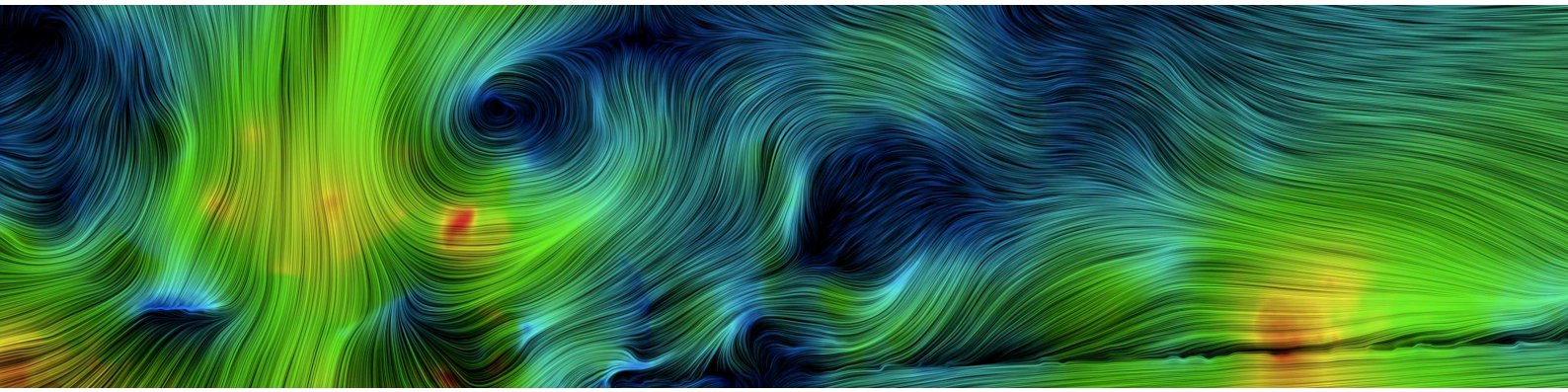
Optical Flow PIV

Improving the Accuracy and Applicability of Particle Image Velocimetry



Optical Flow PIV: Improving the Accuracy and Applicability of Particle Image Velocimetry

By: Matthew Stark

Submitted to the Eidgenössische Technische Hochschule Zürich (ETH) Department of Mechanical and Process Engineering (D-MAVT) on February 7, 2013, to complete the requirements for the degree of Master of Science in Mechanical Engineering.

Professor: Prof. Dr. Horst-Michael Prasser

Supervisor: Dr. Ralf Kapulla

Abstract

A technique to analyze Particle Image Velocimetry (PIV) images with strong refractive distortion is presented. Based on a muti-level optical flow algorithm, the approach has been benchmarked with synthetically generated PIV images, and has been successfully applied to both experimental PIV and Laser Induced Fluorescence (LIF) images. To improve velocity field estimation, new symmetric spline and advection warping methods are proposed together with a data-driven method to impose boundary conditions. To address a historical weakness of optical flow PIV methods, a framework is presented to select the smoothness parameter for experimental images. An analysis of an experimental image series with severe blurring and intensity distortions demonstrates the viability of the technique.

keywords: data-driven boundary condition, distributed smoothing, horn and schunck, optical flow, particle image velocimetry, smoothing parameter, symmetric advection, volume of fluid, warping

doi: 10.3929/ethz-a-009767070

Contents

Preface

This report, and the previous work on optical flow at the Paul Scherrer Institute (PSI), Switzerland, is a direct result of experiments performed at the GEneric MIXing (GEMIX) facility. Consisting of a 5 cm $\times$ 5 cm $\times$ 80 cm square Plexiglas channel, GEMIX is equipped with two inlets, one above the other, where two streams of water are injected. The flow rate and the fluid density of each inlet can be controlled independently, with the densities modified by adding solutes (i.e. sugar) to the water supply tanks. Measurements are performed via wire mesh sensors, wall capacitance sensors, laser induced fluorescence (LIF), and particle image velocimetry (PIV).

