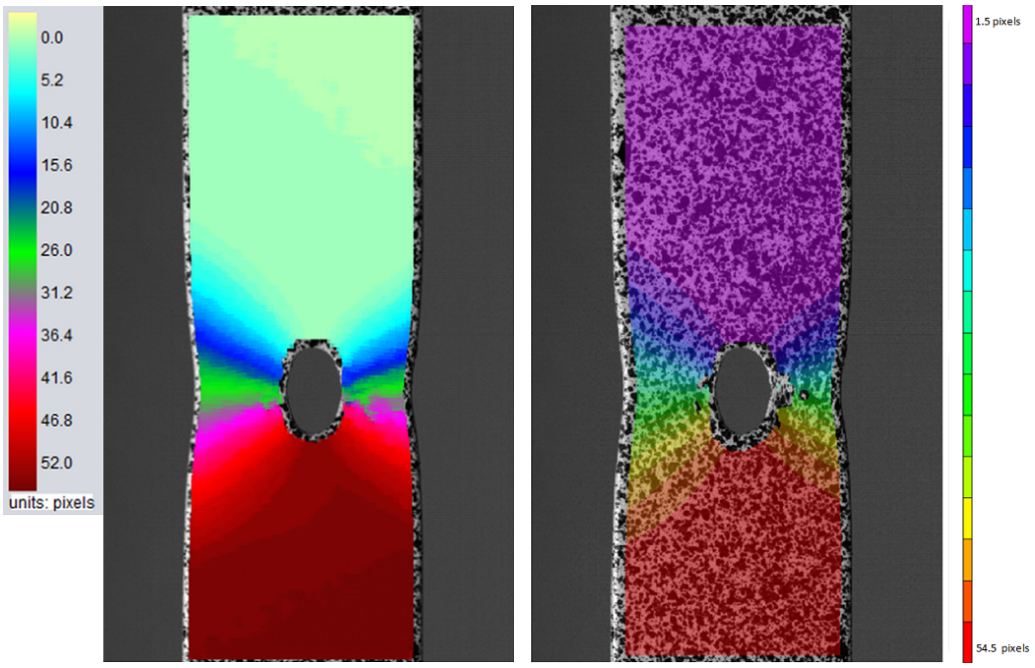
To validate the DIC program's effectiveness in a real environment, image sets from the experiment outlined in Chapter 3, using the steel sample are compared to the output of a program called VIC-2D (Visual Image Correlation - Two Dimensional [Solutions, 2017]). VIC-2D is a professional software package used widely in industry and academia, using a proprietary algorithm that is extremely fast and resistant to complications arising from sample deformation and pattern decorrelation. It is considered to be a baseline for this results comparison due to its widespread use and regard.

4.4.1 Contour plots

For deformation mapping, contour plots are highly effective for visualizing the deformation undergone by a sample during testing. Sometimes a deformation map is the only desired outcome, especially in a teaching laboratory environment, so this program must be verified to produce an accurate and reliable contour plot. To prove the contouring function, 175 images are analyzed with this program and VIC-2D using the following input parameters:

- Kernel size of 21
- Five (5) pixel step size

The minimum correlation coefficient for this program is set at 0.8 (VIC-2D does not require inputs other than those outlined above).



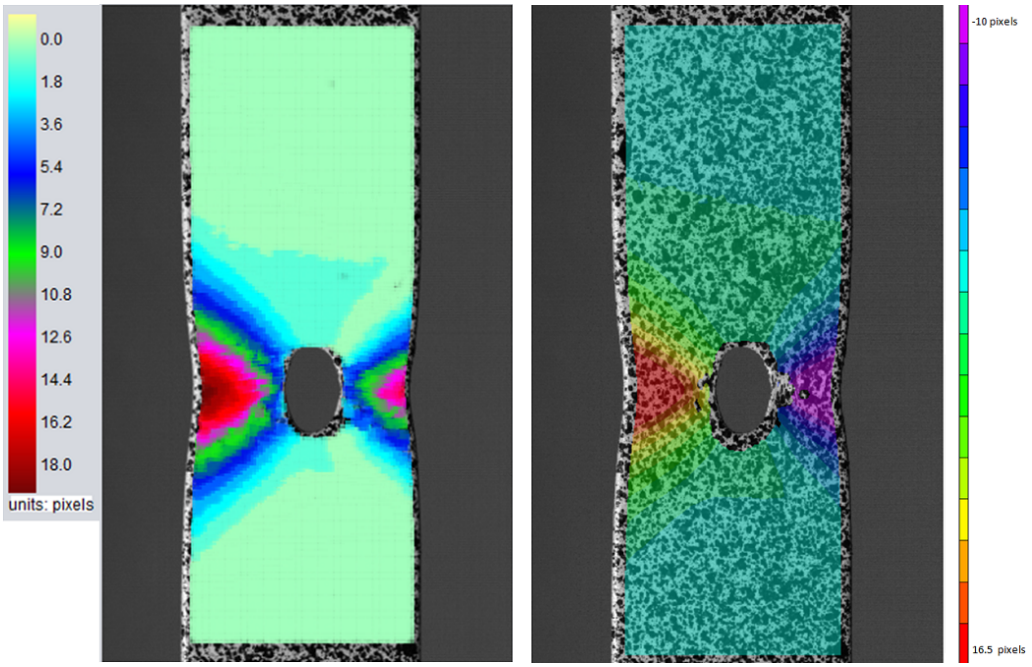
(a) (b)

Figure 4.3: Non-increment y-axis deformation for (a) this program, and (b) VIC-2D.

Static reference image analysis The contour plots in Fig 4.3 show the y-axis (v) displacement results using a non-incrementing reference image update scheme (static reference). The VIC-2D output contour map treats the data to 18 discrete colors. This program has a broader color palate(66), so the intervals do not match exactly. However, it is apparent that the results are quite similar to each other and show the same overall displacement outlines and characteristics.

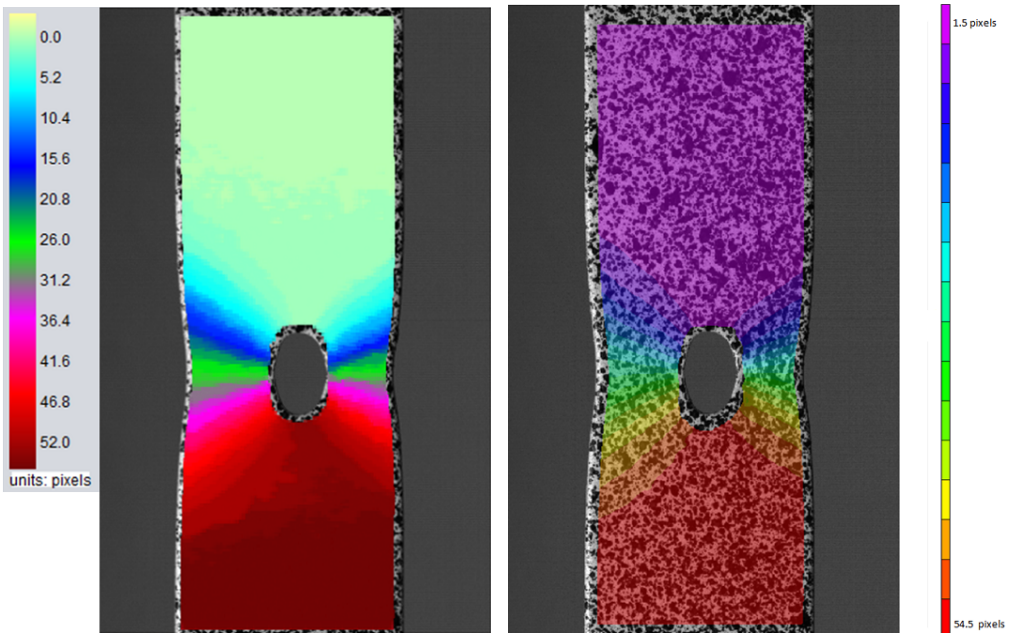
The middle region of the plots around the center cutout show interesting features, namely a high gradient of color change that indicates the highest strain rate in the sample, as well as missing pixels (VIC-2D image) or a muddled color scheme (this program). This is indicative of decorrelation effect, whereby the speckle pattern gets deformed to the point where a subset cross-correlation cannot be made with a high degree of confidence. VIC-2D chooses to show the contours from the very last image data, so when a correlation is not made, no pixels are colorized. This program contours with the last known good value, resulting in pixels that don't match up with their surroundings in highly decorrelated regions.

Fig. 4.4 illustrates the results of the previous run in the x-axis. Again, the contour lines are of similar placement and breadth. The left side of the sample shows higher total displacement in both images, with a much broader band of deformation.



(a) (b)

Figure 4.4: Non-increment x-axis deformation for (a) this program, and (b) VIC-2D.



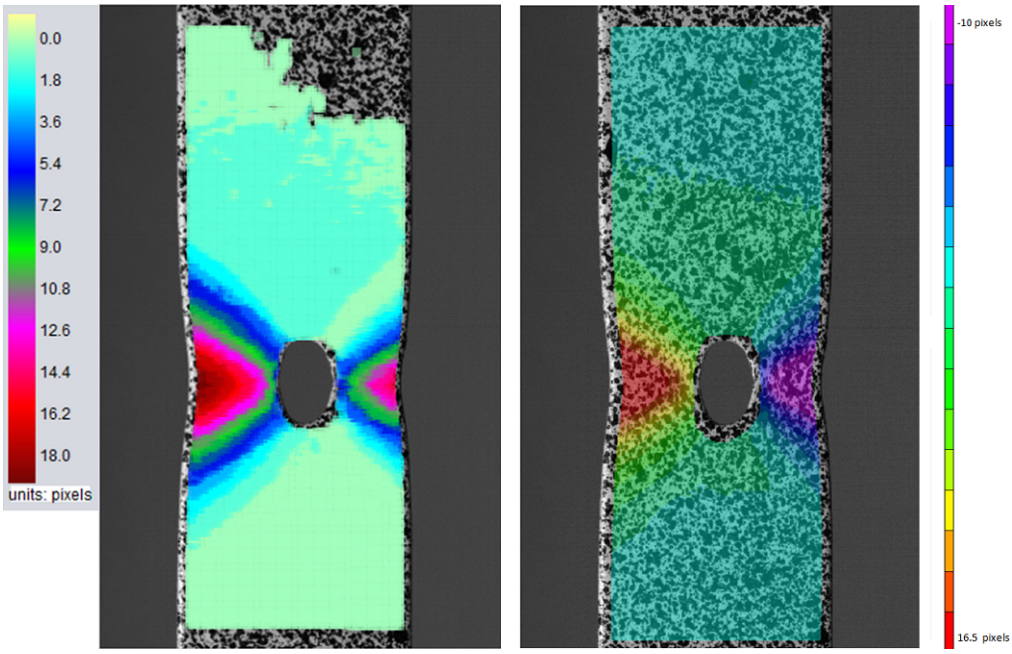
(a) (b)

Figure 4.5: Incremental reference update, y-axis deformation. The contour lines are much smoother using this technique, providing a cleaner contour map.

Reference image updating VIC-2D offers an option to update the reference data after each image, so that the reference image is the one directly preceding the current image. This program offers more flexibility, where the user can choose the number of non-correlated subsets before a reference image update occurs. There is also an option to update after every image, which is selected for this analysis to match with VIC-2D.

Using identical input parameters to the preceding analysis, the image set is evaluated in each program using the outlined incremental updating scheme, with the results presented in Fig. 4.5 and Fig. 4.6. The results are very similar to the non-incremented output, however the response in the decorrelated region is much smoother and without any missing pixels for VIC-2D, or for this program, mixed-color contour lines. Reference image updates, however, do introduce noise and error into the data, but for a strictly visual interpretation, the results are more presentable and readable.

The contour plot from this program in Fig. 4.6 A is missing pixels in the top-right region. This indicates that there is indeed some amount of difference between the non-increment and incremental analysis, as there were no missing pixels in the previous contours. VIC-2D colors its subsets at non-integer levels, so does not suffer from the same discrepancy.



(a) (b)

Figure 4.6: Incremental reference update, x-axis deformation for (a) this program, and (b) VIC-2D. The contours are much smoother, with some missing pixels in the top-right of this program, due to the integer referencing scheme.

4.4.2 Data comparison

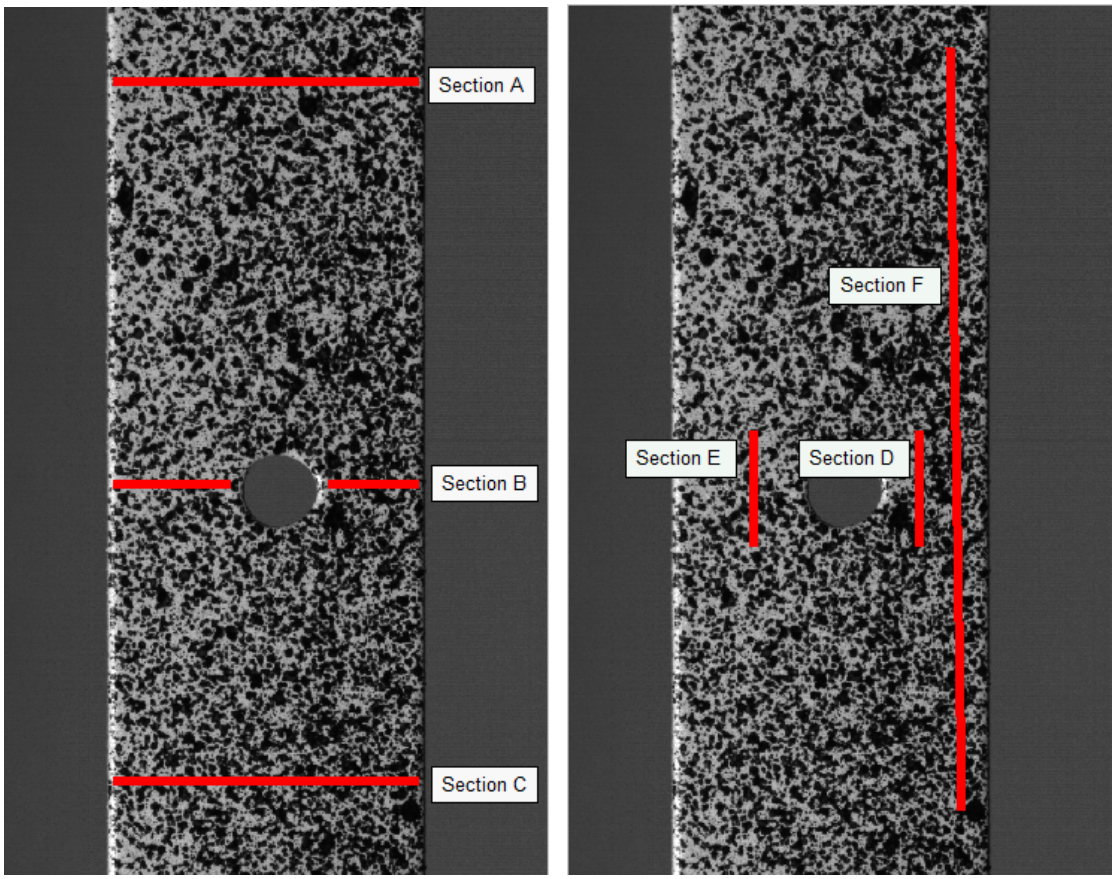
Segment location The previous contour plots provide a reasonable comparison to the overall quality of the output provided by this program. To quantify the accuracy of the data against the VIC-2D baseline, several discrete cross-sections of the image set, as seen in Fig. 4.7, are analyzed independently in both programs. For the horizontal sections labeled A, B and C, the lengths are 320 pixels wide by 5 pixels tall. The vertical sections labeled E and D are 150 pixels tall by 5 pixels wide.

As seen from the previous contour maps, the top and bottom of the sample undergo fairly uniform deformation (mostly in v), whereas the middle experiences a much wider variation in strains. Therefore, segments A & B should offer a fairly uniform displacement curve, whereas segment B should be more challenging, especially toward the end of the image set when heavy decorrelation effects arise in this region. Since the sample is being pulled fairly uniformly along the y-axis, little u displacement occurs overall, as the sides are moving toward each other and cancel out. To capture movement for comparison in the x-axis, sections D, E & F in Fig. 4.7 (b) are placed in the most active region of the sample.

To compare the segments between the programs, each segment is treated as a mask, with the underlying deformation point data averaged for each image. Since more data points are desired for comparison, the step size is reduced to one (1) so that every pixel underlying the mask is correlated. The parabolic subpixel estimator is used for this program.

Results Fig. 4.8 shows the v deformation results from each program. Segment A shows little displacement, as expected from the contours. Sections B & D both show a rather large jump in the displacement around image # 100, then level out. Of significance is the preciseness of the curve matching. The results from this program follow the curves and topography of the VIC-2D output, only showing a very slight separation towards the end of section B.

In Fig. 4.9, the data for u displacement from segments E & D is displayed. The topography of the results again matches very well, the output from this program following almost exactly the character of VIC-2D data. However, at around image 165, separation starts to occur between the lines. From the non-incremented contour plots seen in Fig. 4.4, the region in which these two segments are located shows decorrelation at the end of the image set. The right side of the circular cutout shows more missing data than the left side, which matches this comparison. Separation between the lines occurs earlier and more markedly in segment E than D. Therefore, if VIC-2D is considered to be the baseline, the algorithm used by this program suffers in accuracy when large deformations are present. The probable cause of the line differentiation is that because the algorithm used by this program loses track of more points than VIC-2D when decorrelation occurs, the average along the segment is influenced more heavily by the smaller displacement areas located at the top and bottom of the segment masks.



(a) (b)

Figure 4.7: Locations for DIC results comparison. Segments in (a) are used for vertical displacement comparison, and (b) for horizontal.

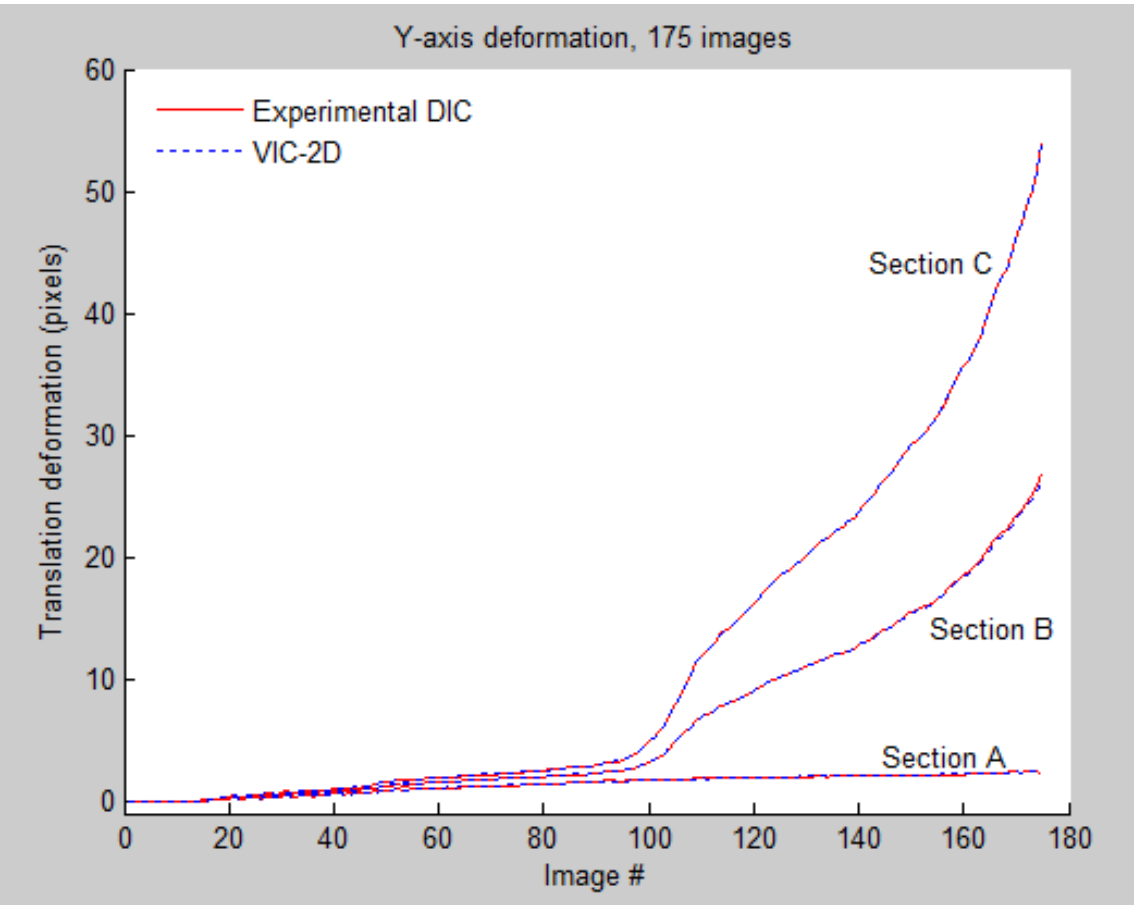


Figure 4.8: *Y*-axis displacement comparison. The plots comparing this program to VIC-2D are virtually identical, with a very slight deviation at the end of section B.