During GEMIX experiments where streams of different densities are mixed, refractive distortions cause the PIV tracer particles in the mixing region to appear blurry. The extent and severity of the blurring increases with flow rate and density difference to the point where laser light is refracted into the camera, causing large white blotches on the images. Although the flow structure is still captured in the images, the blurring of the point-like particles, combined with the spurious blots of light causes traditional PIV algorithms to perform poorly, or fail completely. In short, it is not possible to extract flow information from the region of most interest with conventional cross-correlation PIV techniques.

A promising solution to this problem presents itself in Horn and Schunck's optical flow algorithm, well known in the field of machine-vision, but only recently applied to fluid measurements. To characterize the algorithm and to determine its accuracy, an array of synthetic PIV images from an on-line database, and images generated from a large eddy simulation of the GEMIX facility, were analyzed using an optical flow PIV code developed at PSI. Although the velocities calculated with optical flow are comparable, and better resolved than traditional cross-correlation PIV, the increased resolution of the method exposes difficulties in resolving flow boundary layers. Investigating this phenomenon, with the hope that it would improve the accuracy of the optical flow PIV technique, provided the starting point for this thesis.

Introduction

Particle Image Velocimetry (PIV) is a widely used technique to measure the velocity field of a flowing fluid. Particles (small solids or liquid droplets) are entrained in, and are advected by the gas or liquid being investigated. A high intensity laser sheet illuminates a flow plane, and successive images are taken of the particle ensemble which follows the movement of the fluid. By calculating the distance travelled by each particle, and knowing the time between successive frames, the velocity field can be reconstructed.

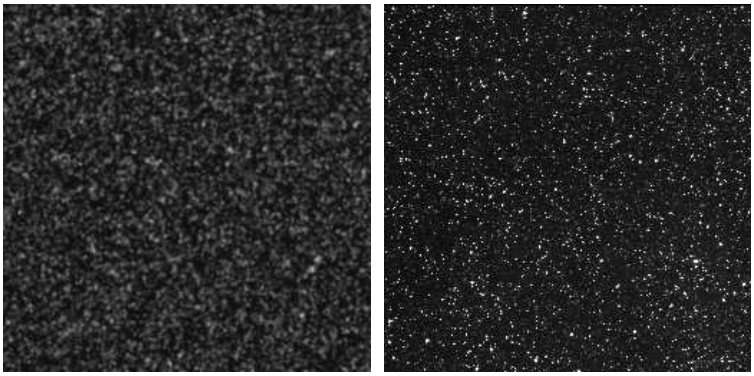


Figure 1.1: Example PIV images, synthetic image (left), and experimental image (right) (Carlier, 2005, Stanislas et al., 2003).

Originally measured by hand, the increase in computing power and digital imaging technology of the last decades has enabled computer algorithms to automate the extraction of particle displacements between image pairs (Westerweel, 1993). The most established of these algorithms cross-correlates square sub-divisions of the image (interrogation windows) between successive frames. Combined with high-order window deformation (Scarano, 2002) and multiple correlation passes, the technique is very robust in calculating complex flows.

Although very successful, cross-correlation PIV has some limitations. Flows with strong gradients or strong vorticity can cause incorrect correlation and degrade velocity estimation. Coupled with low velocity field resolution caused by the interrogation windows, incorrect correlations are particularly problematic when a boundary layer is only a few pixels thick, or when a velocity field must be interpolated to calculate a vorticity map. The low resolving power of correlation techniques is also a factor in turbulence analysis, which limits the size of the smallest measurable eddies.

For these reasons, researchers have been investigating other algorithms to supplement or replace traditional cross-correlation methods. One of the most promising alternatives is optical flow (Horn & Schunck, 1981), developed by the computer science community for machine-vision. The algorithm works on the premise that for two, 2D projections of a 3D scene (i.e. video frames) there exists a two-dimensional velocity field that moves the first image toward the second. The analogy is easily extended to PIV images, where 2D images are taken of a 3D flow field.

Since optical flow is predominantly used to analyze video sequences—mainly comprised of rigid body motion—most advances have been in resolving velocity field discontinuities and occlusions (two overlaid flow fields). For its success in overcoming both of these problems, the TV-L1 variant is considered superior to the original formulation of Horn and Schunck, and most literature focuses on this form. Unfortunately, neither flow discontinuities or occlusions are a major concern for traditional PIV measurements, aside from the possible case of flow discontinuities in shock waves. And contrary to the traditional case, a calculated velocity field that is smooth and continuous is generally preferred. Therefore, most optical flow techniques to analyze fluids use the original optical flow formulation of Horn and Schunck.

The earliest mention of optical flow being applied to fluid measurement can be traced back to Fitzpatrick (1988), where blood flow was measured from angiograph images. There was also some success in applying an optical flow-like algorithm to analyze LIF images (Su & Dahm, 1996a, 1996b). These early attempts paved the way for more recent application of the technique to schlieren (Atcheson et al., 2009), radiography (Wildes et al., 2000), atmospheric (Corpetti et al., 2002), (Doshi & Bors, 2007), and PIV images (Quénot et al., 1998; Ruhnau et al., 2005).

Since optical flow PIV is relatively young (the formal groundwork was only laid in 2005), no commercial codes are currently available to take advantage of the technique. Therefore, with the goal of analyzing the density driven mixing PIV images produced by the GEneric MIXing (GEMIX) experiment, an in-house code was developed at the Paul Scherrer Institute (PSI), Switzerland. Based on the work of Ruhnau et al. (2005), "OFN" is a highly modular Matlab code designed for rapid customization/testing cycles. Using OFN as the primary test bed, the work done for this thesis was aimed at benchmarking and improving the performance of optical flow PIV. While a small part of the performance improvements involved code speedups (see Appendix A), this thesis considers performance as *accuracy* in calculating the underlying flow field between two images. To objectively measure the accuracy of the calculated flow fields, computer-generated PIV images, created from known velocity fields have been extensively used.

Theory

This section presents brief summaries of the theoretical concepts relevant to this work. The experimental application of PIV is presented, along with cross-correlation analysis techniques. The focus is then shifted to optical flow, its historical development beginning with the theory laid by Horn and Schunck, up to its application to PIV image analysis. The chapter finishes with a review of common error metrics used to measure algorithm accuracy. Sections on the practical implementation of PIV and cross-correlation techniques have been summarized from the book "Particle Image Velocimetry: A Practical Guide" by Raffel et al. (2007), the authoritative work on traditional PIV methods.

PIV Imaging

A prototypical experimental PIV setup is shown in Figure 2.1.

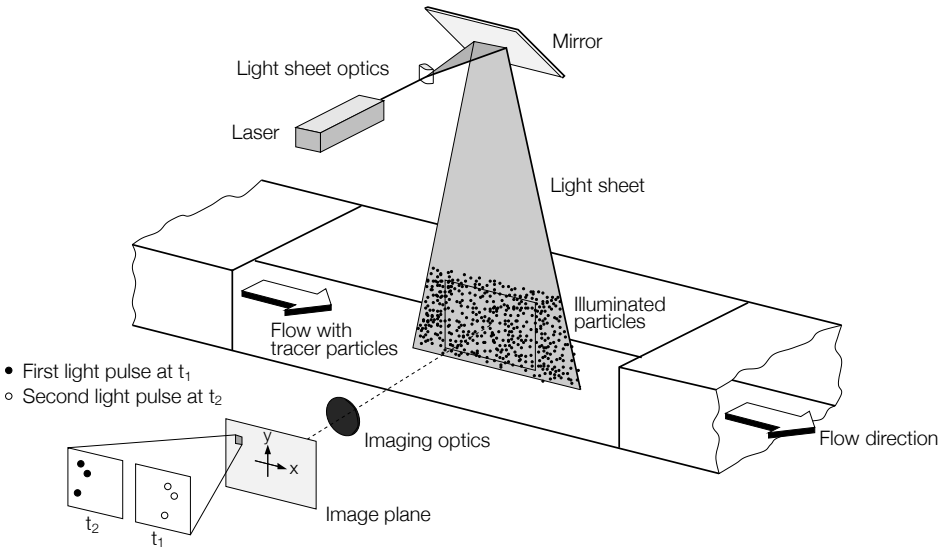


Figure 2.1: Typical experimental PIV setup, from Raffel et al. (2007).