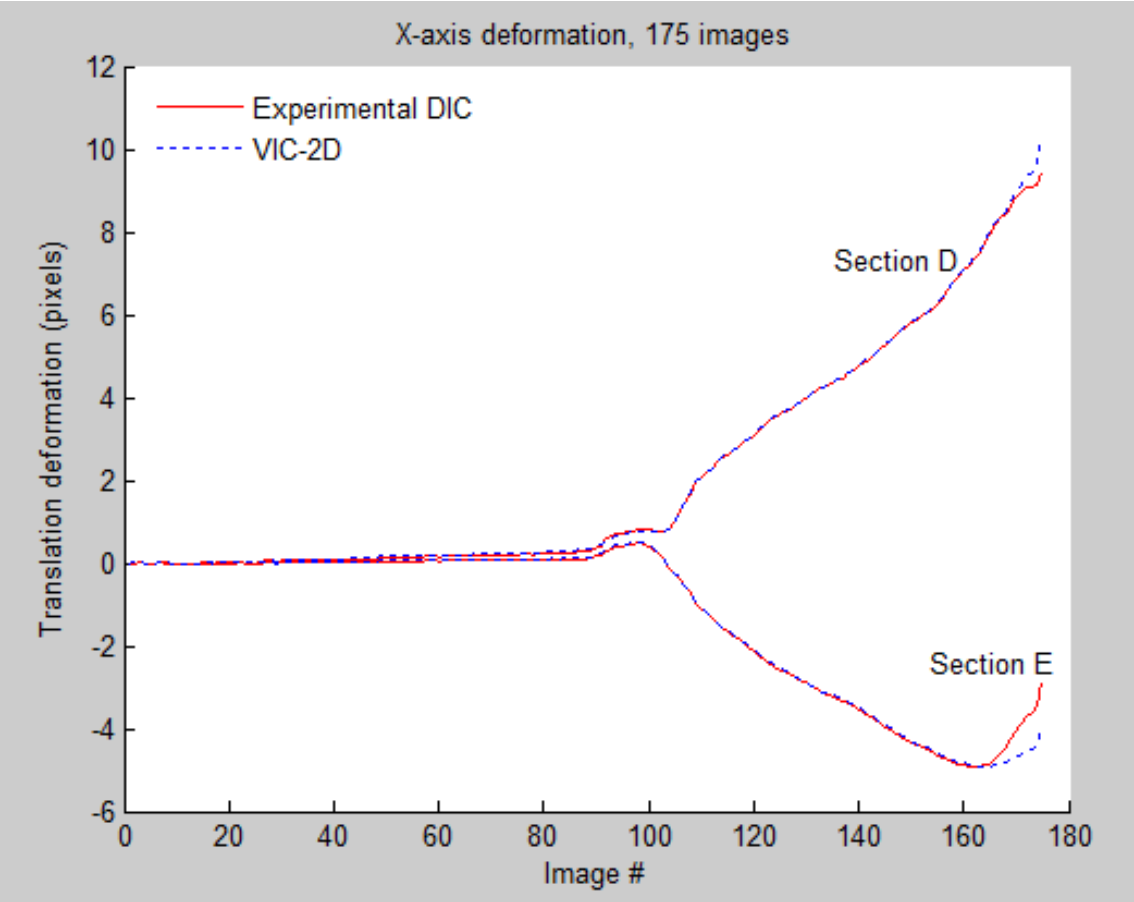


Figure 4.9: *X*-axis displacement comparison. Plots are very similar until around image # 165, which is the point that high speckle pattern deformation on the pattern occurs.

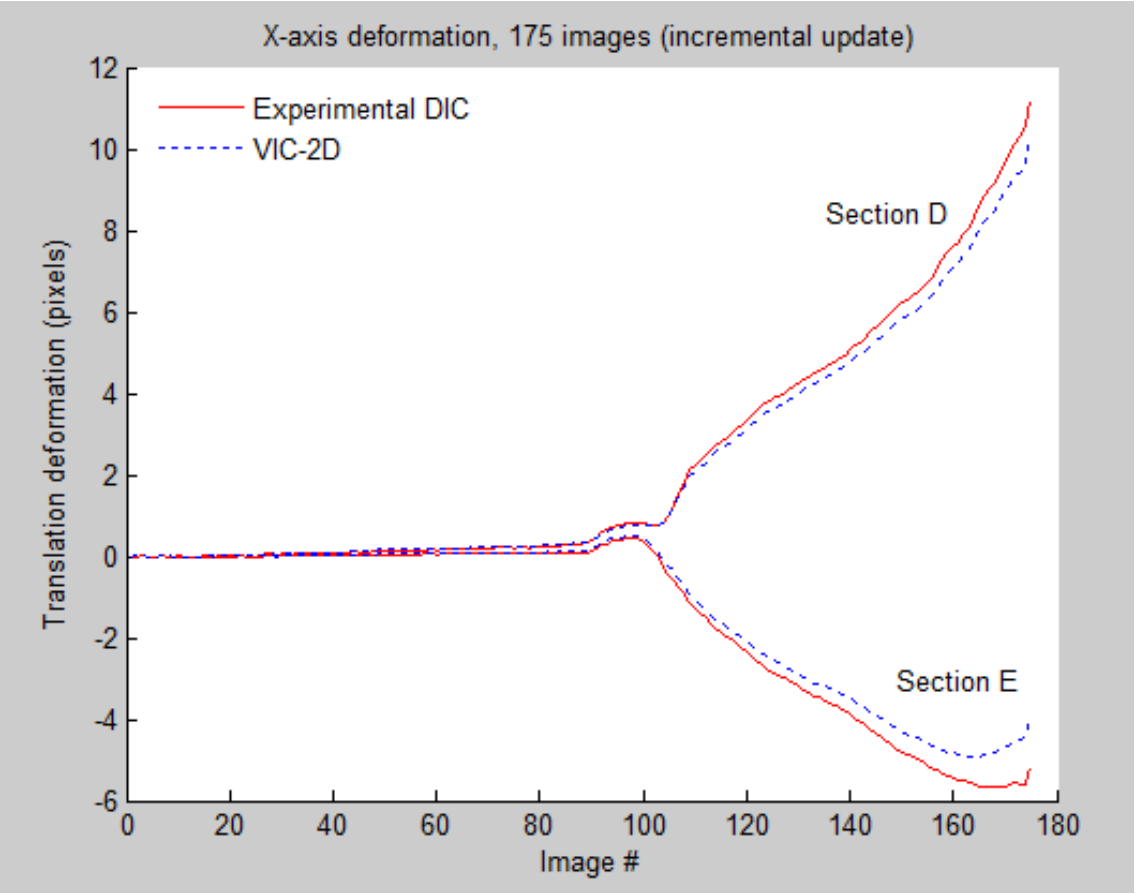


Figure 4.10: Displacement in u using incremental reference image update. The plots deviate considerably due to the integer-update scheme used by this program.

The displacement profile for segment F from Fig. 4.7 (b) is shown in Fig. 4.11. For this comparison, the data is taken from image # 155, before significant decorrelation and data loss occur. At each y-axis pixel location, a 10-pixel wide set of displacement calculations are averaged together from the horizontal plane centered on the segment. The results show the data from this program are consistent with VIC-2D, especially as the sample deforms noticeably in the x-axis.

Using an incremental reference-image update scheme, the algorithm can make up some ground on lost/decorrelated subsets. In Fig. 4.10, the results from VIC-2D (non-incremented) are shown against results from this program using an incremental scheme of 20 bad correlation points before reference image update. Gains are made in the shape and form of the line, as there is no sharp uptick at around image # 160, and the topography of the lines remains fairly constant. However, when the reference updates start to occur around image # 110, the data

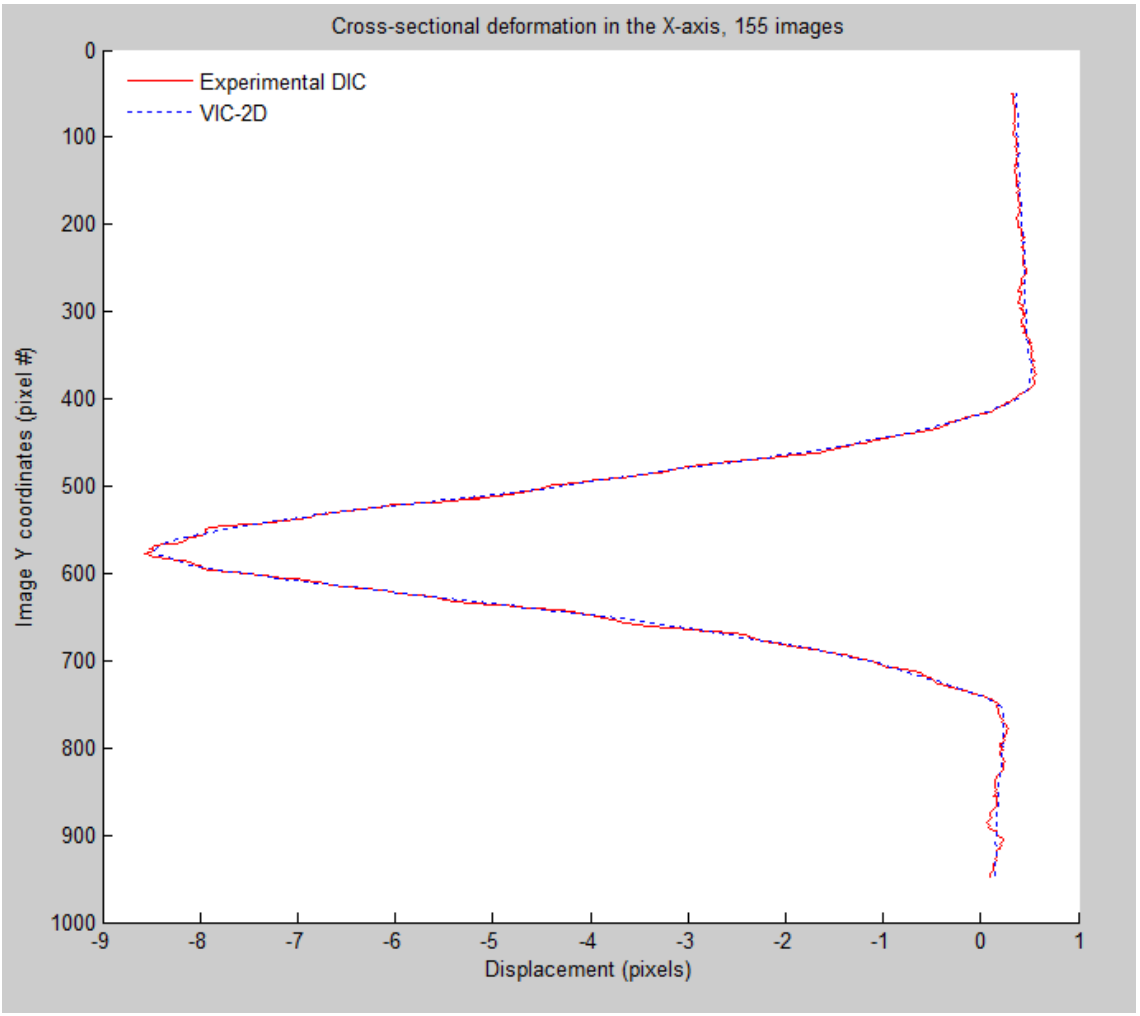


Figure 4.11: Displacement contour for section F from Fig. 4.7 (b), with displacement in the horizontal direction. The segment is treated as a 10-pixel wide mask, using the displacement values at image # 155. Contours follow closely, with slight deviations occurring.

starts to separate from the baseline, again evidencing the introduction of error using this method.

Discussion Even though the data from this program diverges from the baseline when decorrelation occurs, until such a circumstance arises, this program provides good results. VIC-2D, while not dropping as many data points, still struggles when the specimen markings are deformed beyond a reasonable expectation of recovery.

4.5 Particle Image Velocimetry validation

To validate the accuracy and consistency of the PIV algorithm using Fourier transforms, two other software programs are used to create velocity vector field plots and to obtain and analyze the raw data for statistical comparison. The foundational principle of each of these programs is the same: the Fourier transform is applied to a subset of the image to locate the pixel coordinates of the correlation peak.

4.5.1 Comparison of software programs

The first program, Insight 4G, is a comprehensive "Global imaging, acquisition, analysis and display software suite" from TSI [Inc., 2017]. Insight 4G is a complex, industrial-scale program available for license purchase, with a steep learning curve due to the many different input parameters available for adjustment. The program runs extremely fast, each image pair taking roughly a third of the time to analyze as the same pair in this program. Some of the speed advantage is due to the base language being C, which is an inherently faster computational language than Java. Other factors, such as a better Fourier transform library, and probably more importantly, professional code optimization, also influence the speed differences. However, Insight 4G is only available for use on the Microsoft Windows platform.

The other software used for comparison is a free program available under the BSD license called PIVlab [Initiative, 2017], created as part of a PhD program analyzing the aerodynamics of bird flight. The program has been used in academic papers and journals, and is well-cited [Thielicke and Stamhuis, 2014a,b, Thielicke, 2014]. PIVlab runs entirely as an application in the Matlab environment, so requires the Matlab license or use of the platform. Since Matlab uses a combination of programming languages (Java for the interface, C and Fortran for computation), it can reach higher speeds than a pure Java implementation. The Fourier transform used is the non-free (non-GPL) licensed version of the FFTW Fastest Fourier Transform in the West, [MIT, 2017, Matlab, 2017]. PIVlab also offers a "multi-pass" approach, which refines the search by decreasing the size of the Fourier window a user-defined amount in up to three passes.

Input variables Since all three programs operate under a similar FFT approach (all three can also operate using direct cross-correlation, which is a more time-consuming process), the only two variables required to be held constant across the platforms are the Fourier window size and the grid step size. After a few trial and error passes, the optimal Fourier subset size for the image set from the vaporizer experiment is in the neighborhood of 80 pixels, so the window size for all three programs is set at 82 pixels square. This subset size produced roughly the most successfully correlated points across all three programs.

Typically in a PIV analysis, the grid step size would be associated with the Nyquist sampling rate. The Nyquist Theorem states that the sampling rate should be twice the highest desired recorded frequency in the data, from Higgins [1996]. Therefore in a PIV analysis utilizing the frequency domain, the smallest object recorded in the data would be the Fourier window size, so the grid step (sample rate) would equal half the window size. For the 82-pixel kernel used here, the step size therefore would be 41 pixels. However, more data points are desired for this comparison analysis so that the statistical cross-comparison has more information to work with. Therefore the step size for the following analysis is set at 15 pixels, which gives roughly 4,500 subset centers to analyze from the ROI shown in Fig. 3.8.

4.5.2 Velocity vector fields

The first results comparison is a visual observation of the velocity vector fields obtained from each programs. The results from each analysis were exported to a data file and loaded into Matlab, which offers a function to draw vector fields using the same scale and structure for each data set.

Vector rejection Any PIV analysis, due to randomness as well as out-of-plane motion of the seed particles, results in a large number of outlying vectors (noise). Regions where the Fourier window is completely encapsulated by background pixels also produces many outliers in a Fourier analysis. Fig. 4.15, using the Insight 4G raw results as a background, shows this effect.

To combat this, both manual and statistical rejection of vectors can be perfomed. Manual rejection requires the program to allow users the option to select any vectors that do not visually reconcile with the velocity field around them, which can then be removed from the data. Both Insight 4G and PIVlab provide such functionality, this program does not.

Statistical rejection relies on deviation from the mean:

$$\sigma = \sqrt{\frac{\Sigma(x - \bar{x})^2}{n - 1}} \tag{4.2}$$

Where x represents each value in the set, and $\bar{x}$ is the mean of the set. In a typical case the standard deviation (σ) from the mean is calculated for the well-correlated subset indexes. Then, vectors may be rejected by setting a threshold for the number of standard deviations from the mean each data point is allowed to surpass. For example, if the set mean is 10 pixels and the σ is five (5), by using a threshold of one (1) σ, any vector over the length of 15 or under the length of five (5) pixels would be excised.

Vector rejection in a PIV velocity field is a subjective process dependent on the user, the goals of the analysis, and other factors. For this visual comparison, a velocity field without much noise from outlying vector lengths is desired. With a little experimentation, the σ threshold for rejection was set at three (3), resulting in an even vector field with few outliers. Results from each program are shown in Figs. 4.12, 4.13 and 4.14.

Discussion Overall, each velocity field possesses equivalent flow trends and a conical shape, with approximately equal dimensions. The general trend is upwards, with longer, overlapping vectors at the bottom (source) where the velocity is highest. A slight rightward flow appears in the upper right area of the cone, starting at around x = 850, y = 800. Both this program and PIVlab show a higher velocity in the x = 650 plane, whereas Insight 4G shows a more evenly distributed field throughout the conical region.

In the upper-right quadrants of both this program's output and that of PIVlab, there appears to be difficulty obtaining enough reliable vectors. Both fields show square-shaped regions of vectors non-conforming with the fields around them. Insight 4G also has trouble in the same areas, but to a lesser extent. Most likely, there were fewer seed particles flowing through these regions from image to image, so the averages from the images pairs did not have enough data to normalize with the surrounding areas. Each algorithm had varying degrees of success handling this lack of data.

4.5.3 Vertical data comparison

To quantify the similarities and differences between the program outputs across the entire vertical plume, the region of interest is segmented into 22 sections across the conical area showing the most consistent movement, as seen in Fig. 4.15. The high-variance regions outside this area are excluded because they do not lend useful data to the analysis, being composed of primarily background pixels. Each of these line segments is 90 pixels thick, to encapsulate 5 subsets centers across each band (observing the 15-pixel grid step size).

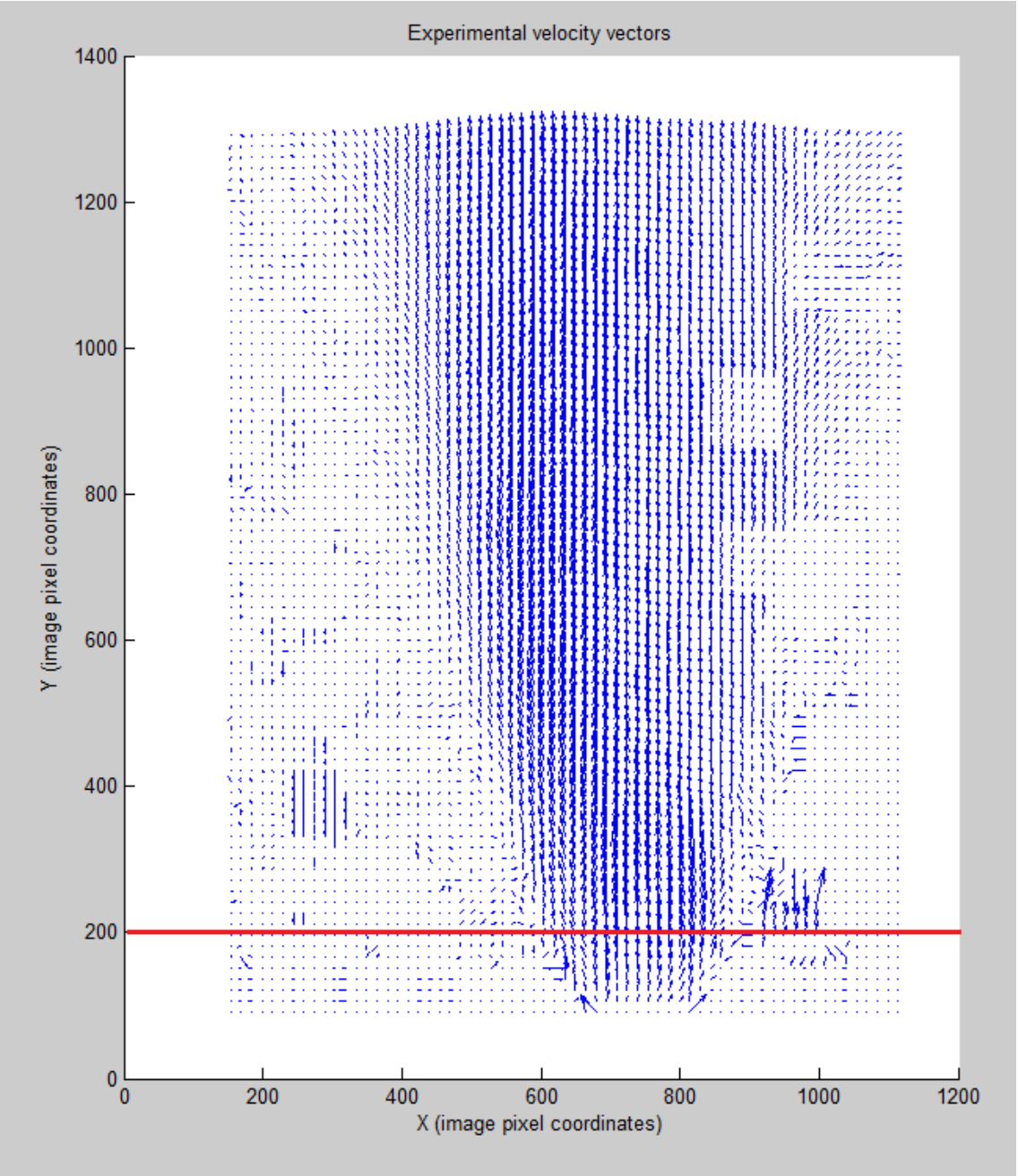


Figure 4.12: Velocity vector field from this program's output. Data points are filtered to two standard deviations from the mean.

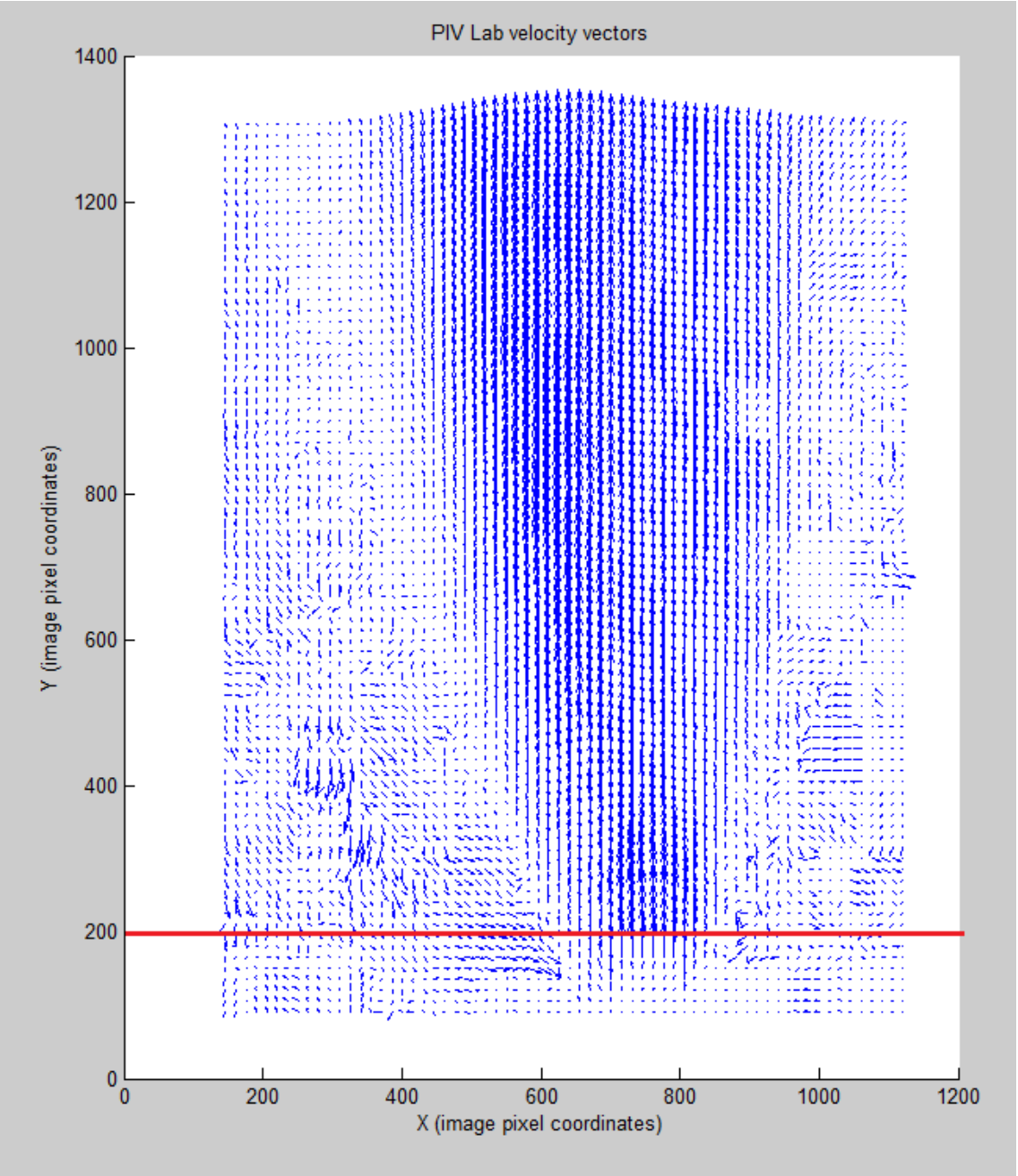


Figure 4.13: Velocity vector field from PIV Lab. Data points are filtered to two standard deviations from the mean.

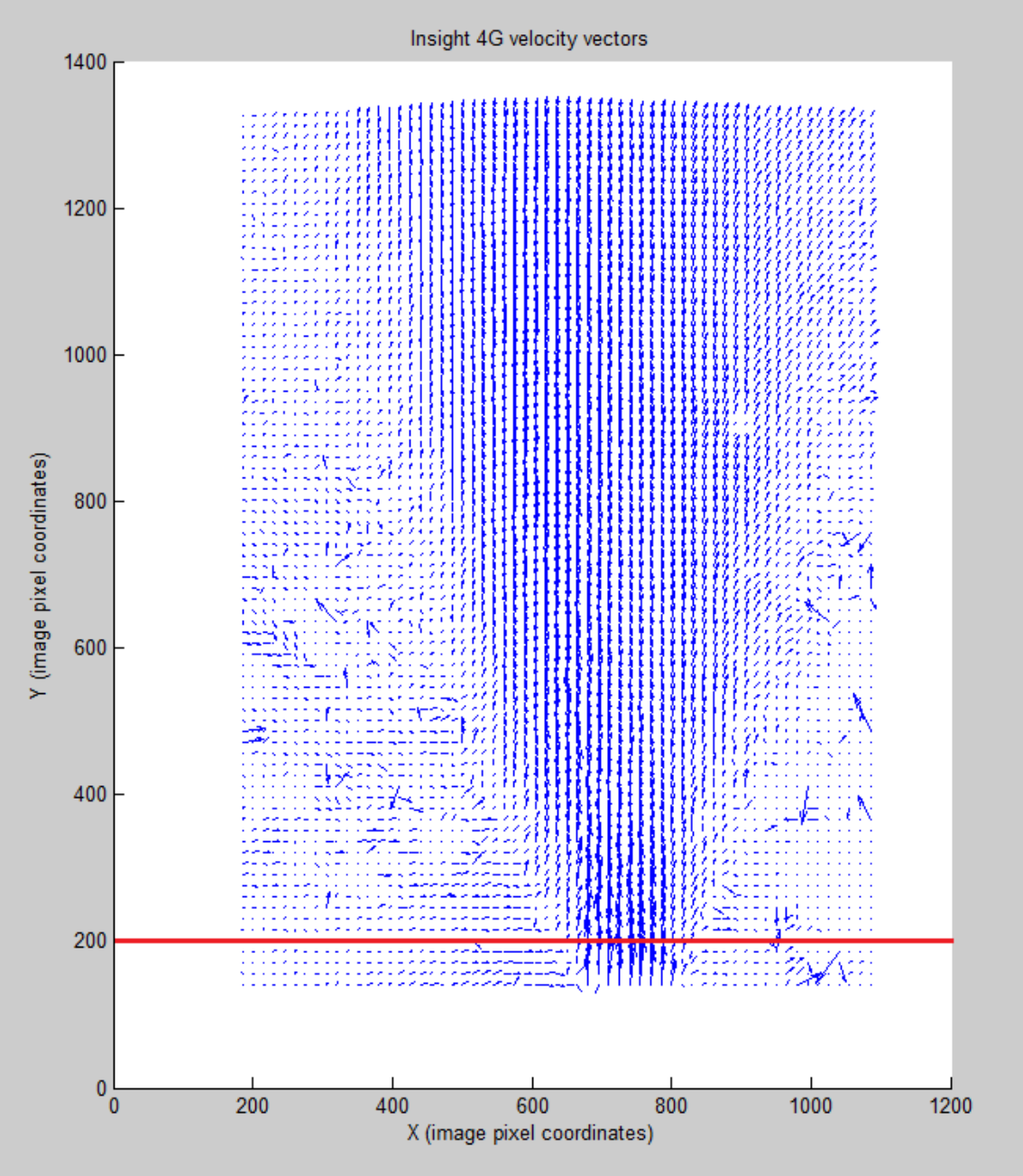


Figure 4.14: Velocity vector field from Insight 4G. Data points are filtered to two standard deviations from the mean.

	Experimental		Insight 4G		PIV Lab	
Segment	$\sigma\ u$	$\sigma\ v$	$\sigma\ u$	$\sigma\ v$	$\sigma\ u$	$\sigma\ v$
1	0.17	0.43	0.20	0.56	0.11	0.41
2	0.17	0.44	0.19	0.62	0.11	0.41
3	0.17	0.47	0.17	0.58	0.12	0.42
4	0.49	0.58	0.16	0.57	0.13	0.48
5	0.56	0.57	0.18	0.58	0.13	0.49
6	0.32	0.49	0.24	0.70	0.10	0.49
7	0.17	0.52	0.24	0.65	0.10	0.50
8	0.17	0.58	0.21	0.57	0.11	0.53
9	0.21	0.75	0.34	0.61	0.11	0.54
10	0.21	0.76	0.34	0.62	0.12	0.54
11	0.17	0.62	0.17	0.61	0.12	0.55
12	0.18	0.60	0.21	0.65	0.11	0.54
13	0.19	0.72	0.23	0.71	0.11	0.52
14	0.18	0.70	0.19	0.75	0.09	0.50
15	0.17	0.70	0.21	0.83	0.07	0.49
16	0.17	0.71	0.25	0.91	0.06	0.49
17	0.16	0.70	0.25	1.89	0.05	0.48
18	0.20	0.70	0.32	1.88	0.04	0.44
19	0.21	0.65	0.35	1.43	0.04	0.36
20	0.19	0.52	0.66	2.30	0.04	0.29
21	0.28	0.56	0.72	3.71	0.06	0.36
22	0.37	0.65	0.47	4.56	0.09	0.69

Table 4.3: Standard deviation of velocities in segment regions. In areas of low velocity (lower numbered segments), deviations are low for all methods. Insight 4G shows a large increase in deviation in the high velocity region (high numbered segments), whereas the other two methods do not.

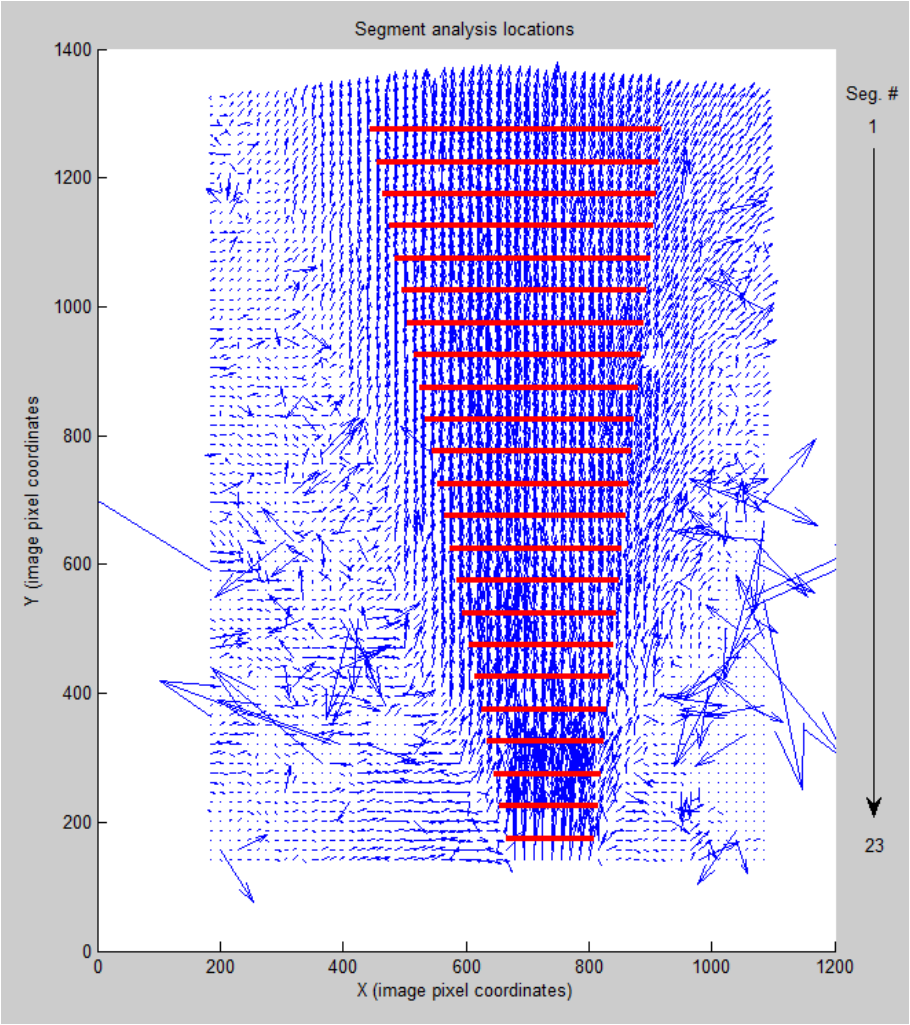


Figure 4.15: Segment locations used to compare horizontal averages across the flow between each program. Outlying background pixels are excluded from the analysis to eliminate noise from the data as much as possible.

Standard deviation The standard deviation for subset data from each discrete line segment is shown in Table 4.3. Examining the SD can help identify whether the output of this program is consistent with itself, as well as the comparison programs. Wild swings in SD from segment to segment would mean that the algorithm was not doing a dependable job of calculating subset displacements. From the data, deviations in the v direction are all steadily within a range of [0.4-0.75] pixels. The maximum deviation between neighboring sections is in segments 8 - 9, where the deviation jumps by 0.18 pixels. The v SD data from Insight 4G indicates quite a few of these jumps, whereas PIVlab has a single large jump at segment 22. In the u velocity direction, the data is not nearly as dependable for

either this program or Insight 4G, however PIVlab shows a very tight deviation range.

It is interesting to note that the professional software shows a much wider fluctuation between segments than this program's or PIVlab. In fact, the variance (SD^2) in v from the combined segments is 1.17 pixels for Insight 4G, whereas both PIVlab and this program display a variance of only 0.1 pixels. Up until segment 17, the variance of all three programs is exactly 0.01 pixels, however the measurements start to fluctuate much more in the higher velocity region in Insight 4G.

Segment average velocities For each section in Fig. 4.15, the subset velocities across each band are averaged for all 200 image pairs in the set. Fig. 4.16 shows the v velocities, and Fig. 4.17 shows u.

In v, the profiles up to around segment 10 are very similar, showing a steady rise in velocity. From this point, each program shows varying levels of slope decrease up until section 18, where all three velocities increase sharply. The profiles of the data from Insight 4G and PIVlab more closely mimic one another than the data from this program, however they differ by about $\frac{3}{4}$ to 1 pixel in magnitude. This program's output follows along more closely with Insight 4G in magnitude of v velocity, and the profiles increasingly overlap as the velocities become higher.

The velocity profiles in u from Fig. 4.17 do not show any discernible trends. The horizontal flow for this program and PIVlab both average around zero pixels, however Insight 4G reports that the flow in u throughout the vertical sections is slightly rightward by about $\frac{2}{3}$ of a pixel.

4.5.4 Horizontal plume profile

The previous data is used to show the comparability between the data across the range of the profile from top to bottom, and does not use a typical PIV profile comparison. In a standard PIV analysis, a cross-sectional profile at a certain region will typically be used to show the flow characteristics. In Fig. 4.18, the cross-section taken from Fig. 4.15 is shown, this time the data points are averaged in the y-axis using five (5) vertical neighbors to smooth the data. The result is a velocity profile across the band of the mist moving upward.

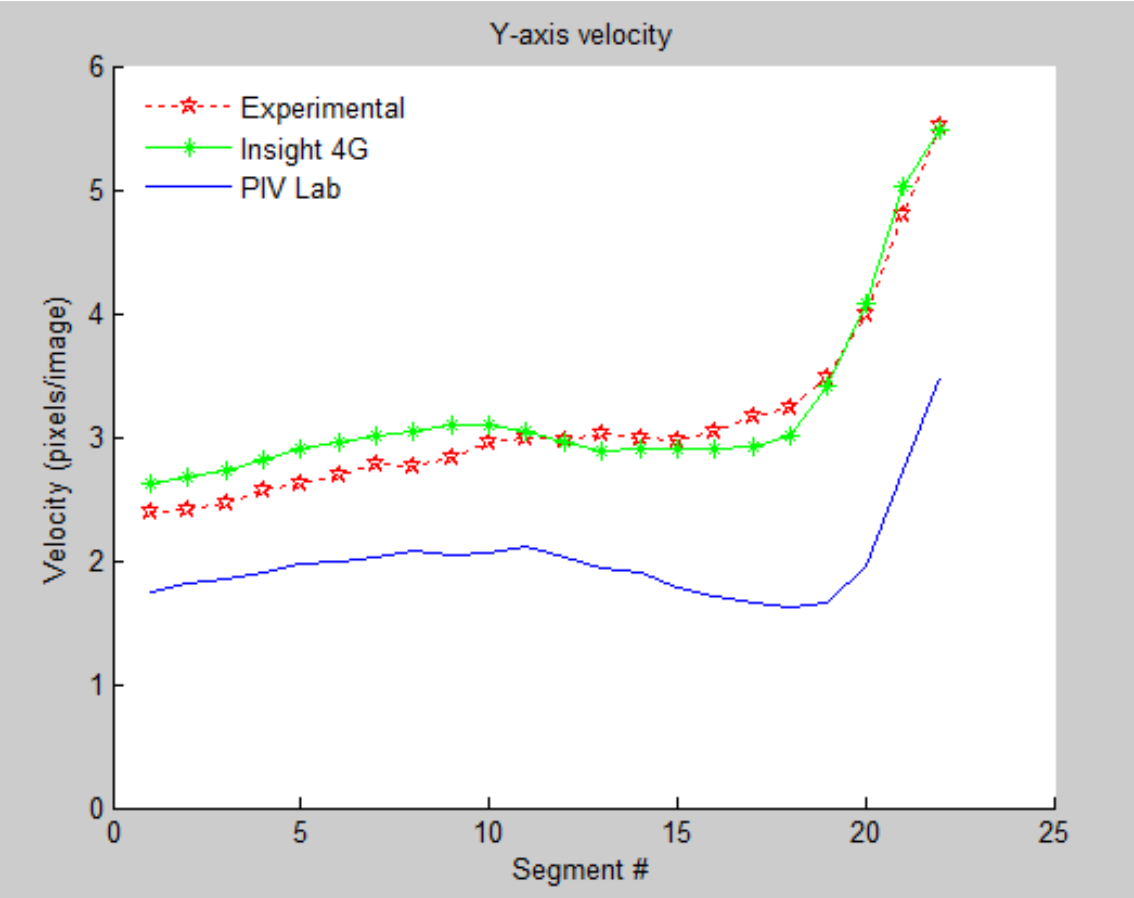


Figure 4.16: Velocity in the y-axis at segment locations. The velocity profiles are similar however the magnitudes differ. The magnitude of this program closely matches Insight 4G, however the profile does not match as well as PIVLab.

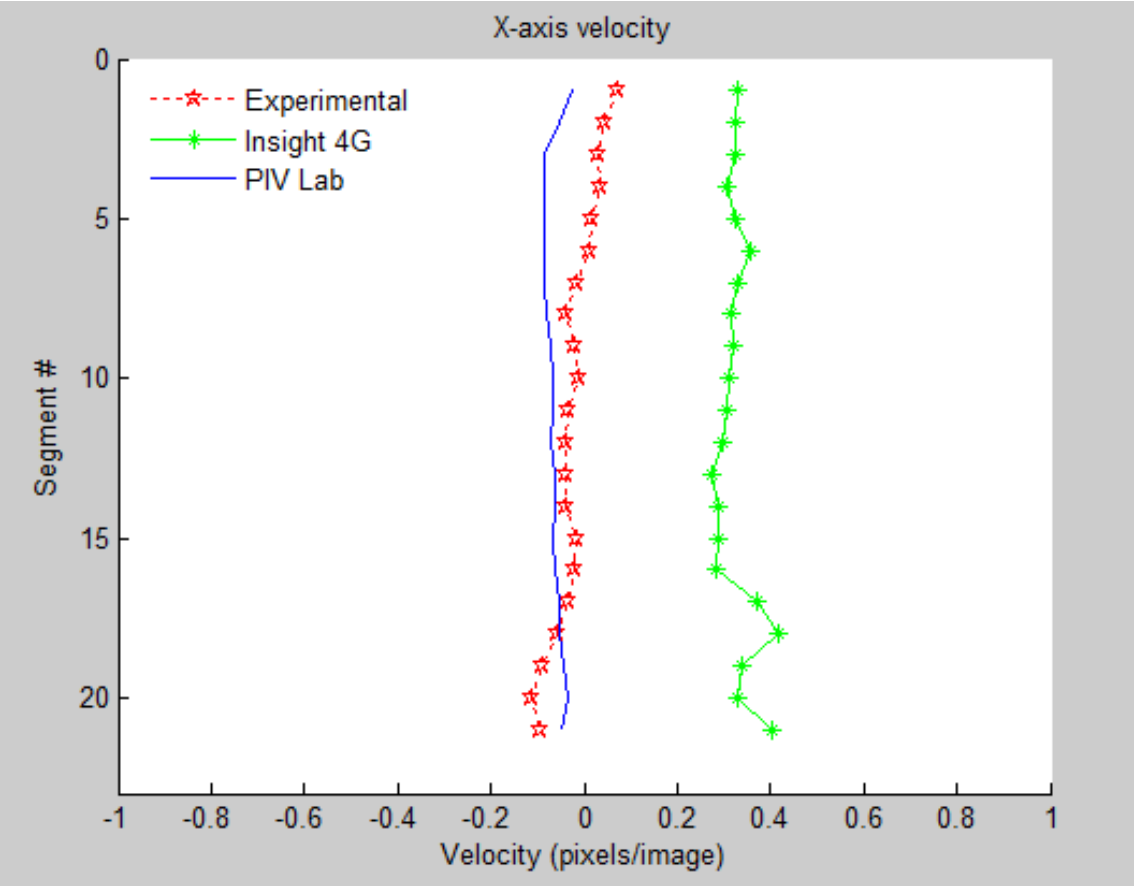


Figure 4.17: Velocity in the x-axis at segment locations. The velocities follow a similar profile, yet like y-axis velocity, are different in magnitude.

Discussion of output Particle flows are clearly difficult to measure accurately and consistently. The amount of data is a problem, because there are statistically few data points available for comparison, due to the sporadic nature of correlation success. Both PIVlab and the commercial Insight 4G disagree with each other. The magnitude of velocities in both u and v differ greatly, although they do see some similarities in profile. The data from this program falls somewhere in the middle of the two, which is a promising outcome that shows that while the algorithm does have deficiencies, it also has a good success rate in comparison to established programs.

For further comparison and analysis, single image pair flow patterns from the previous PIV analysis are captured and displayed in Appendix B. In Appendix C, the output of this program is compared against PIVLab for two other patterns: a digitally induced swirl pattern, and the flow extruding from a Karman vortex sheet.