The flow of interest, a transparent fluid such as air or water, is seeded with small tracer particles. A high power laser (typically a double-pulsed Nd:YAG) is positioned in the same plane as the flow to be investigated, and a hemi-spherical lens is used to spread the laser beam into a laser sheet to illuminate a plane in the flow. A pair of images are taken fractions of a second apart, so particles move only a couple of pixels between frames. Exposure times are on the order of nanoseconds, requiring intense

flow illumination by the laser, which is synchronized with the camera shutter. The image pairs, along with timing and spatial calibration data are saved to a hard disk during the experiment, to be analyzed offline.

Seeding particles are chosen for their ability to accurately follow the flow, and efficiently scatter laser light. To ensure particles *will* follow the flow, the relaxation time τ_s in Equation 2.1 must be sufficiently small (i.e. lower than the lowest characteristic timescale of the flow one wishes to resolve).

$$\tau_s = d_p^2 \frac{\rho_p}{18\mu} \tag{2.1}$$

Where d_p is the particle diameter, ρ_p the particle density, and μ the dynamic viscosity of the fluid.

Synthetic Images

Since the ground truth of experimental images is not known, and image parameters are impossible to independently control, it is common to generate "synthetic" image pairs from a known velocity field, calculated analytically or through computational fluid dynamics (CFD) modelling. Together with the underlying velocity fields, these images can be used to test the algorithms performance when varying parameters such as particle size, seeding density, and noise (Raffel et al., 2007).

To generate synthetic images, a particle field (with the desired particle quantity, distribution, and diameter) is initialized on a grid. Particle centroids are advected through the flow field using either a Runge-Kutta or finite difference scheme for a given time interval. The resulting particle fields can then be used as an input to PIV algorithms. Advanced image generators include models for the width and distribution of the laser light sheet to give realistic particle illumination. Additionally, optical paths and digital sensor characteristics may also be included to increase image realism (Figure 2.2). Of course, not all phenomena can be reconstructed; non-uniform laser sheets, spurious reflections, and dead camera pixels increase the noise of real PIV images. If one wishes to quantify the residual uncertainty cause by these effects, it must be done with experiments (Kähler et al., 2012).

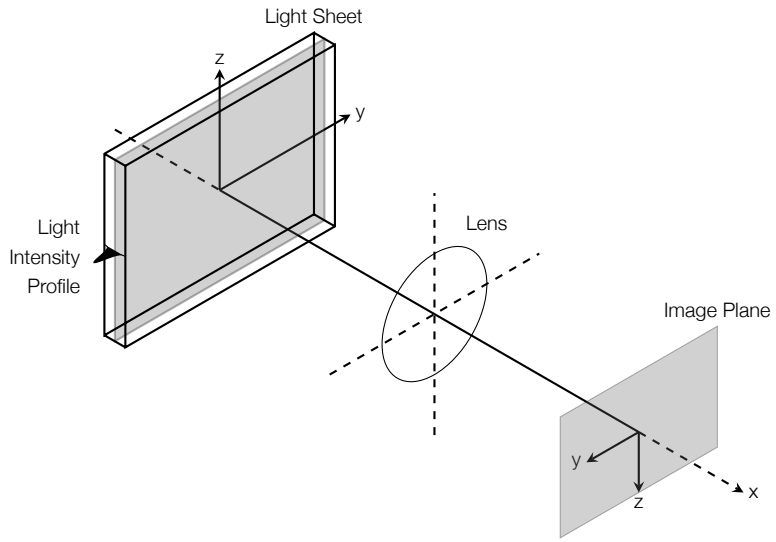


Figure 2.2: Synthetic PIV image generation model, adapted from Lecordier and Westerweel (2004).