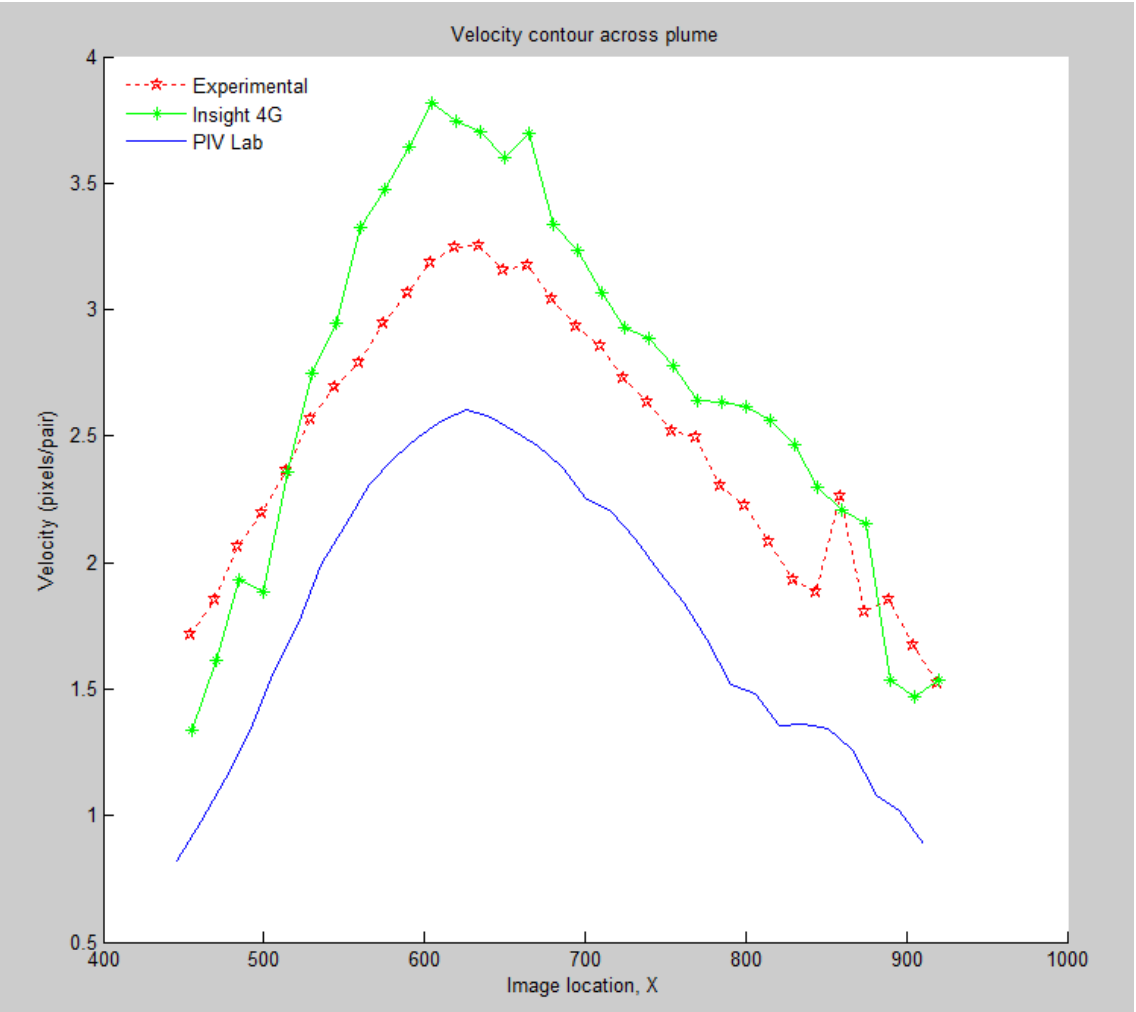


Figure 4.18: Cross-sectional flow profile at segment # 5 from Fig. 4.15. The plot trends are similar, with differing magnitudes. This program agrees in velocity magnitude with Insight 4G, however the trend of PIVLab is closer.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

A software program combining digital image correlation and particle image velocimetry has been developed that provides a suite of features and functionality appropriate for an undergraduate teaching environment. The graphical user interface (GUI) is developed such that an instructor or student user may use the program without prior knowledge of its operation by providing on-screen guidance and instruction. The program provides easy access to help files describing the operation of PIV and DIC systems. The program runs quickly and efficiently, and provides feedback to the user in the form of on-screen vector (PIV) and contour (DIC) fields.

Experiments is performed at New Mexico Tech in both material deformation and particle flow verify that the equations and algorithms used in the program are accurate and produce reliable results. DIC results, measured against a commercial program, are found to be accurate to within $\frac{1}{8}$ of a pixel on average, up until the point of high deformation and significant decorrelation of the speckle pattern. PIV results, when compared against two programs, showed on average that the velocities (excluding background regions) stayed within $\frac{1}{2}$ a pixel of the commercial program, and within 1 pixel of the free program. These results are well within the range of accuracy for the target audience.

5.1 Future work

During development, compromises were made due to the programming experience of the author, as well as the extensive development inherent in developing a sophisticated, algorithm based GUI computer program. The current state of the program is satisfactory for the application, however as in any piece of software, there is always room for improvement.

On-screen calibration Currently, the graphical user interface (GUI) provides feedback on the location of the mouse pointer in the image. This gives the user a tool for calibrating the data by hand. Ideally, there should be a tool that provides a line-drawing function to calibrate the pixel dimensions in real-time.

PIV multi-pass The most accurate method of assessing particle movement via phase transform is using a multi-pass approach. Currently, only a single pass option is proffered.

Affine transformation of the subset The subset-based approach does not deform the kernel in the search, which causes lost correlation points and also does not provide information about rotation in the sample. An interpolation-based method should be offered on top of the existing method that provides this level preciseness. The current method is a great way to perform a really fast examination, however it can struggle to operate efficiently in highly deformed areas.

Fourier calculation The fast Fourier transform programming library used is an unoptimized approach. A more sophisticated Fourier library should be used to increase the speed and effectiveness of the PIV search algorithm.

PIV algorithm improvements While the current PIV approach works sufficiently, the baseline data comparison shows that there are still improvements to be made. A more detailed study and implementation of published PIV algorithm research is desired.

Improved analysis feedback The current software provides a basic feedback of the results of each analysis. However there are many more results parameters that could be detailed in a more effective manner, such as plots or charts.

The software developed for this project is ready to use in a classroom or lab environment out of the box. It also has an extremely strong foundation, and is primed for expansion and upgrade of the core components. Due to its uniqueness in the current market, there is real opportunity for it to be used in its current form, and upgrades to the algorithms and features can make it a tool to be used in many more applications.

REFERENCES

A. Arnott, G. Schneider, K.P.Neitzke, and B. Sammler. Multi-window piv for high-lift measurements. Instrumentation in Aerospace Simulation Facilities, 2003.

N. Athreyas, Z. Lai, J. Gupta, and D. Gupta. Analog signal processing solution for image alignment. In *Third International Conference on Advanced Information Technologies and Applications*. Newlins Inc, 2014.

H. A Bruck, S. R. McNeil, M. A. Sutton, and W. H. Peters. Digital image correlation using newton-rhapson method of partial differential correction. *Experimental Mechanics*, 29:261–267, 1989.

M. Debella-Gilo and A. Kaab. Sub-pixel precision image matching for measuring surface displacements on mass movements using normalized cross-correlation. *Remote Sensing of Environment*, 115:130–142, August 2011.

Hana Druckmullerova. *Phase correlation: the mathematical background and application to image registration*. LAP Lambert academic publishing, 1st edition, 2011.

FFMPEG.org. FFMEG resources and download link, 2014. URL `https://ffmpeg.org/`.

J.R. Higgins. *Sampling Theory in Fourier and Signal Analysis*. Oxford University Press, Oxford, 1st edition, 1996.

TSI Inc. Insight 4g product website, 2017. URL `http://www.tsi.com/products/`.

Open Source Initiative. 2-clause bsd license, 2017. URL `https://opensource.org/licenses/bsd-license.php`.

D. lecompte, A. Smits, and Sven Bossuyt. Quality assessment of speckle patterns for digital image correlation, 2005.

D. Lecompte, S. Bossuyt, S. Cooremna, and H. Sol. Study and generation of optimal speckle patterns for dic. society for experimental mechanics, 2007.

H. Lu and P.D. Cary. Deformation measurement by digital image correlation: implementation of a second-order displacement gradient. *Experimental Mechanics*, 40:393–400, 2000.

Mathworks. Matlab home page. URL `www.mathworks.com`.

Matlab. Matlab fft implementation, 2017. URL https://www.mathworks.com/help/matlab/ref/fft.html.

MIT. Fastest fourier transform in the west, 2017. URL http://www.fftw.org/.

H. Nobach and M. Honkanen. Two-dimensional gaussian regression for subpixel displacement estimation in particle image velocimetry or particle position estimation in particle tracking velocimetry. *Experiments in Fluids*, 38: 511–515, 2005.

Openpiv. Open piv product information, 2017. URL http://www.openpiv.net.

U. Oslo. Matpiv homepage, 2017. URL https://www.mn.uio.no/math/english/people/aca/jks/matpiv/.

B. Pan. Reliability-guided digital image correlation for image deformation measurement. *Applied Optics*, 8(48):1536, March 2009.

B. Pan and K. Li. A fast digital image correlation method for deformation measurement. *Optics and Lasers in Engineering*, 49:841–846, February 2011.

B. Pan, H. M. Xie, B. Q. Xu, L. Xiong, and G.D Liu. Performance of iterative gradient-based algorithms with different intensity change models in digital image correlation. *Optics and Laser Technology*, 44:1060–1067, November 2012.

Correlated Solutions. VIC-2D product website, 2017. URL http://correlatedsolutions.com/vic-2d//.

A.M.R. Sousa, J. Xavier, and M. Vaz. Cross-correlation and differential technique combination to determine displacement fields. *Strain*, 201:87–98, 2011.

W. Thielicke. *The flapping flight of birds - analysis and application*. PhD thesis, 2014. URL http://irs.ub.rug.nl/ppn/382783069.

W. Thielicke and E.J Stamhuis. Pivlab - towards user-friendly, affordable and accurate digital particle image velocimetry in matlab, 2014a.

W. Thielicke and E.J Stamhuis. Pivlab - time resolved digital particle image velocimetry tool for matlab (version: 1.41), 2014b. URL http://dx.doi.org/10.6084/m9.figshare.1092508.

S. Yoneyama, A. Kitagawa, K. Tani, and H. Kikuta. Bridge deflection measurement using digital image correlation. society for experimental mechanics, 2006.

Z. Zhang, Y. Kang, H. Wang, Q. Qin, Y. Qiu, and X. Li. A novel coarse-fine search scheme for digital image correlation method. *Measurement*, 39:710–718, March 2006.

Y. Zhou, B. Pan, and Y.Q. Chen. Large deformation measurement using digital image correlation: a fully automated approach. *Applied Optics*, 51:7673–7678, 2012.

APPENDIX A

PROGRAM FEATURES AND FUNCTIONS

The software created for this project analyzes images to track the motion of particles in either fluid flow or solid materials, in order to develop a full-field strain or velocity profile. This chapter overviews the user input screens in the GUI, the options available to the user to set up the program or change parameters, and the overall flow of the program operation.

A.1 Program overview

A Java software program can come in many different installation flavors. A program may require only a simple double-click of the executable Java file (.jar extension), which contains all data and resources needed to operate, or else a more intensive procedure where many files and libraries are unpacked an copied to the installation folder is required. This program uses the former, where only a single file is needed to operate the program. A still-frame capture feature to extract frames from video is included which requires a single add-on file from an open-source third party.

Once the program is executed, the first interface that opens lets the user select between PIV and DIC, and provides the path to the folder containing the images.

After these selections are made, the main screen for either DIC or PIV opens. The user then selects the appropriate input parameters and commences the run.

When the evaluation is complete, more runs using different parameters may be commenced, or the raw data exported and the program exited.

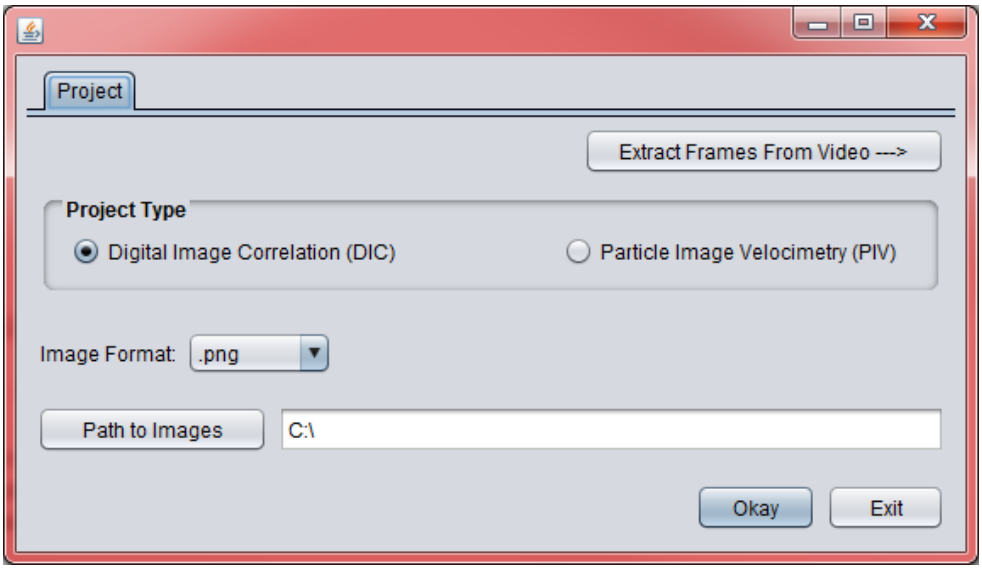


Figure A.1: Program entry point interface. The user selects between DIC or PIV, indicates the image format and path, and also can enter the video fram extraction tool.

A.2 Program entry and still frame extraction

Upon starting the program, the user may choose between entering the main program with either DIC or PIV functionality, seen in Fig. A.1. The following image formats are supported:

- JPEG
- PNG
- TIFF
- GIF

Images may be in the following 8-bit representations: black/white, grayscale, or three-plane RGB (the program will automatically convert a three-plane image to its grayscale counterpart). Images stored in raw format must first be processed into their 8-bit representations for the program to have success.

A feature of the entry screen is an option to use an integrated utility to extract still frames from video files. The tool requires the use of an open-source Java library called FFMPEG, which is both provided with this software package and can be downloaded from the FFMPEG website [FFMPEG.org, 2014].

The extraction interface is designed to be quick and easy to use, shown in Fig. A.2. The user only needs to provide folder paths to the input video, output

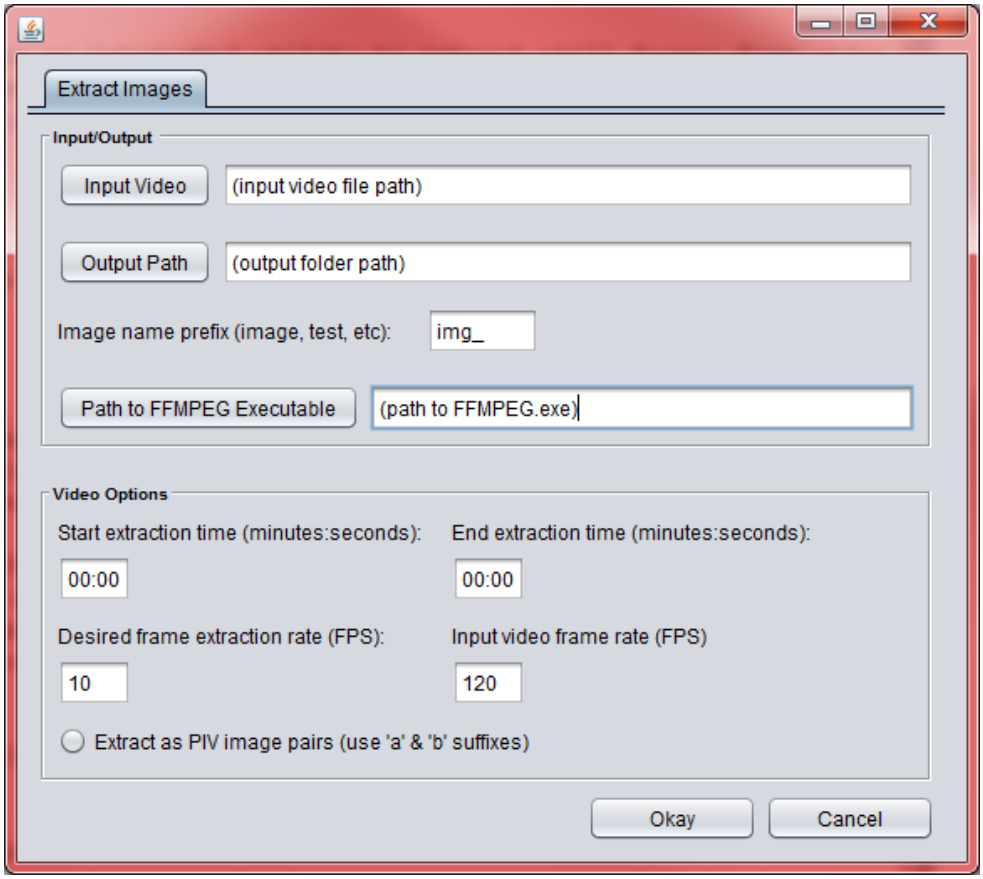


Figure A.2: The image extraction utility provides a method to extract still frames from a video, with the option to label as PIV image pairs.

folder, and FFMPEG location. There is also an option to prefix each frame with a unique string identifier to differentiate between different tests and videos.

The extraction options are limited to keep the process streamlined. The user may input a starting and ending time from the video to only extract a certain segment of stills, and how many frames to remove per second. It is recommended that the user know the input video frame rate per second (FPS), so that extra images aren't extracted. If the input rate is higher than the actual video FPS, extra images will be extracted that are duplicates of others, as there are only a limited amount of frames available. Also provided is the option to pull out the stills as PIV pairs, which accomplishes two things: the first is the program will take out two successive frames at the indicated extraction interval, and the second is that the image pairs will be labeled with "a" and "b" suffixes.

A.3 Main program functions

A.3.1 Main interface

The main interface, shown in Fig. A.3, is called when the program entry screen exits. The program can be run entirely from this interface, with secondary screens available to adjust algorithm search variables, data export options, and runtime environment. The program enters with the first image in the collection in the view screen. The subset default size is 21 x 21 pixels, initially centered in the upper half of the image. The user may then resize the kernel by grabbing the small square located at the bottom right side of the subset representation, so that the subset envelops the desired speckle pattern size. The kernel can also be moved around the screen so that the user can find the appropriate place to make the measurement in the image.

A.3.2 Guidance viewport

The guidance viewport, located in the upper left-hand corner of the screen shown in Fig. A.4 (a), helps the user step through the process so that the program may be set up with no previous experience as quickly as possible. The view provides step-by-step instructions to set up the appropriate kernel size and select a region of interest.

Once the program starts running, the guidance viewport switches over to an output of the analysis, shown in Fig. A.4 (b), so the user can keep track of the progress of the program. Also provided are progress bars for image correlation (top) and subpixel analysis (bottom).

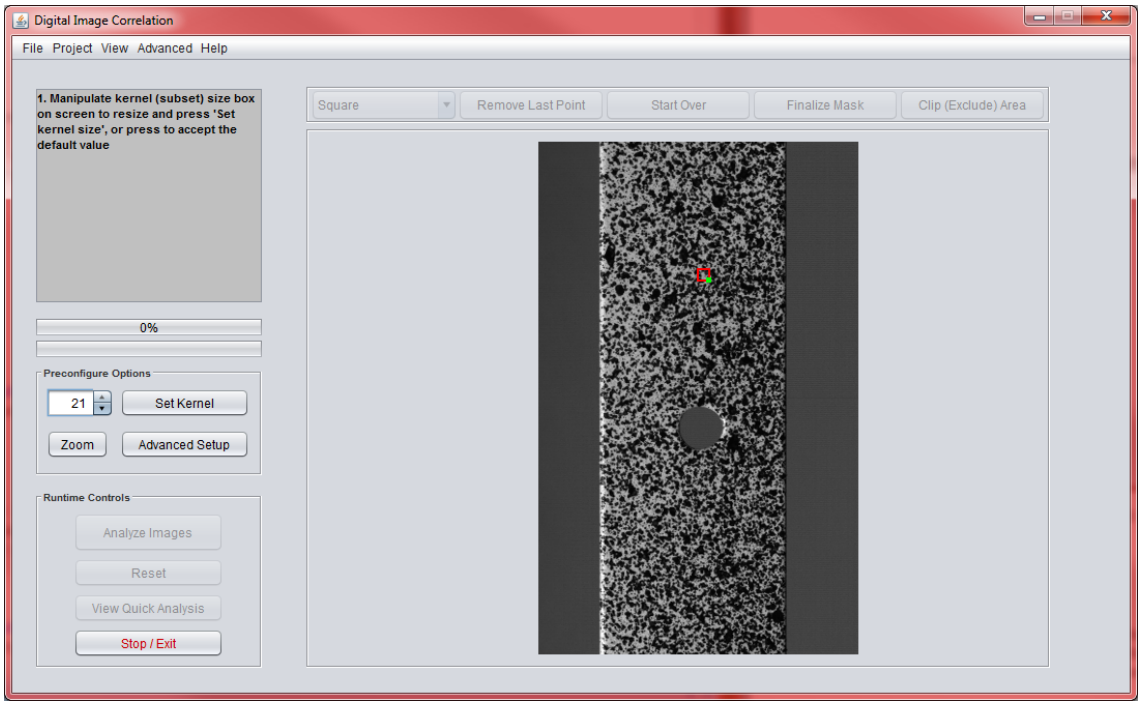


Figure A.3: Main screen contains all features and interactions needed to run the program, as shown upon first entry.

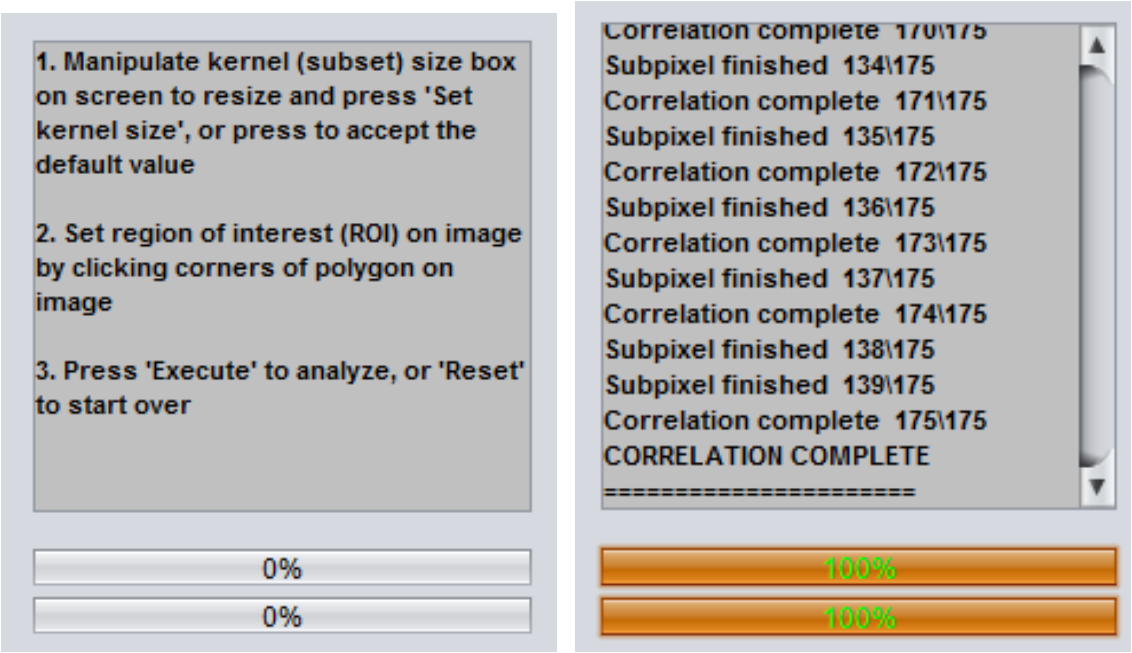
A.3.3 Preconfigure options

The preconfigure options, shown in Fig. A.5, provide several functions:

- The kernel size may be adjusted to an exact number manually using the spinner widget.
- Zoom feature increases the image size in a separate screen and places the adjustable kernel box on that image so that the user can more precisely identify the appropriate subset size.
- The advanced setup button takes the user to more adjustable features used in the analysis

A.3.4 Run-time controls

Basic start, stop and exit functions are provided in the runtime controls box seen in Fig. A.6. The buttons are be enabled and disabled at the appropriate times, so that the user has only the option of pressing the correct buttons at the correct



(a) (b)

Figure A.4: The guidance viewport helps the user move through the proper steps in configuring the program. (a) shows the pre-run viewport configuration, and (b) shows post-run output.

times. Otherwise, unwanted results may occur when the program's threads are interrupted unsafely.

The "View Quick Analysis" button is enabled after the run is complete so that the user may see view a report on the number of correlated points, poorly correlated points, success rate, and average displacements, shown in Fig. A.7. The benefit of having instantaneous access to this data provides the user with insight into the correlation process. The user may decide to readjust the setup parameters and re-run the analysis after viewing these results.

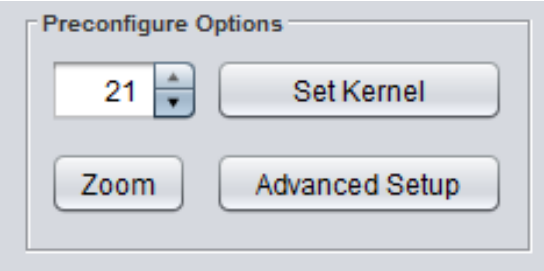


Figure A.5: The preconfigure controls provide the user with tools to set up the analysis.

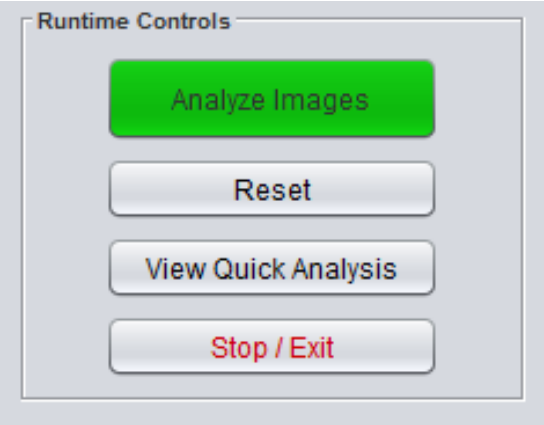


Figure A.6: Runtime controls provide the user with options for starting, stopping and resetting the analysis, as well as viewing the quick analysis results when a run has finished.

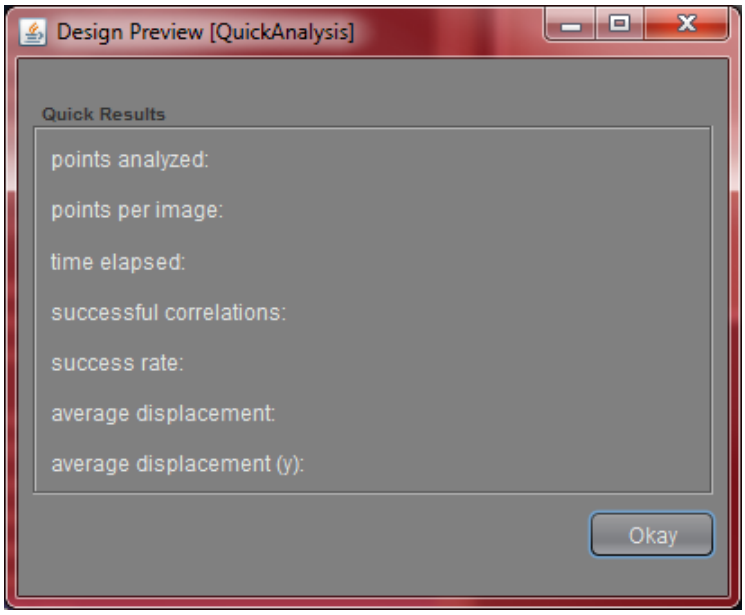


Figure A.7: Quick results provides the user with an instant view of the success parameters of the correlation analysis.

A.3.5 Region of interest button bar

Located above the image on the main screen, the region of interest buttons provide the user with easy ways to define an ROI on the image. The user has two options to draw an ROI:

1. By-hand selection of polygon points. The user may select a complicated region of interest by clicked any area of the image and creating an multi-vertice polygon.
2. Preconfigured shapes in the drop-down menu are provided to provide the user with quick, easy, and perfect shapes:
   (a) Square (holds the length and width equal).
   (b) Rectangle
   (c) Circle
   (d) Ellipse

The shapes are resizable to any dimension on the image, and the user may also drag the shape around the screen to the appropriate location.

The "Finalize Mask" and "Clip Area" buttons are enabled in the following sequence:

1. Once the user has closed the by-hand polygon, or placed a preconfigured shape on the screen, the "Finalize Mask" button (as well as the "Start Over" button) becomes enabled.
2. The program inquires from the user whether they would like to clip out an area of the ROI. This is useful when analyzing samples such as the dogbone in the sample image where a hole has been excised. The user then selects another ROI using the tools provided to clip from the selection. After ROI completion, the buttons are again disabled, and the Runtime Controls become enabled.

A.3.6 Menu bar

The menu bar provides the following features as separate tabs:

- **File** Exits the program, while saving the current ROI configuration to disk for later use.
- **Project**
    1. **Use last mask** puts onscreen the last ROI mask the user configured in the current session
    2. **Use previous session mask** provides the same function as with using the last mask, however will pull the last ROI used from disk. This is useful if the user wants to quickly call up the same mask used in a previous session.
    3. **Export to CSV** calls a save file interface so the user can export the analysis to disk in comma separated value (CSV) format.
- **View** The View sub-menus provide the following functionality:
    1. **Show cursor coordinates** provides the user with the x-y pixel location in the image coordinates. This is helpful if the user is trying to pick a specific portion of the image for the ROI.
    2. **Contour map axis** Affords the user with the option of viewing the image deformation in either the vertical or horizontal axis. The contour plot is provided by the legend, measured in pixels.
    3. **Contour map output** Contributes the option to show incremental deformation in the contour map, or total distortion. Choosing incremental distortion shows the deformation between each image, wheres total deformation shows the output from the first image.
- **Advanced** In the Advanced menu the user may go to the Advanced Setup interface (the same link as on the main screen in Preconfigure Options). The resolution may also quickly be changed here between subpixel and integer-pixel resolution.

- **Help** Provides a link to the main help screen which illustrates in depth how to operate the program, as well as an explanation of the variables available for the user to change.

A.3.7 Advanced setup

The Advanced interface consists of three tabs, outlined in the following section, in which the user may change the operational input parameters of the algorithm. Advanced setup is either accessed from the main interface, or through the menu bar.

Preconfigure As seen in Fig. A.8, the user may adjust the following input variables:

1. **Step size** is defaulted at five pixels, which means the subset box will step five pixels after each completed correlation. The user can change this from 1 up to a maximum of 100.
2. **Kernel size** can also be adjusted here, as well as on the main screen.
3. **Minimum correlation coefficient** determines the minimum threshold that the algorithm decides is a proper correlation match.
4. **Reference image update** is defaulted to "off," however the user may change that here. Also the number of bad correlation points before the reference image is updated is inputted.

Resolution Offers the option to toggle sub-pixel resolution analysis on and off. When on, there are three sub-pixel methods to choose from, as discussed in Chapter 2:

- Lagrange
- Gaussian
- Parabolic

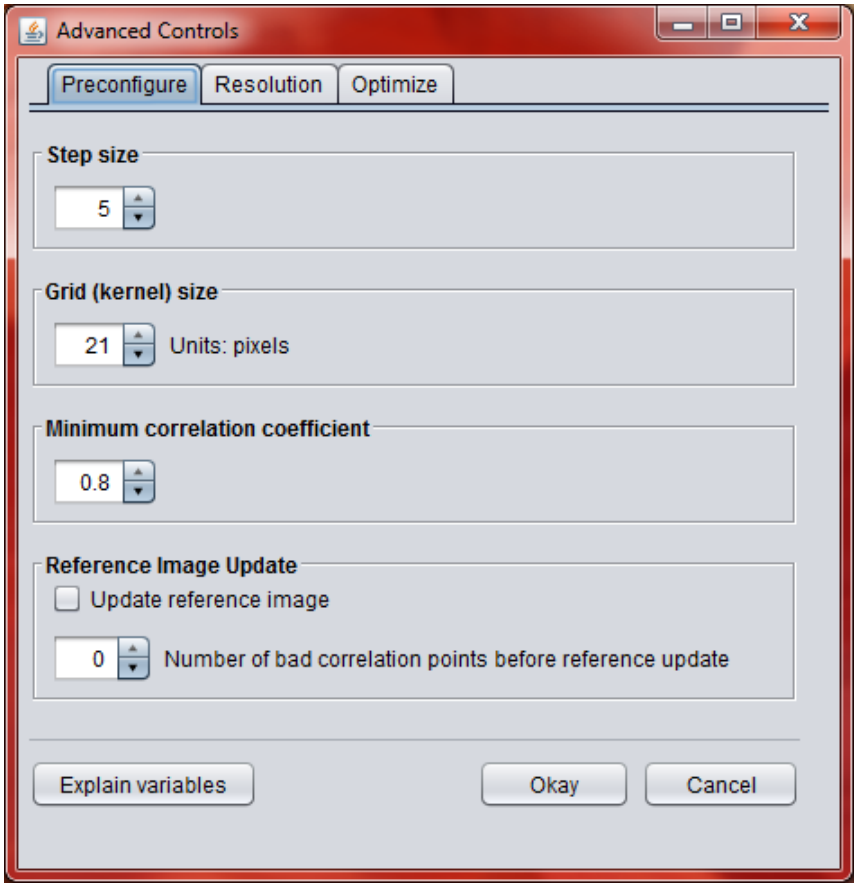


Figure A.8: The preconfigure interface provides interaction with the correlation algorithm input variables.

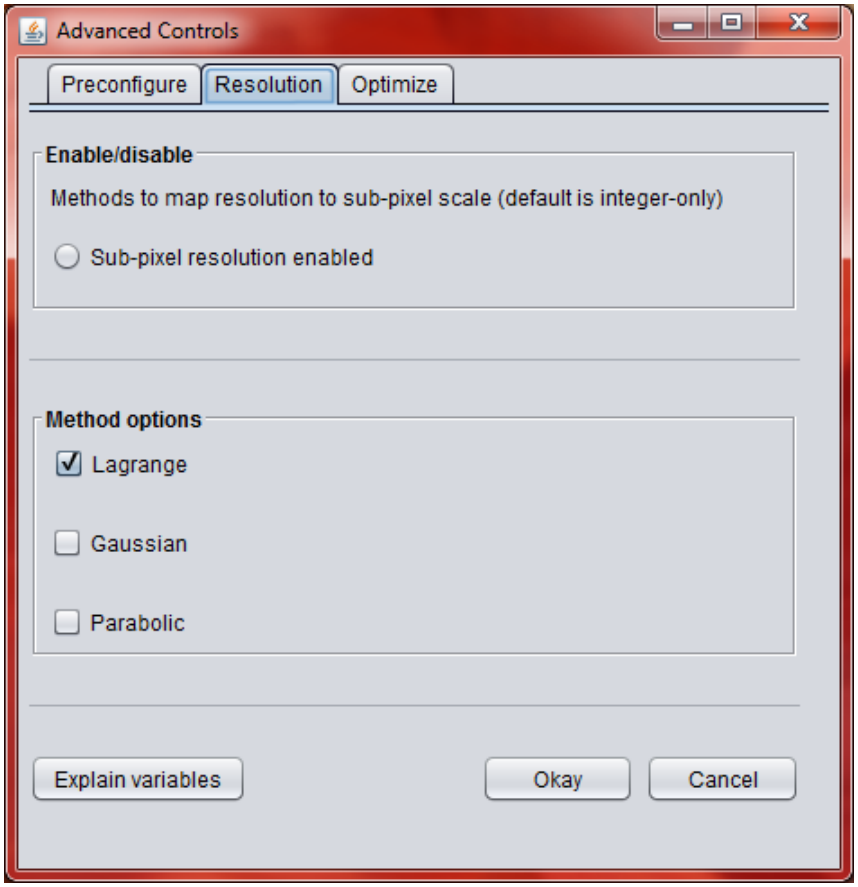


Figure A.9: The resolution tab provides options for sub-pixel resolution scanning of the image. Subpixel may be turned off completely, or toggled between options.

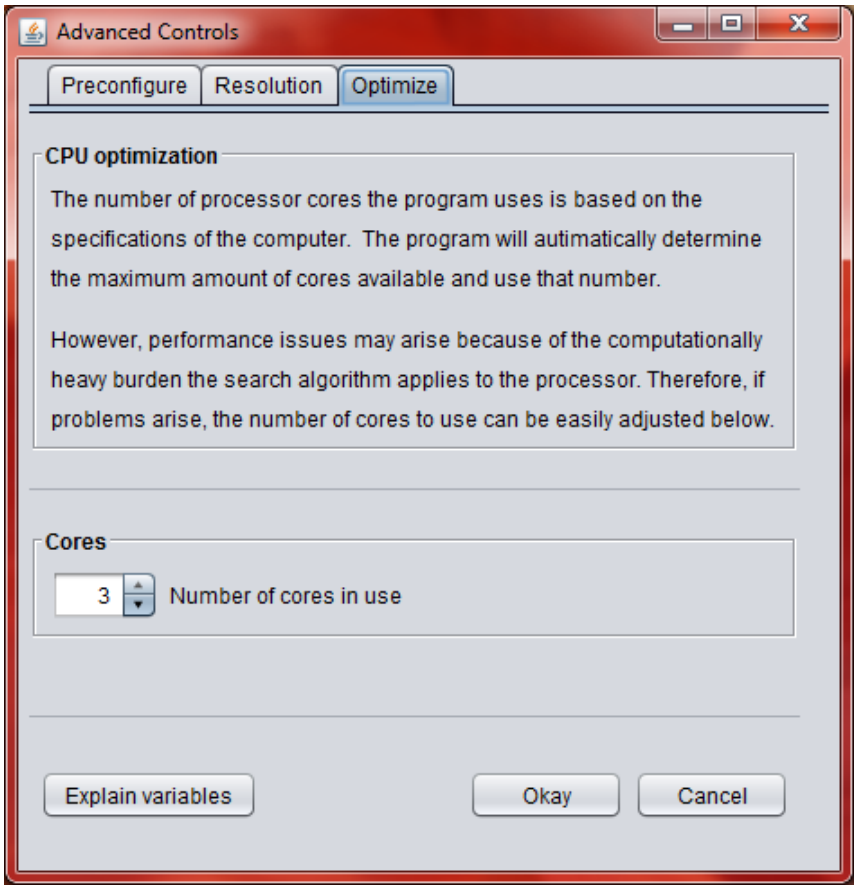


Figure A.10: The Optimize tab allows the user to select the number of cores (threads) among which to split the image analysis.

Optimize In this interface, the number of processor cores the user would like to utilize in the correlation analysis is indicated, seen in Fig. A.10. The default is set to the maximum available cores, adjustable down to one core. Technically "cores" is a misnomer, because the amount of processors in use does not change. What this function does is tell the algorithm how many threads to split the image analysis amongst. If there are four (4) cores present, the algorithm will take four sections of image and work on them at once in different threads. If that number is changed to three, the image will be analyzed in three concurrent sections, and so on down to a single thread.

APPENDIX B

SINGLE IMAGE PAIR PIV ANALYSIS

The following images display the velocity vector fields for four arbitrary image pairs captured at various intervals in the PIV validation experiment from Chapter 4. Noise in the background pixel regions of the images has been reduced for all data sets using a vector rejection filter of magnitude [-10 0] in the vertical direction, as the flow is dominated by upward movement (negative y-axis in image coordinates), with a maximum average of six pixels/image pair in the highest-velocity region at the exit of the particle vent. Horizontal noise is reduced using a vector rejection filter of magnitude [-5 5].

B.1 Vector field comparison

The output of this program and Insight 4G show strong resemblance, with a more dispersed and erratic vector profile than output from PIVLab through the high-flow velocity cone. Overall, the program outputs are similar, showing higher velocities at the jet exit and a vortex region in the x,$y \approx$ (650, 1020) region of the plume.

B.2 Cross-sectional comparison

To quantify instantaneous velocity profiles of each image pair, the velocities across the plume at y = [1160 1200], x = [430 930] are used (smoothing the data by averaging three subsets in the vertical direction for each x component). As seen from the vector plots in Figs. B.1- B.12, there appears to be high variation in the data in each image pair. The data in Figs. B.13 and B.14 confirms this. The moisture particulate flow used in this experiment is highly variable, and also suffers from a low diversity in light intensities and particle sizes. These factors cause the phase correlation process to produce varying results when viewed instantaneously, as the programs used here all feature unique implementations.

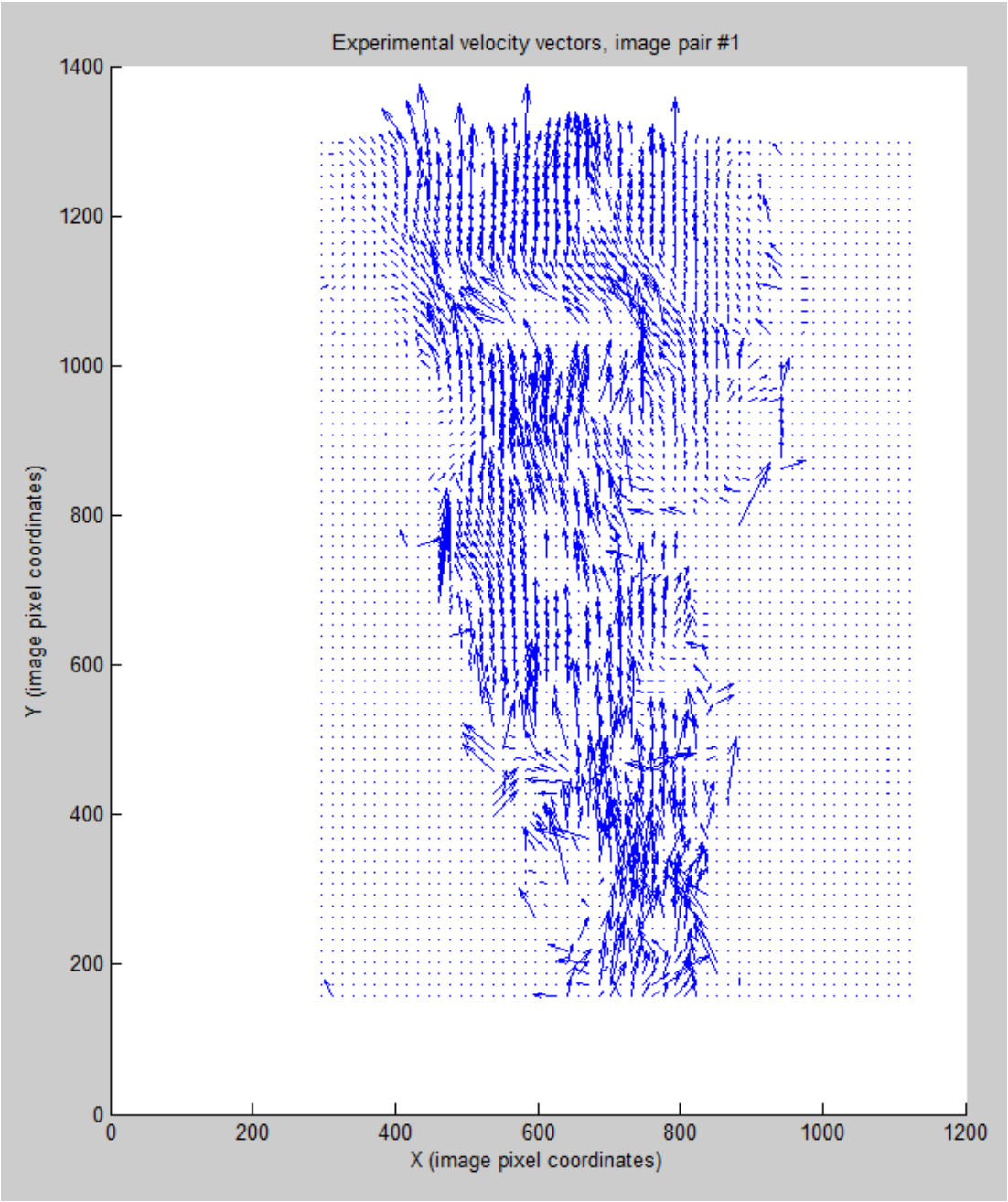


Figure B.1: Vertical flow analysis for this program, image pair # 1/200.