Cross-correlation PIV

To effectively contrast it with optical flow PIV, the theory underlying traditional cross-correlation PIV is reviewed. The process begins by subdividing the first frame of the image pair into square grids, or "interrogation windows". The windows are correlated with an area four times their size in the second frame, as shown in Figure 2.3. The size of the interrogation window should be chosen so that at least half of the first frame window can be correlated with the area of the second frame. For example, if a particle moves 24 pixels between two successive images, the interrogation window size should be at least 48×48 pixels.

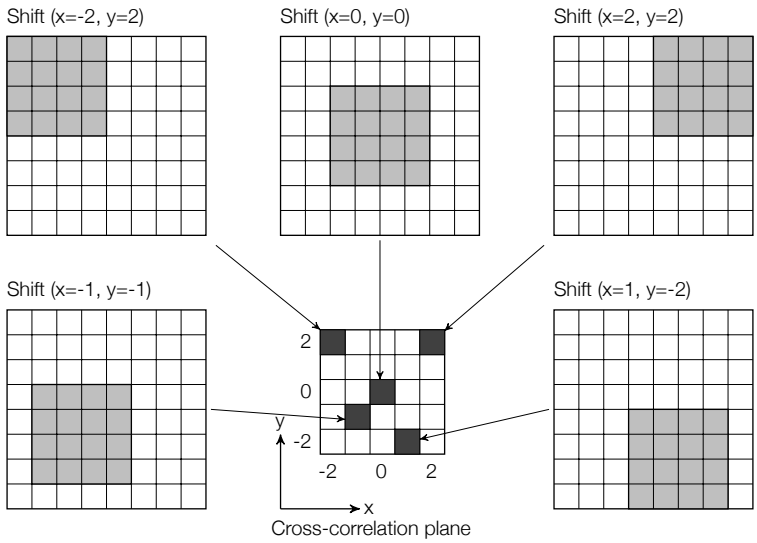


Figure 2.3: Cross-correlation map generation, from Raffel et al. (2007). The interrogation window from the first image (grey) is sequentially correlated over an area 4 times the size from the second image (in white). For each window offset, a correlation coefficient is calculated (dark grey).

To generate the correlation map, the interrogation window (from the first image) is shifted pixel by pixel over the search area (from the second image), and a correlation coefficient is calculated for each (x,y) shift according to the discrete formula:

$$R(x,y) = \sum_{i=-K}^{K} \sum_{j=-L}^{L} I_1(i,j) \cdot I_2(i+x, j+y) \tag{2.2}$$

Where I_1 and I_2 are the intensity values from the two frames, and K and L are half the size of the interrogation window. Variables x and y are indices indicating the window offset. Iterating x and y over the entire correlation domain, a map is generated as shown in Figure 2.4. Calculating the offset of the correlation peak from the center of the domain gives the average distance the particles in the interrogation window travelled between frames. In practise, correlations are calculated via fast Fourier transforms (FFT's) for speed, and the correlation peak is fitted with a gaussian function, to improve the sub-pixel accuracy of the displacement estimation.