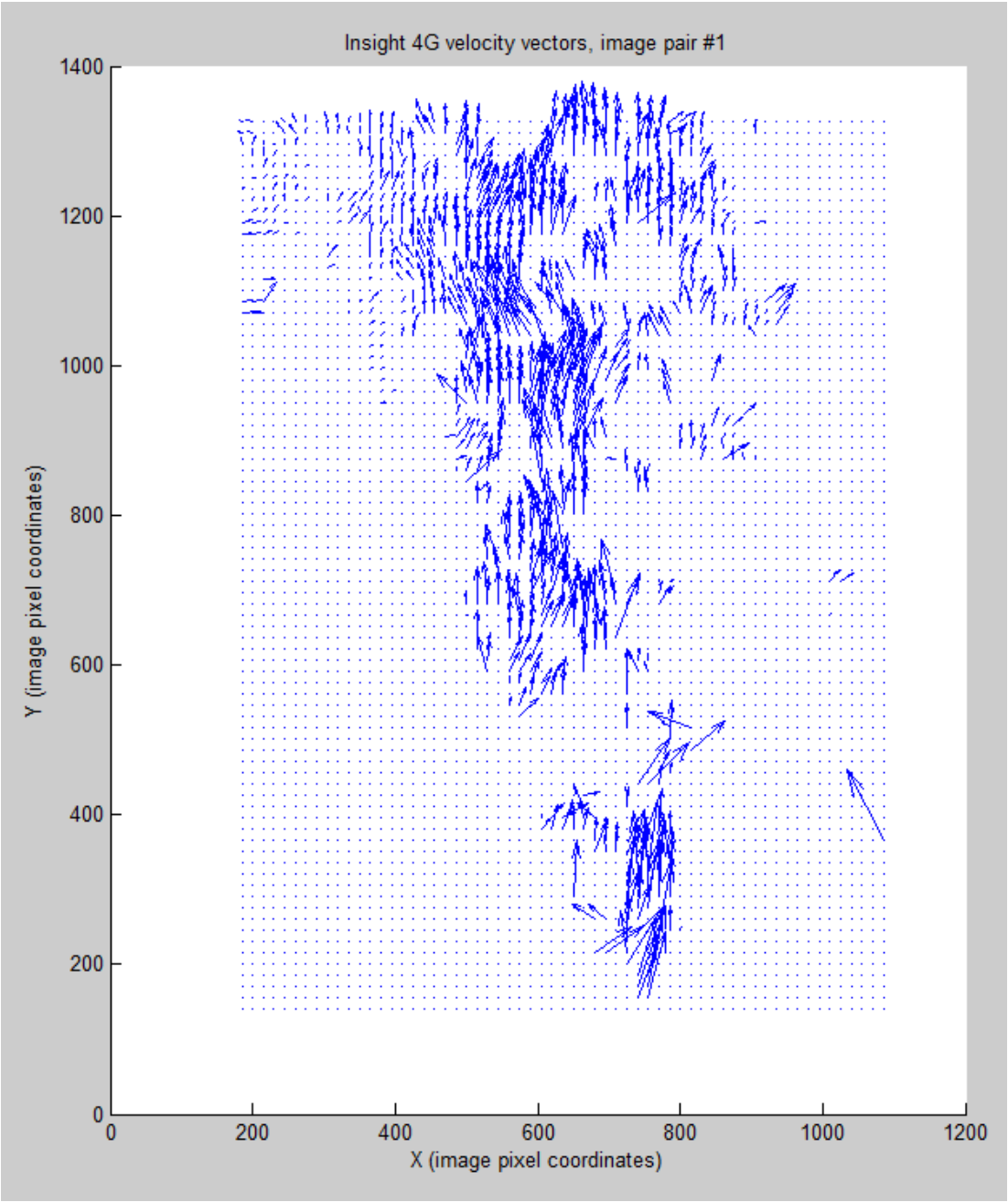


Figure B.2: Vertical flow analysis for Insight 4G, image pair # 1/200.

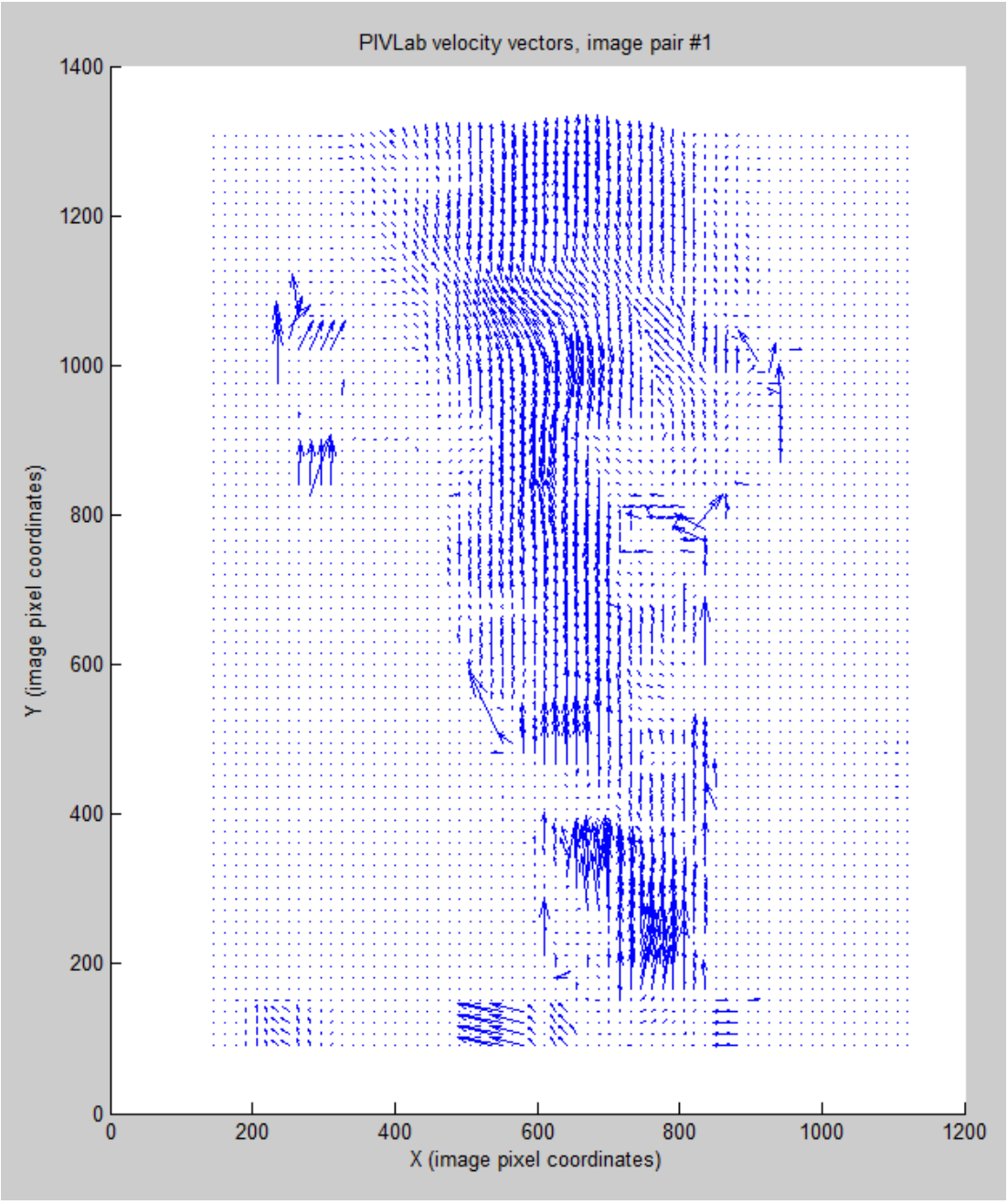


Figure B.3: Vertical flow analysis for PIVLab, image pair # 1/200.

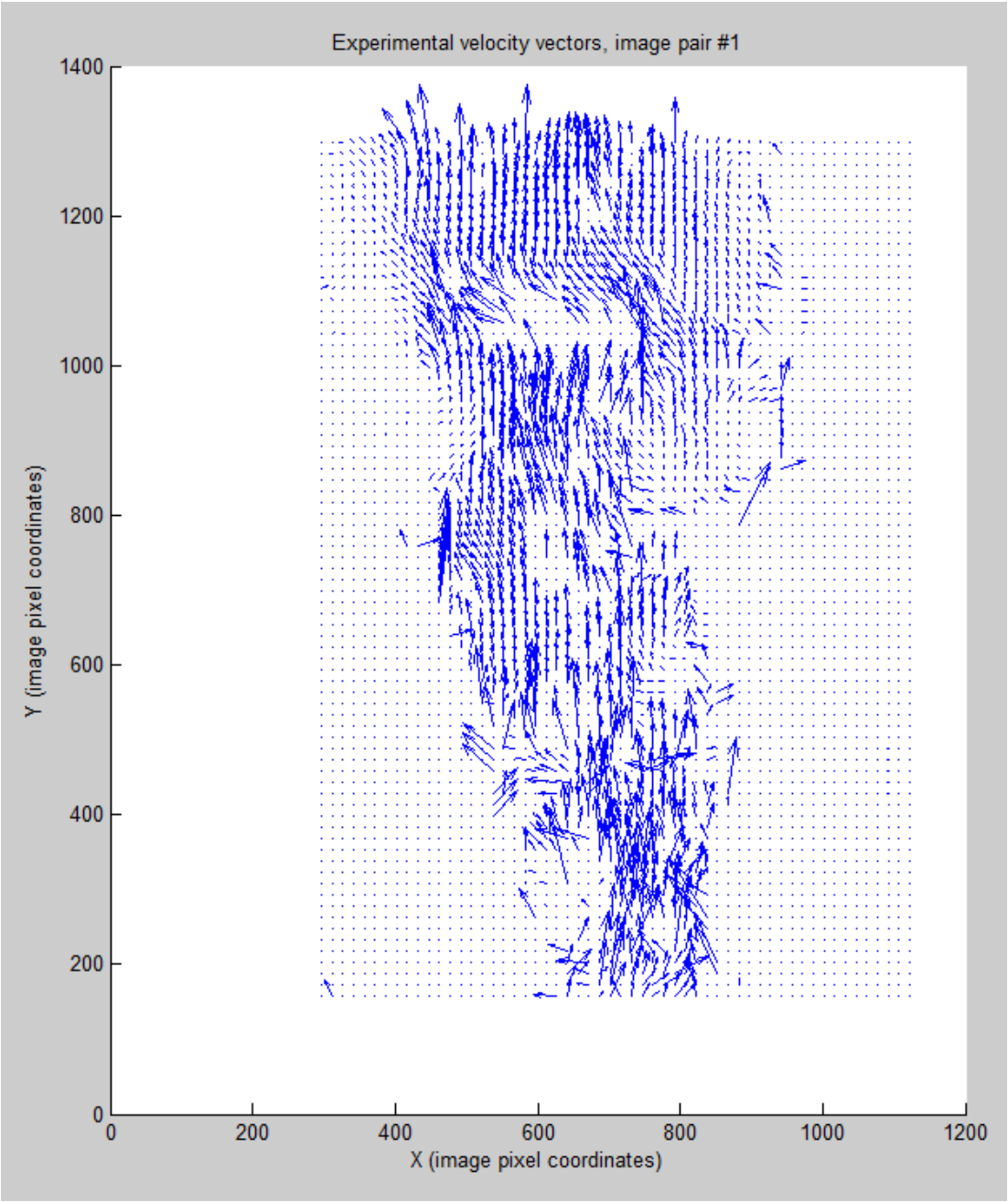


Figure B.4: Vertical flow analysis for this program, image pair # 51/200.

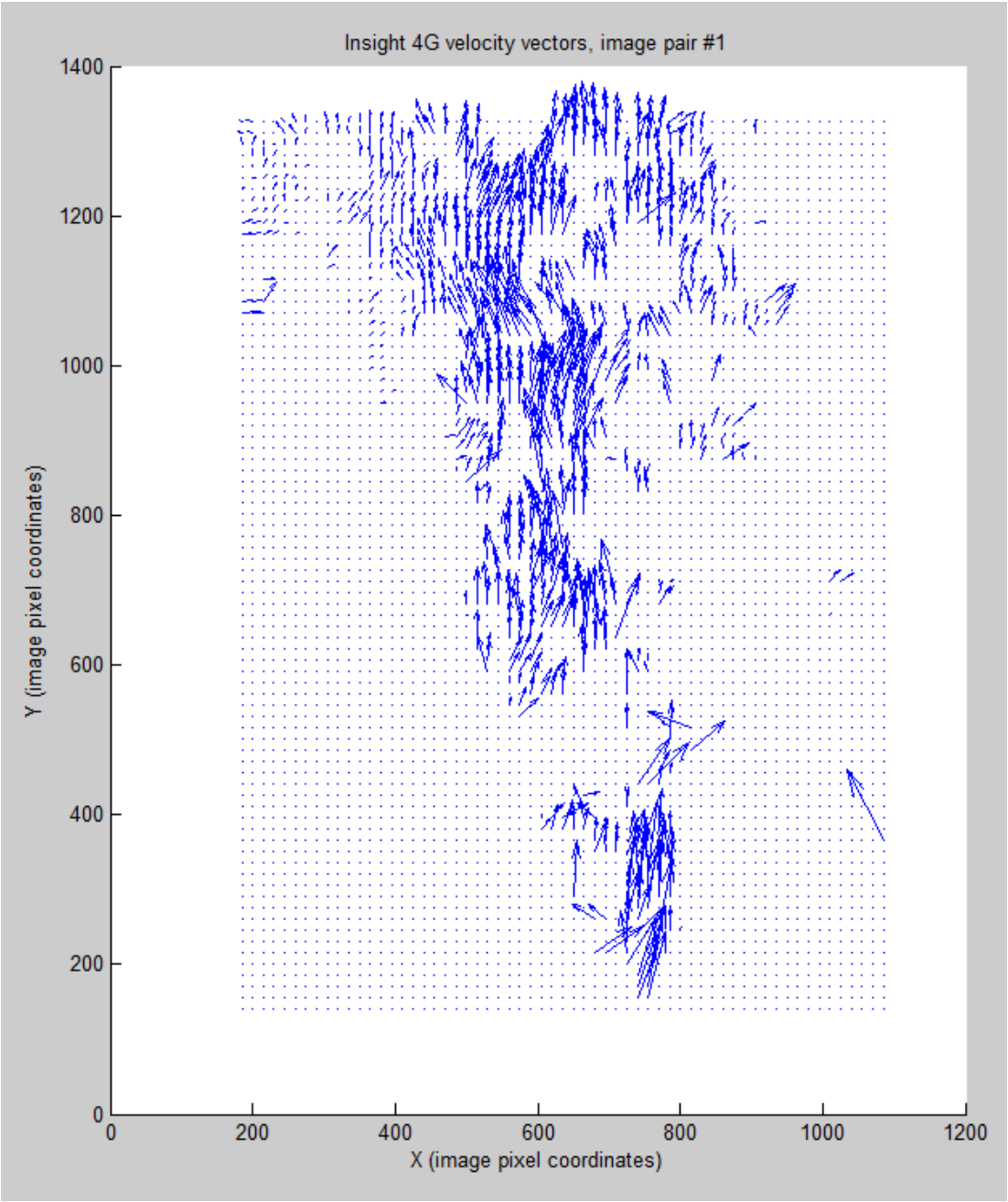


Figure B.5: Vertical flow analysis for Insight 4G, image pair # 51/200.

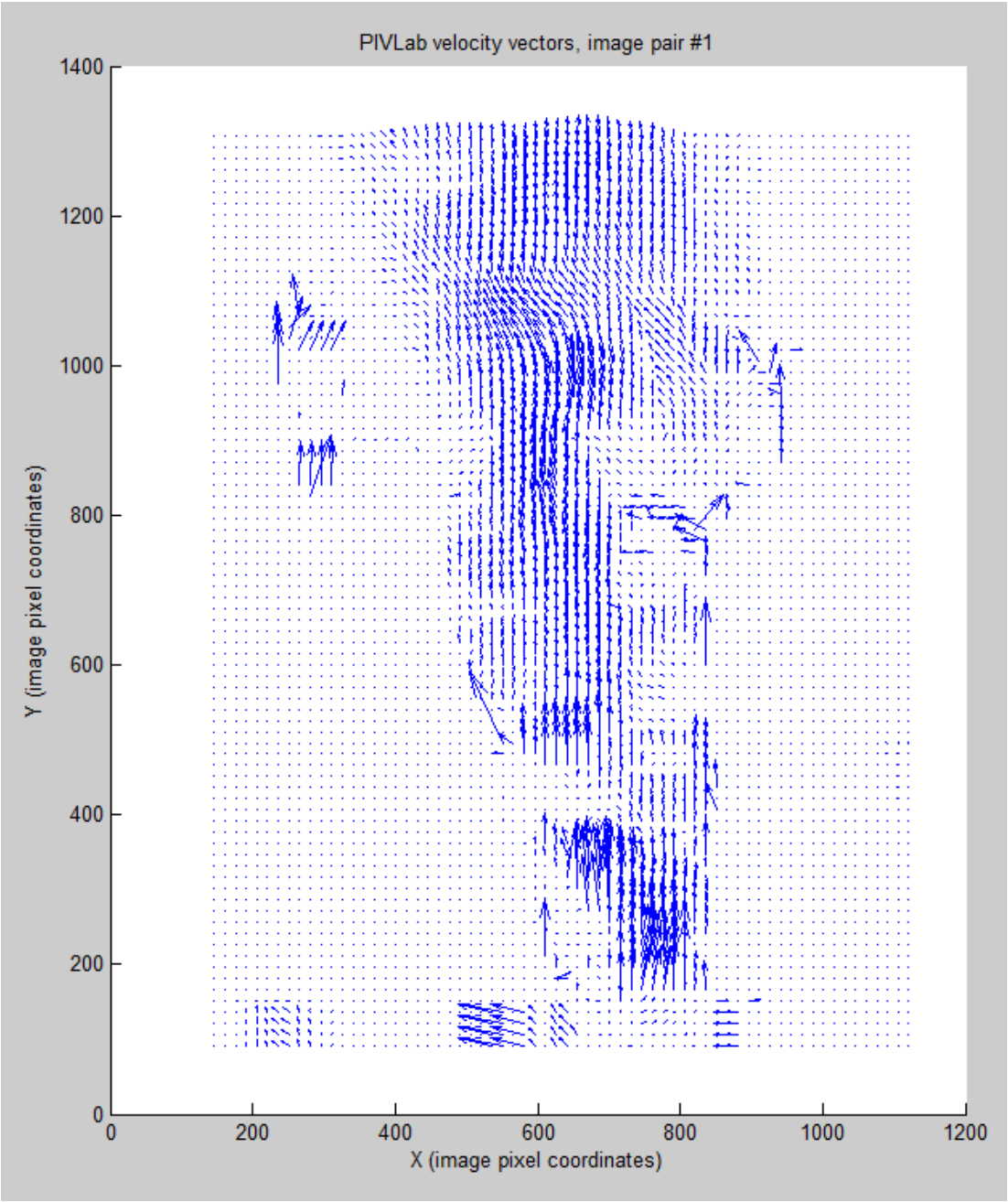


Figure B.6: Vertical flow analysis for PIVLab, image pair # 51/200.

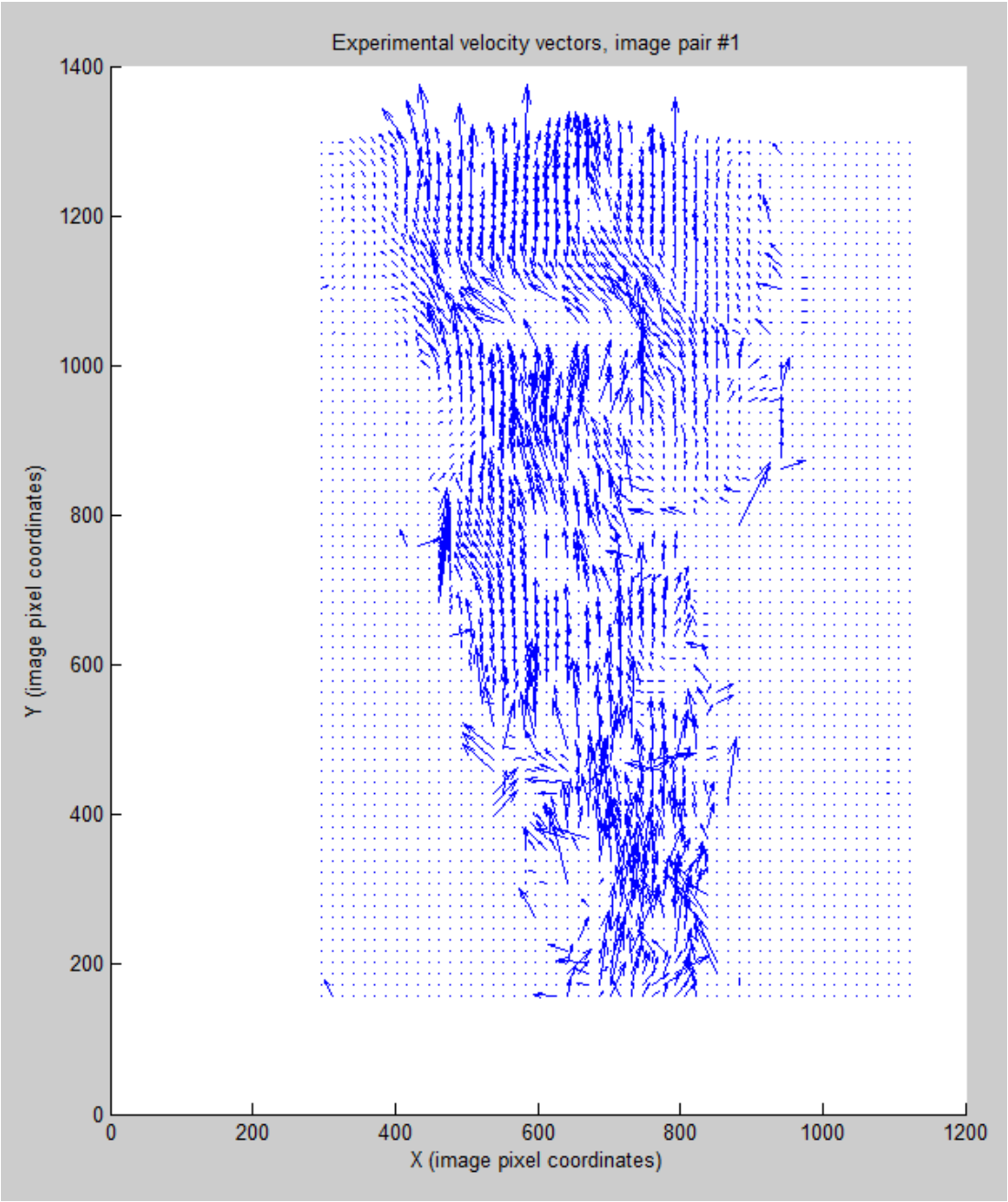


Figure B.7: Vertical flow analysis for this program, image pair # 101/200.

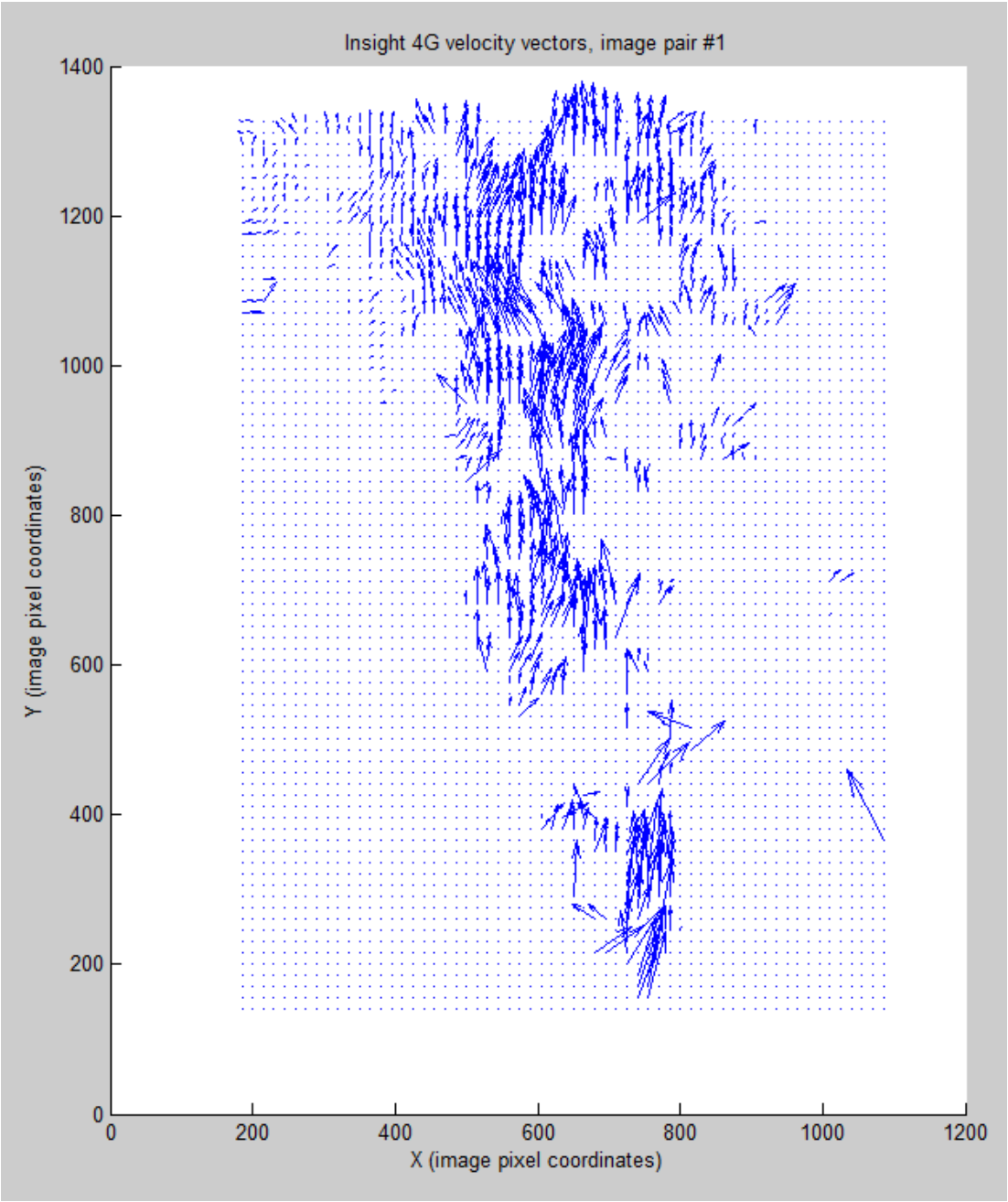


Figure B.8: Vertical flow analysis for Insight 4G, image pair # 101/200.

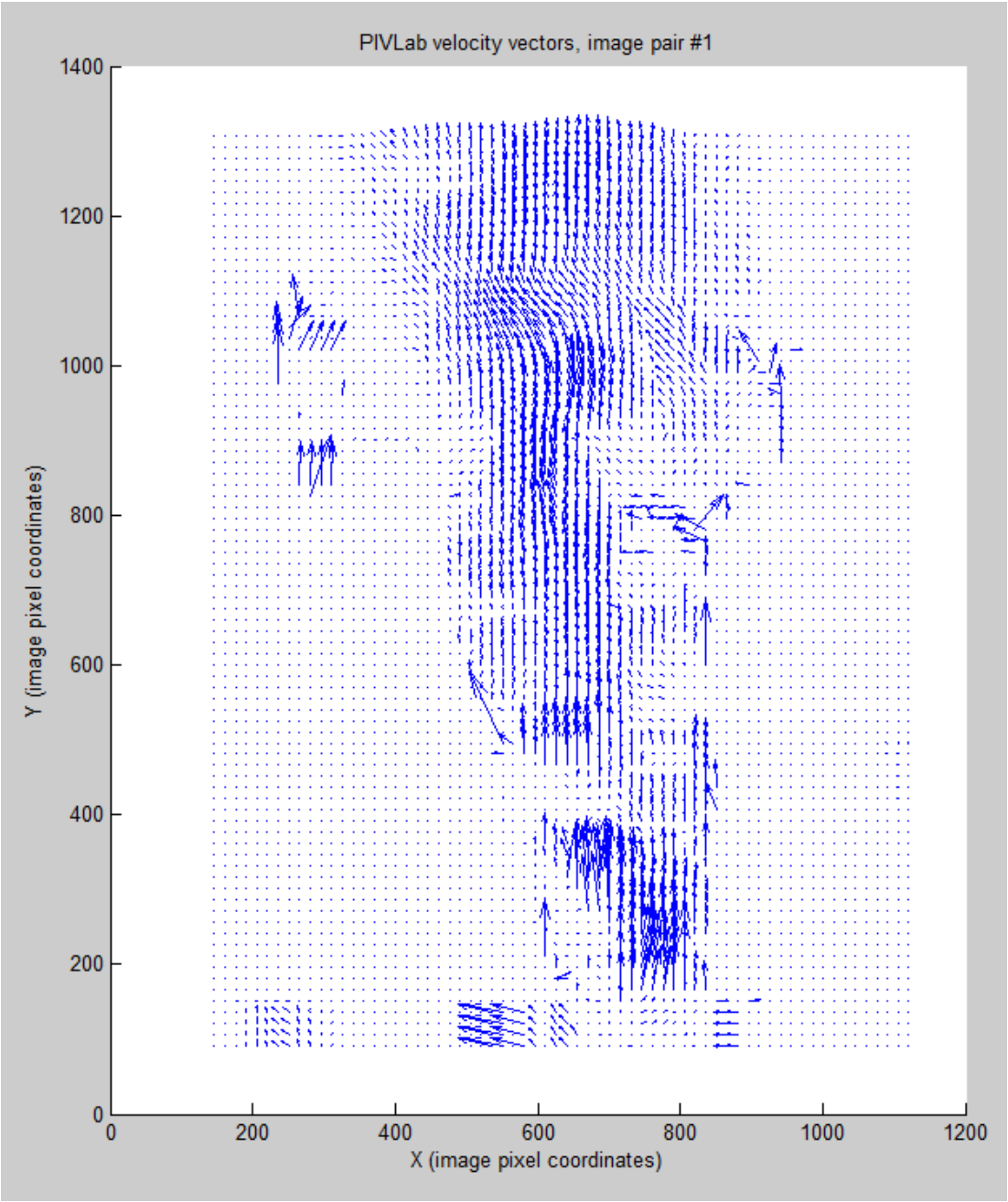


Figure B.9: Vertical flow analysis for PIVLab, image pair # 101/200.

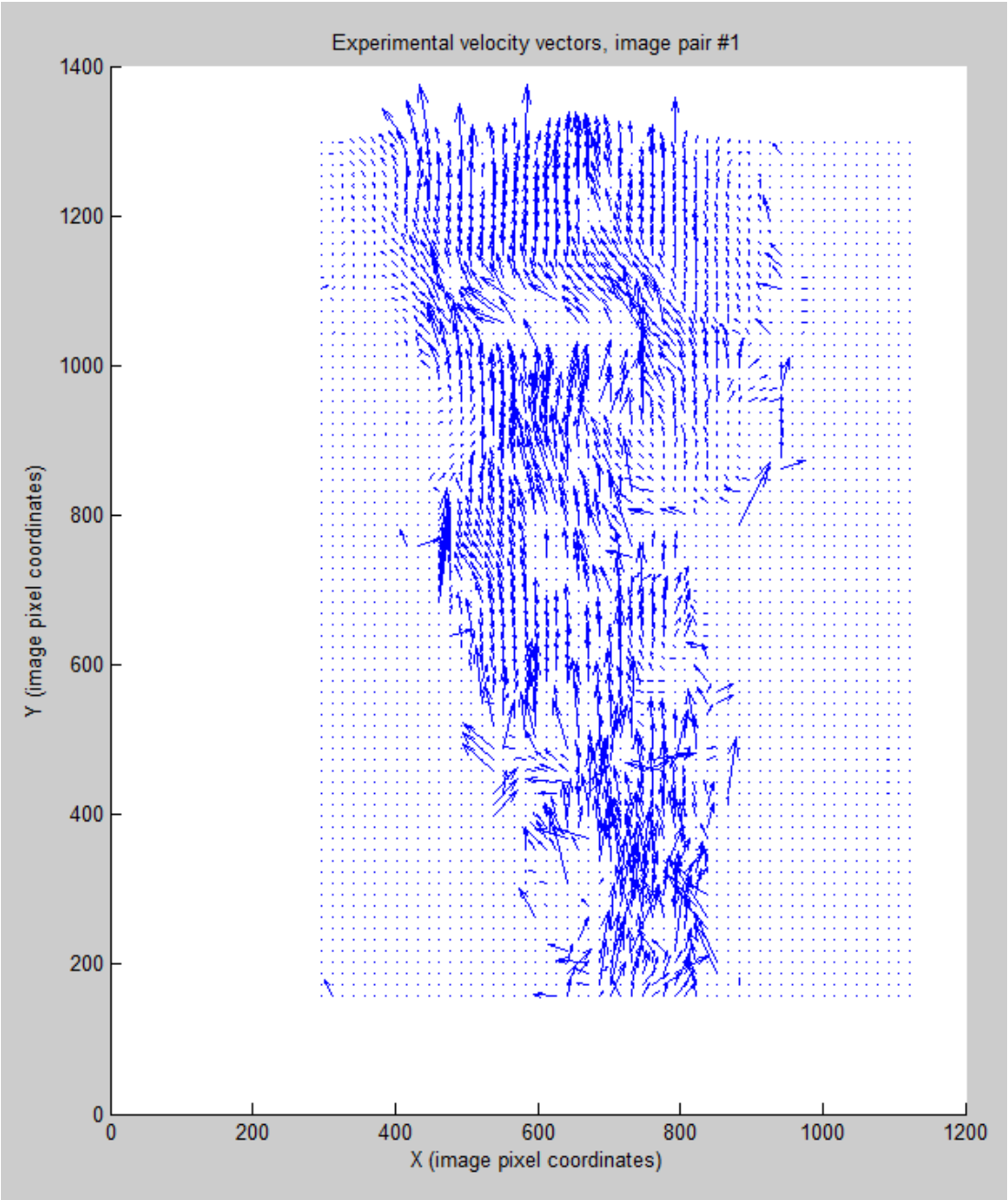


Figure B.10: Vertical flow analysis for this program, image pair # 151/200.

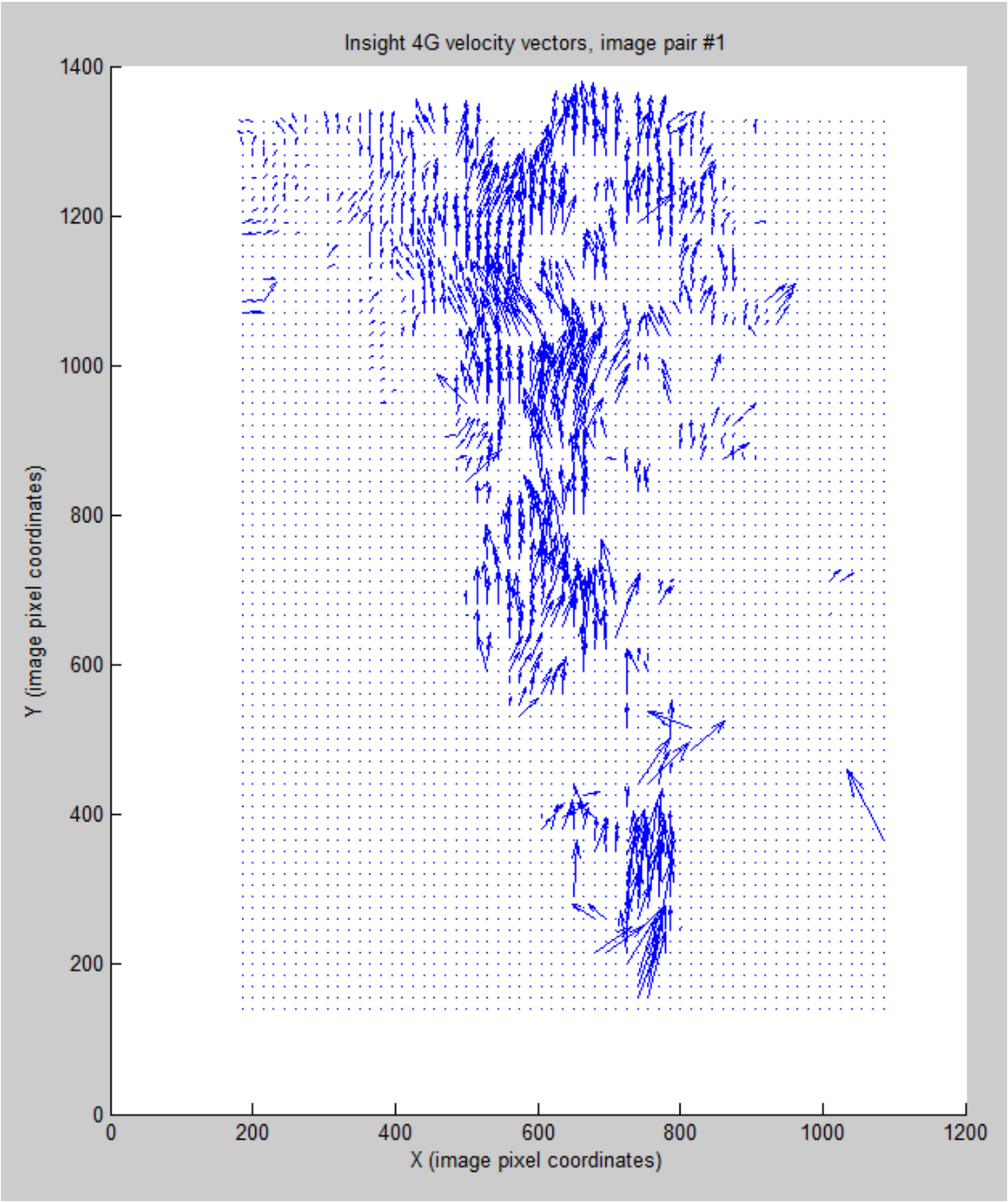


Figure B.11: Vertical flow analysis for Insight 4G, image pair # 151/200.

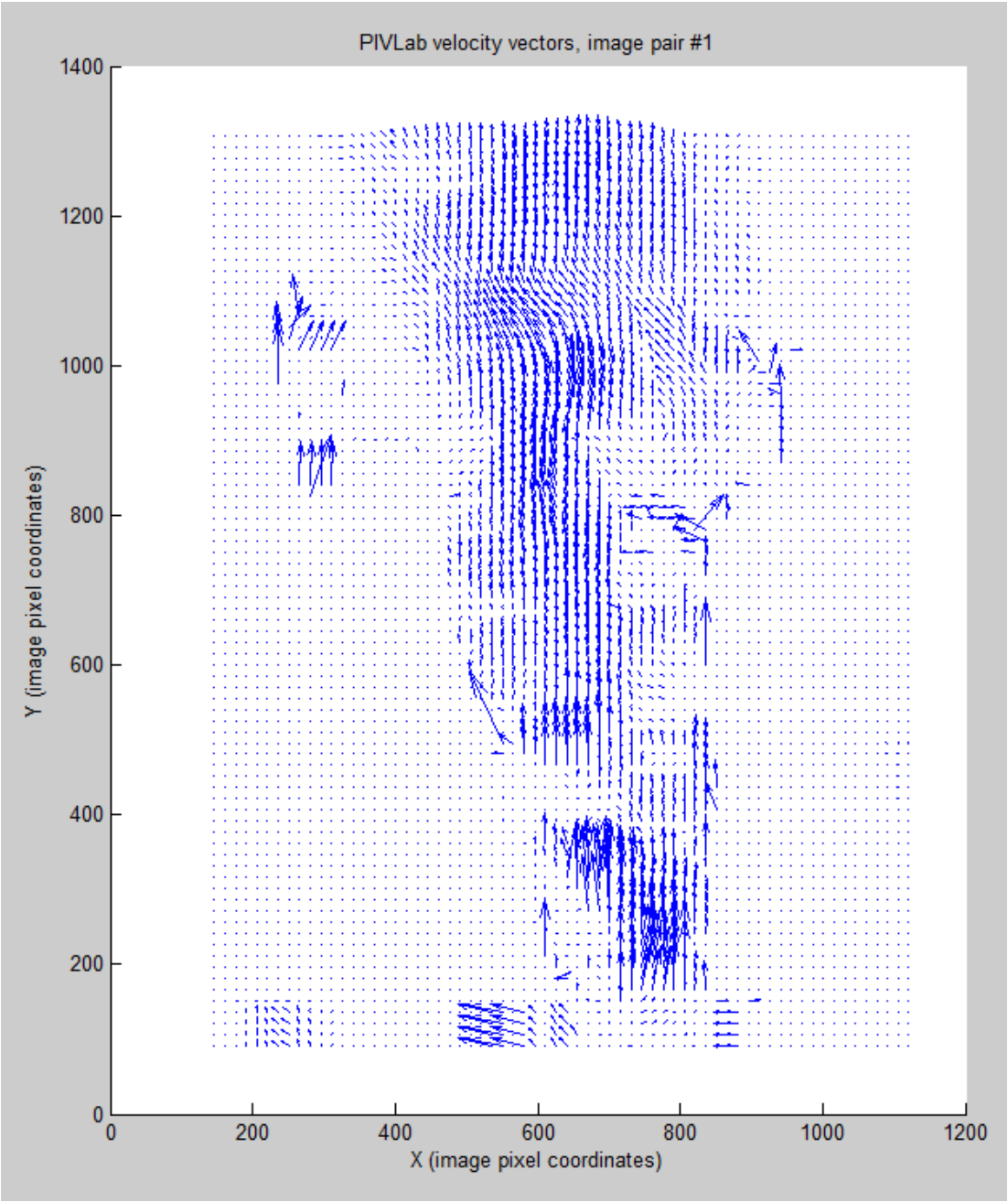
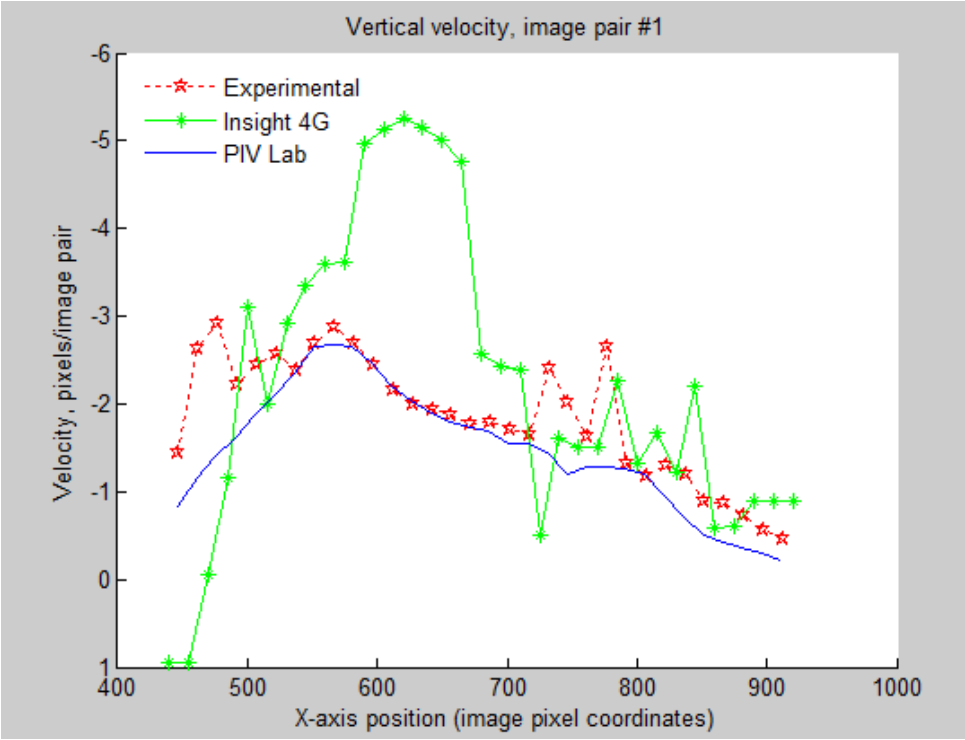
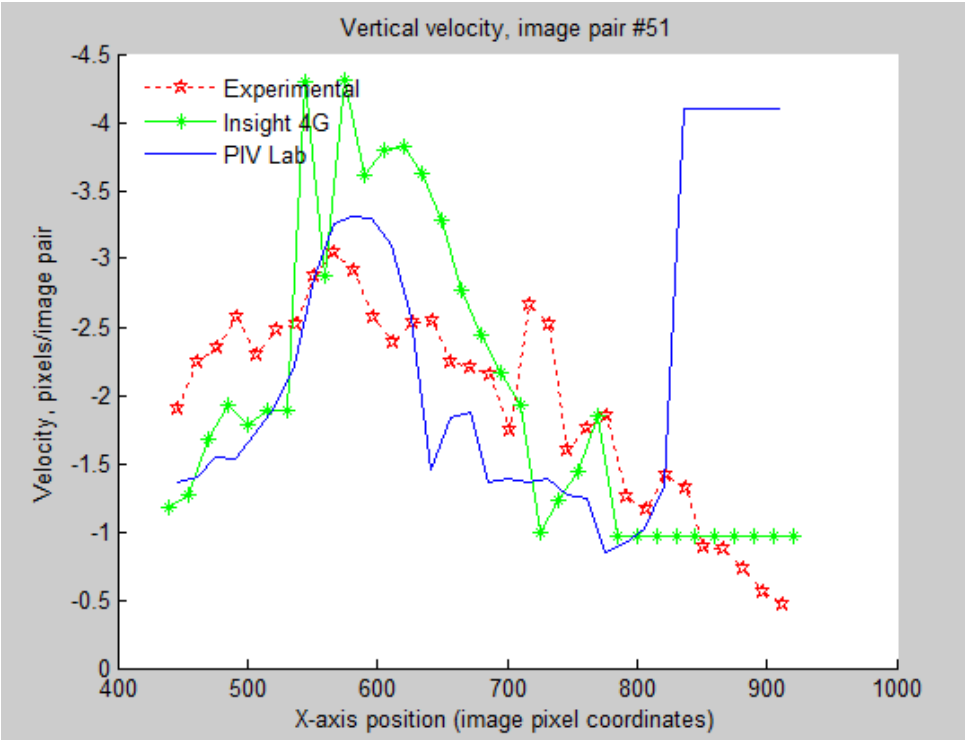


Figure B.12: Vertical flow analysis for PIVLab, image pair # 151/200.