Since its inception, cross-correlation PIV has seen improvements and extensions of this basic idea. For example, once the mean displacement for each interrogation window is known, increasingly

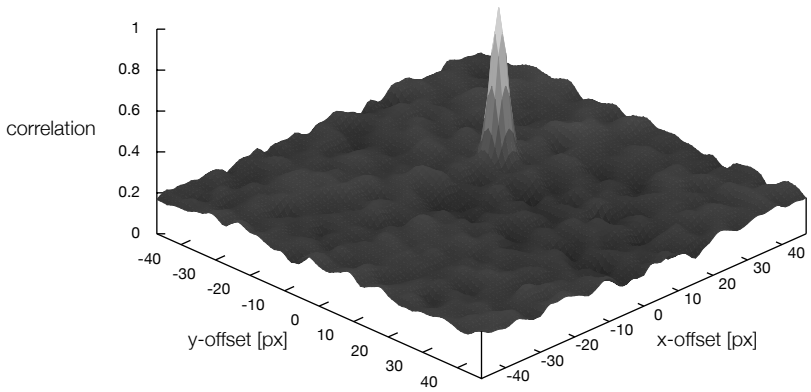


Figure 2.4: Example cross-correlation map. The distance between the peak and the center of the domain is the average distance the particles within an interrogation window have travelled between frames.

smaller interrogation windows can be used to increase the spatial resolution of the calculated velocity field. More sophisticated methods, such as those put forth by Scarano and Riethmuller (2000) and Scarano (2002) use previously calculated velocity fields to deform the interrogation windows to increase correlation strength in highly deformed regions.

Even with the aid of multiple passes and image de-warping techniques, it is still possible to have an incorrect correlation which causes spurious vectors, as seen in Figure 2.5. Thus, it is common in post-processing to remove outliers—velocity vectors that differ from neighbouring vectors as defined by a certain metric—to improve the quality of the vector field. To remove an outlier can mean exactly that, deleting it from the field, or it can be interpolated from neighbouring vectors. In a more sophisticated algorithm, a spurious peak in the correlation could be overpowering the correct displacement, and the algorithm will instead take the next lowest, correct peak.

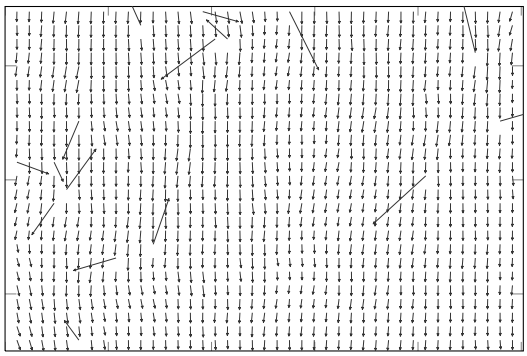


Figure 2.5: Example of a velocity field with outlying vectors.

A more natural way to correct for outlying velocity vectors would be to consider the qualities of the global velocity field. By considering average velocities, directionality, and rates of change, it would much more effective in mitigating spurious velocity vectors. Such a method, which is also unburdened by the resolution reducing effect of interrogation windows and the need for distinct point-like particle images can be found in the optical flow algorithm of the next section.

Horn and Schunck Optical Flow

Optical flow is the name given to the machine-vision algorithm put forth in a seminal paper by Horn and Schunck in 1981. It is a variational calculus approach to determine the velocity field linking two images via energy minimization. While the mathematical foundation presented below is rigorous, the underlying mechanism is intuitive; the algorithm calculates the velocity field which moves the first image towards the second with the smallest residual error. To aid in finding a solution, image data is considered along with a condition that neighbouring velocities are sufficiently similar. This condition helps to avoid incorrect velocity predictions in areas where noise or lack of data causes the problem to be poorly defined.

The basis of Horn and Schunck's algorithm is a conservation of brightness, I (defined at image coordinates x and y, for time t_1 and t_2) which ensures that the same feature in both images maintain their intensity.

$$\frac{DI\left(x,y,t\right)}{Dt} = 0 \tag{2.3}$$

When differentiated with the chain rule,

$$\frac{\partial I}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial I}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial I}{\partial t} = 0 \tag{2.4}$$

and finally, using shorthand notation,

$$I_x u + I_y v + I_t = 0 \tag{2.5}$$

The brightness constancy constraint equation (BCCE) is obtained. Variables u and v are the x and y velocity components, respectively, and the I terms are derivatives of image intensity in the indicated direction. At this point, the problem is ill-posed since there is only one equation for the two variables u and v. The implication of the ill-posedness are shown in Figure 2.6, where an infinite number of velocity estimates produce the same solution.

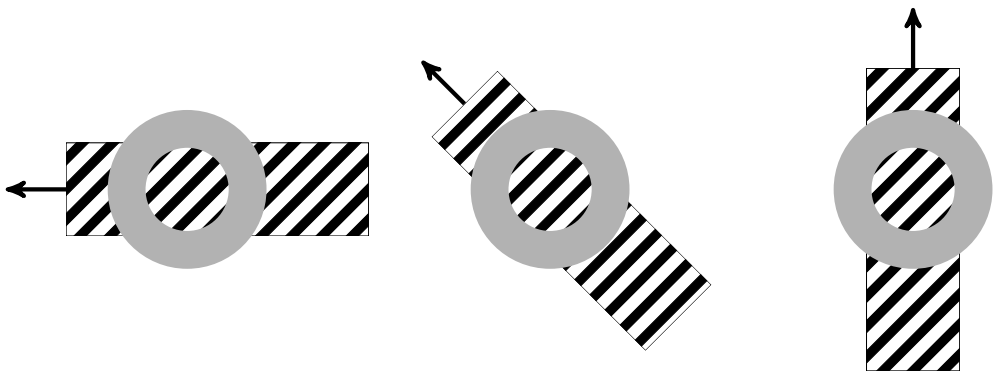


Figure 2.6: Schematic representation of the ill-posed aperture problem. For three *different* motions of the pattern, the apparent motion in the aperture of the grey circle is identical.

To overcome ill-posedness, a second condition is needed. In the case of Horn and Schunck, the L2 norm of the velocity gradients is used, otherwise known as the smoothness constraint, shown in Equation 2.6. Here, $|\cdot|$ is used to denote a vector norm.

$$|\nabla u|^2 + |\nabla v|^2 = 0 \tag{2.6}$$

With the aid of the second equation, it seems a simple matter to calculate the velocity field. However, it should be observed that neither the brightness, nor smoothness constraint equations are ever *strictly* satisfied due to quantization noise, parts of images being occluded, or changes in lighting. Therefore, the problem is reformulated as a minimization of errors, using the calculus of variations. The BCCE and smoothness constraints are equated to error terms, Equation 2.7, which are minimized in an expression over the entire image (the 2D domain, Ω) in Equation 2.8. The weighting of the two components are controlled by a scalar parameter α, which is treated in detail in Section 2.7. To be consistent with the literature, the BCCE is referred to as the *data term*, and the smoothness constraint as the *regularization term*.

$$\mathcal{E}_{dat} = I_x u + I_y v + I_t \tag{2.7a}$$

$$\mathcal{E}_{reg}^2 = |\nabla u|^2 + |\nabla v|^2 \tag{2.7b}$$

combining,

$$L = \int_\Omega \left[\mathcal{E}_{dat}^2 + \alpha^2 \mathcal{E}_{reg}^2\right] dx\, dy \tag{2.8}$$

and substituting,

$$L = \int_\Omega \left[(I_x u + I_y v + I_t)^2 + \alpha^2 \left(|\nabla u|^2 + |\nabla v|^2\right)\right] dx\, dy \tag{2.9}$$

Minimizing the function of Equation 2.9 with the calculus of variations (see Appendix B for a detailed derivation), the two coupled equations of Equation 2.10 are obtained.

$$I_x^2 u + I_x I_y v = \alpha^2 \Delta u - I_x I_t \tag{2.10a}$$

$$I_x I_y u + \quad I_y^2 v = \alpha^2 \Delta v - I_y I_t \tag{2.10b}$$

Numerically, the Laplacians of Equation 2.10 are calculated from a nine-point stencil according to Equation 2.11, and illustrated in Figure 2.7.

$$\Delta u = \kappa(\bar{u} - u) \tag{2.11}$$

where $\kappa = 3$, and:

$$\begin{aligned}\bar{u}_{i,j,k} = \frac{1}{6}&\left(u_{i-1,j,k} + u_{i+1,j,k} + u_{i,j-1,k} + u_{i,j+1,k}\right)\\ &+ \frac{1}{12}\left(u_{i+1,j+1,k} + u_{i-1,j+1,k} + u_{i-1,j-1,k} + u_{i+1,j-1,k}\right)\end{aligned}$$

1/12	1/6	1/12
1/6	-1	1/6
1/12	1/6	1/12

Figure 2.7: Nine-point Laplacian stencil (Horn & Schunck, 1981).