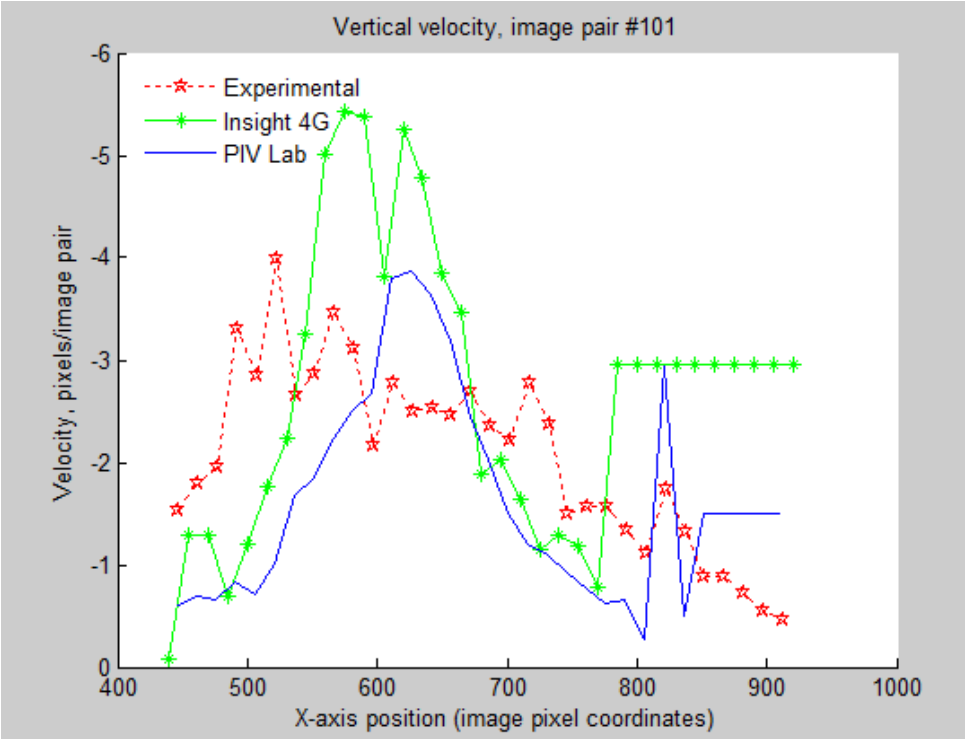


(a)

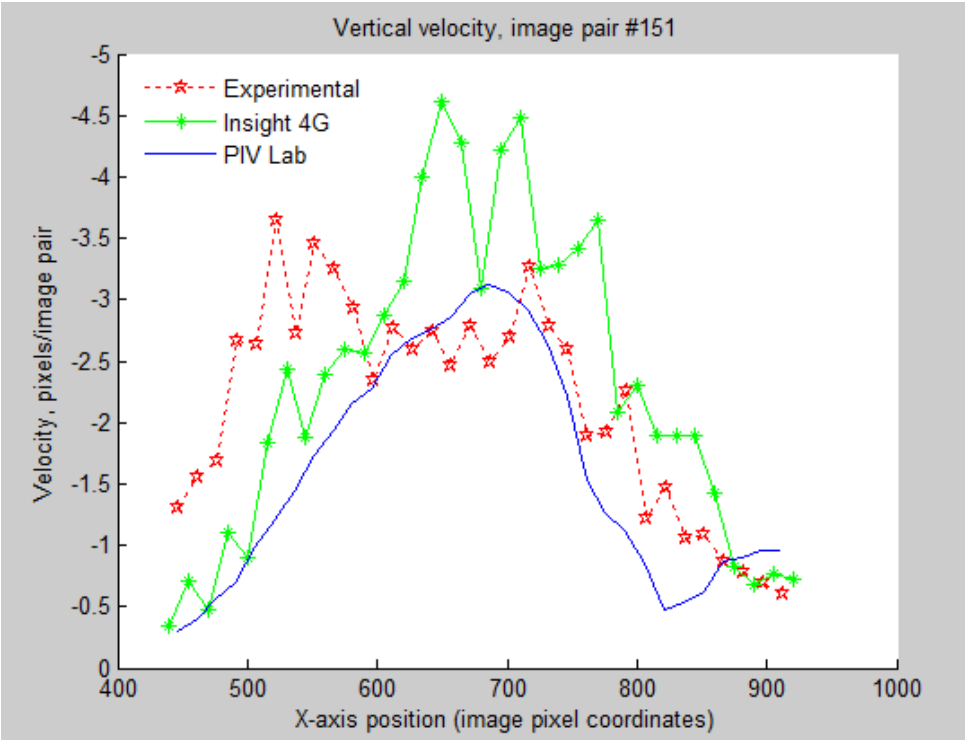


(b)

Figure B.13: Instantaneous cross-sectional flow at y = 1200, for (a) image pair # 1, and (b) image pair # 51.



(a)



(b)

Figure B.14: Instantaneous cross-sectional flow at y = 1200, for (a) image pair # 101, and (b) image pair # 151.

APPENDIX C

ADDITIONAL PIV IMAGE MOTION COMPARISONS

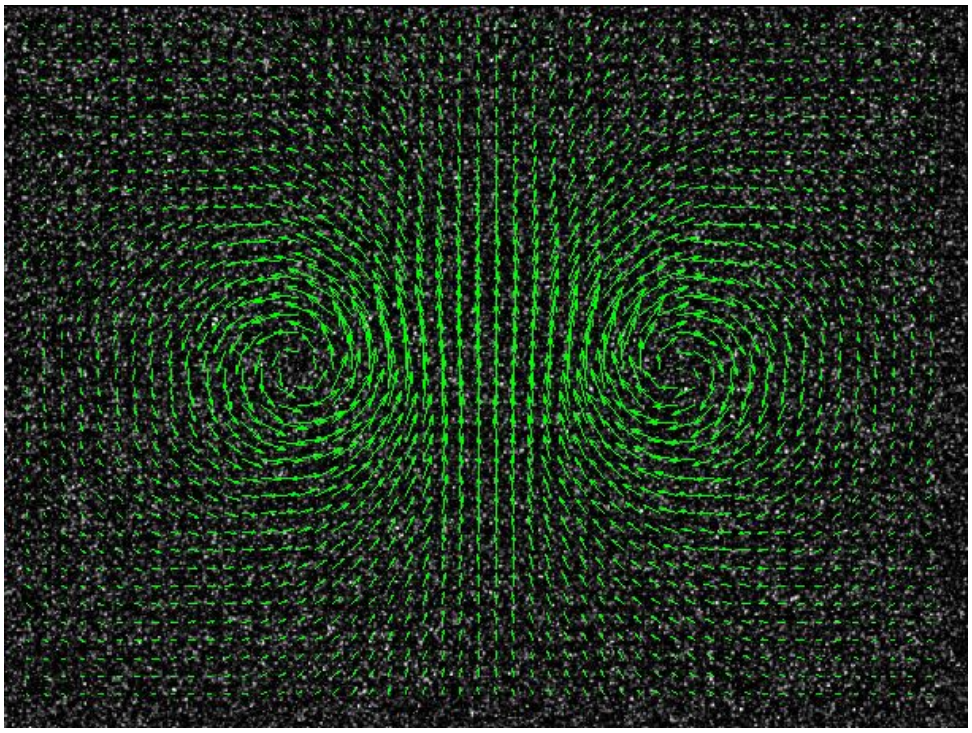
To provide additional confirmation of the success of the PIV search in this program, images from Thielicke and Stamhuis [2014b], are used as inputs.

C.1 Rankine vortex pattern

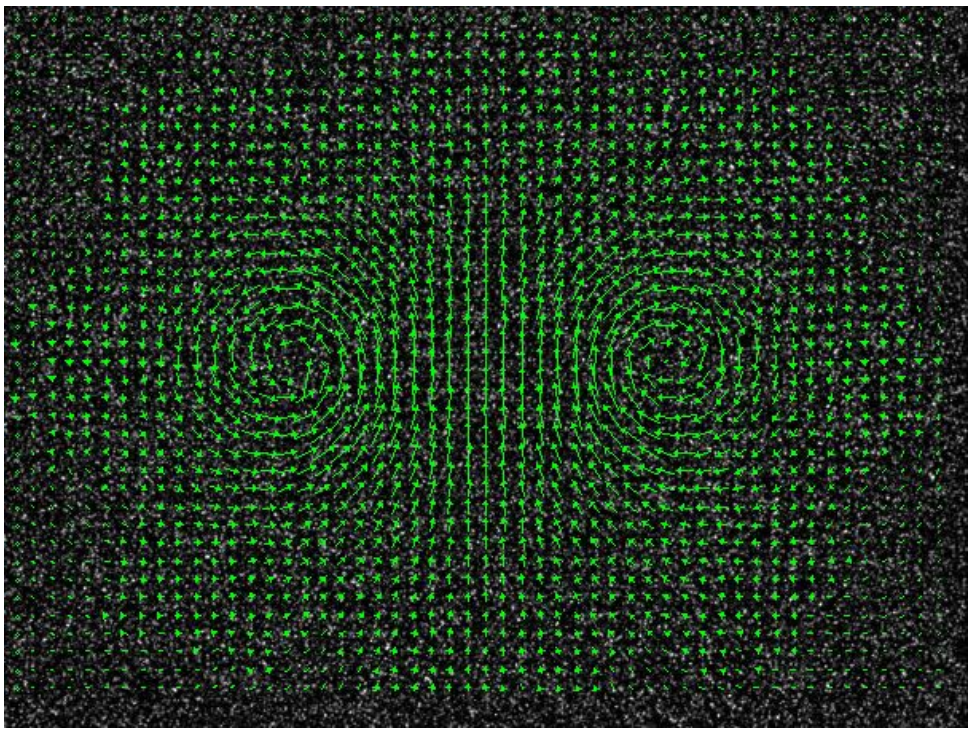
In Fig. C.1, a noisy background is digitally altered to produce a dual-foci vortex pair using a Rankine vortex simulator [Thielicke and Stamhuis, 2014a]. For the analysis, a phase correlation window of 64 is used, with a step size of 15. The output of this program compares well to the output from PIVLab, with the only notable differences around the edges, however this is only due to the graphical rendering of the vectors. The analysis shows that in tight regions with high rotational components, this program is still capable of resolving movement.

C.2 Karman vortex sheet

In Figures C.2 and C.3, two image pairs of a Karman vortex sheet are analyzed (using this program and PIVLab), with a correlation window of 64 and a step size of 15. The results are similar, with variations in vector rendering again the only noticeable differences. Vectors are very small (or missing) in the area directly behind the vortex sheet, likely due to the low-velocity flow pattern typical of this region in a Karman vortex plane.

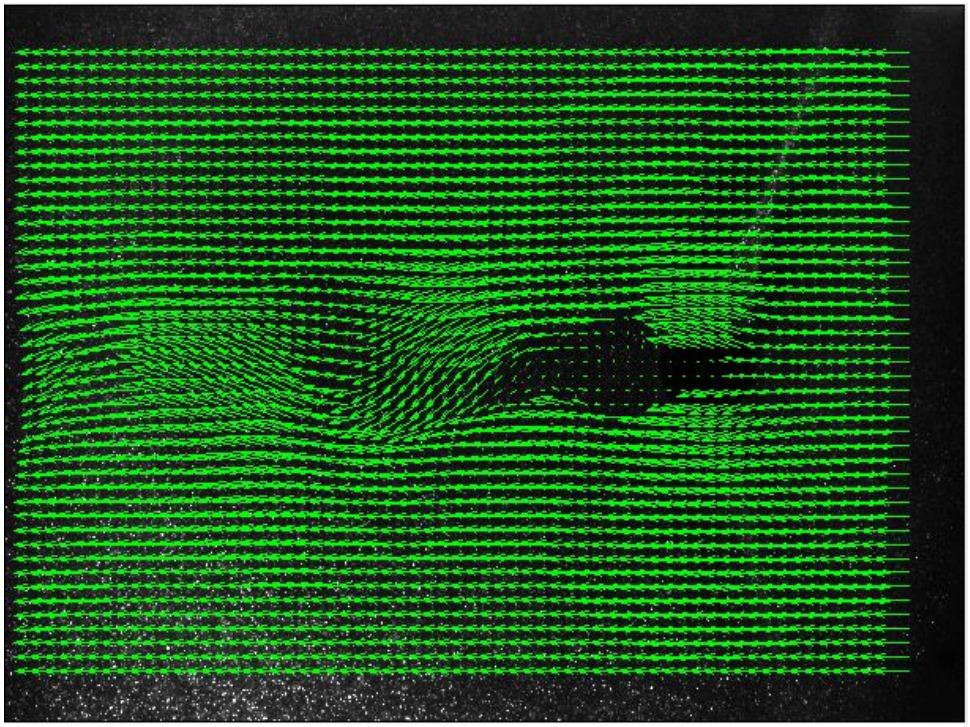


(a)

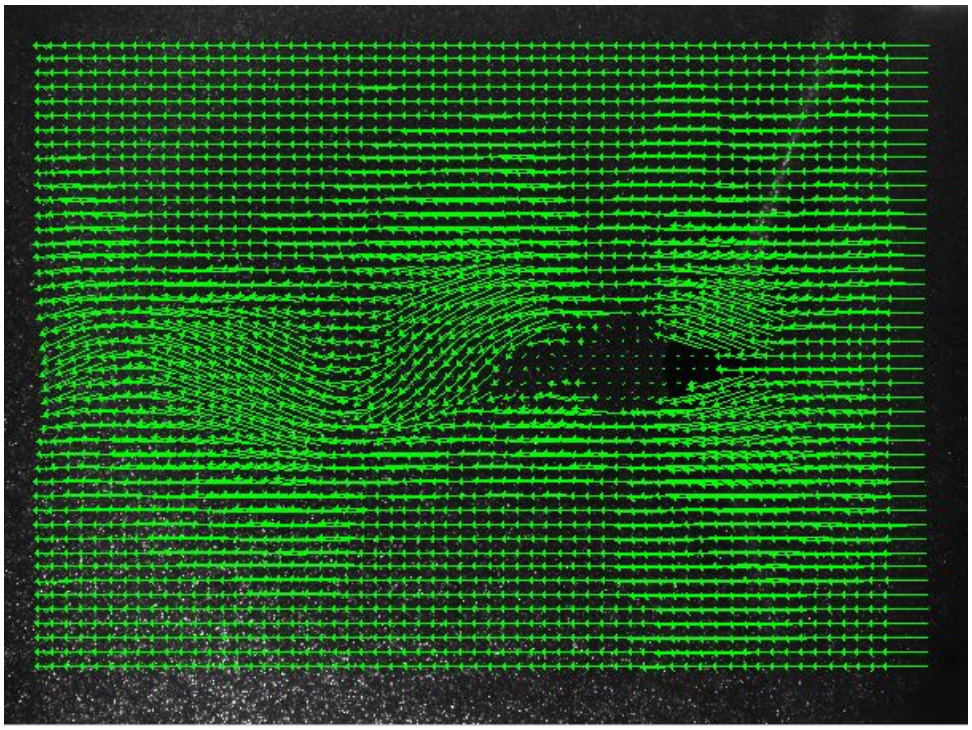


(b)

Figure C.1: Comparison analysis using PIVLab (a) and this program (b) using a digitally induced swirl pattern.

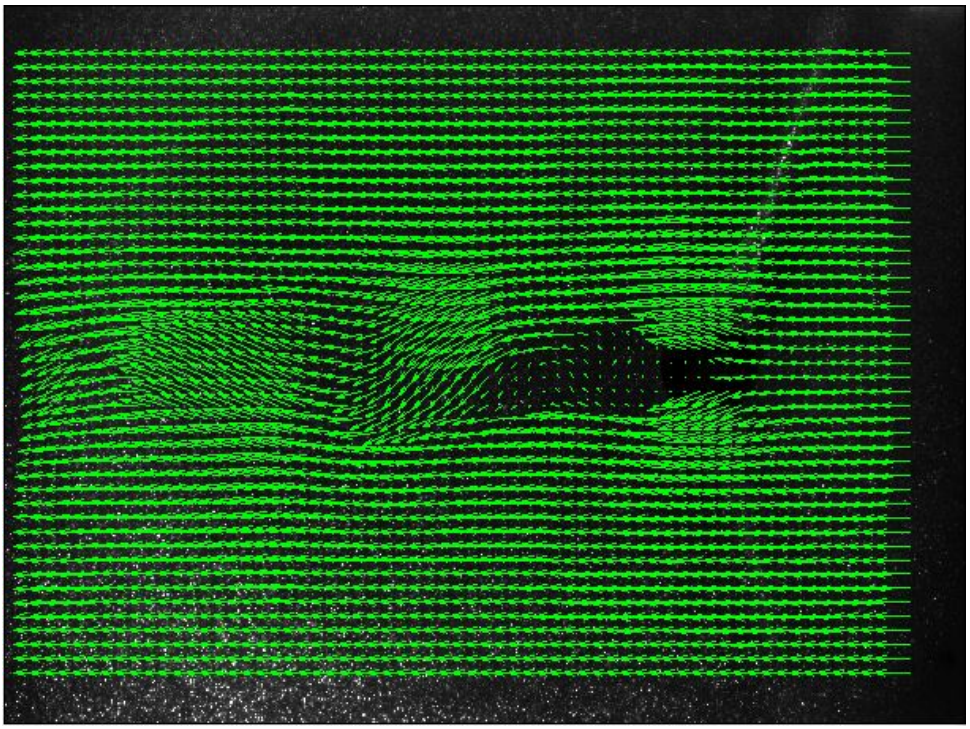


(a)

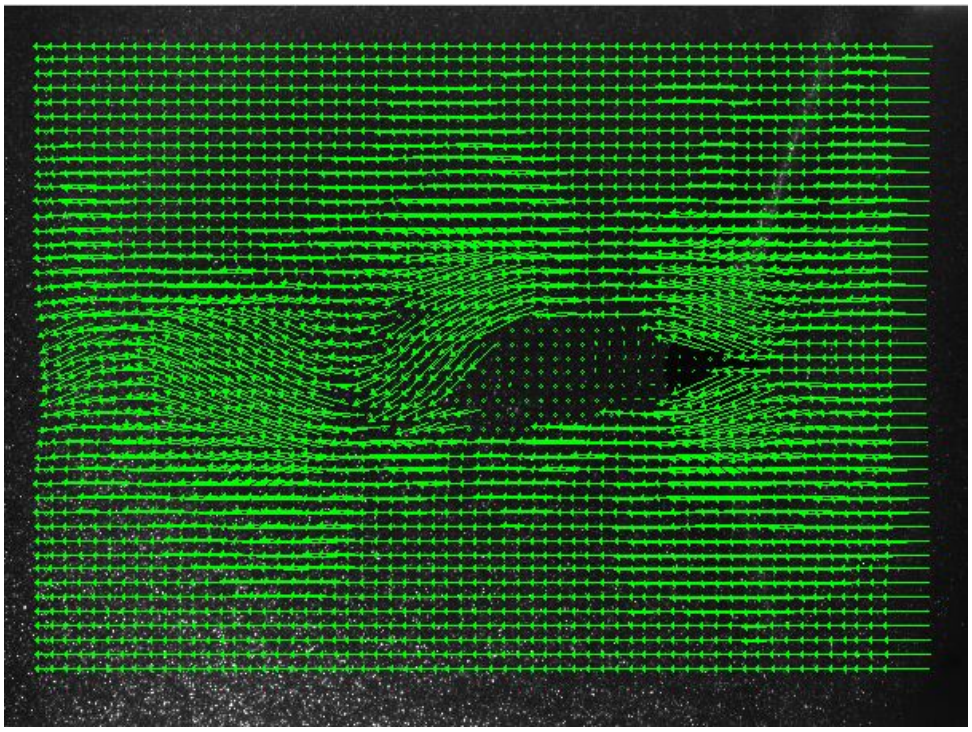


(b)

Figure C.2: Comparison analysis using PIVLab (a) and this program (b) on a Karman vortex sheet (first image pair).



(a)



(b)

Figure C.3: Comparison analysis using PIVLab (a) and this program (b) on a Karman vortex sheet (second image pair).