Substituting Equation 2.11 into Equation 2.10, we arrive at the final formulation of Horn and Schunck's algorithm. For the upper half of the system:

$$\begin{aligned}\left(I_x^2 + 3\alpha^2\right) u_{ij} - \frac{1}{2}\alpha^2 \left(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1}\right) \\ - \frac{1}{4}\alpha^2 \left(u_{i+1,j+1} + u_{i-1,j+1} + u_{i-1,j-1} + u_{i+1,j-1}\right) + I_x I_y v_{ij} \\ = \alpha^2 \Delta u_{i,j} - I_x I_t \quad (2.12)\end{aligned}$$

And for the lower half:

$$\begin{aligned}I_x I_y u_{ij} + \left(I_y^2 + 3\alpha^2\right) v_{ij} - \frac{1}{2}\alpha^2 \left(v_{i-1,j} + v_{i+1,j} + v_{i,j-1} + v_{i,j+1}\right) \\ - \frac{1}{4}\alpha^2 \left(v_{i+1,j+1} + v_{i-1,j+1} + v_{i-1,j-1} + v_{i+1,j-1}\right) \\ = \alpha^2 \Delta v_{i,j} - I_y I_t \quad (2.13)\end{aligned}$$

For an image of size $m \times n$ $(y \times x)$, Equations (2.12) and (2.13) reduce to a simple linear equation system (LES) of size $2mn \times 2mn$. The system is sparse, banded, and positive definite (Hackbusch, 1994), permitting the use of efficient solvers such as successive over-relaxation or krylov-subspace methods. An example of the equation system for a 4×5 image is shown in Figure 2.8, non-zero elements are indicated in black. Note that some diagonals are non-contiguous, especially in the first and third quadrants—a result of the Neumann boundary condition—derivatives at image edges are zero.

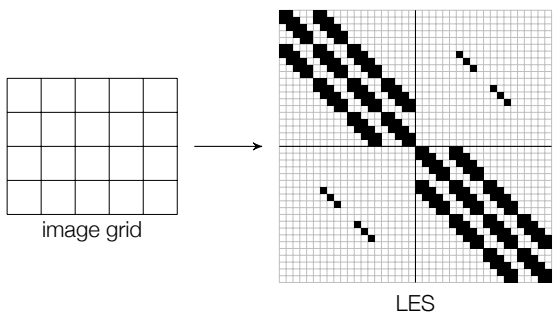


Figure 2.8: Optical flow LES of a 4×5 input image. Non-zero elements highlighted in black.

Optical Flow PIV

While the method of Horn and Schunck is very good at estimating motion, it is only effective if the motions are relatively small (Meinhardt-Llopis & Sánchez, 2012). Since it is very likely that PIV images *will* contain particles that move long distances, this problem must be addressed before optical flow can be of any practical use. One common solution is to construct a gaussian pyramid (Enkelmann, 1988) where an image is blurred with a gaussian kernel, and sub-sampled multiple times, as shown in Figure 2.9. Motion is calculated on the coarsest level, and projected to the next finest levels, which increases spatial resolution while reducing spurious velocity vectors (Quénot et al., 1998). To prevent aliasing between the levels of the pyramid, intermediate images (scale levels) are inserted and blurred with gaussian filters between the respective pyramid levels, as illustrated in Figure 2.10 (PL and SL are used as shorthand for pyramid level and scale level, respectively).

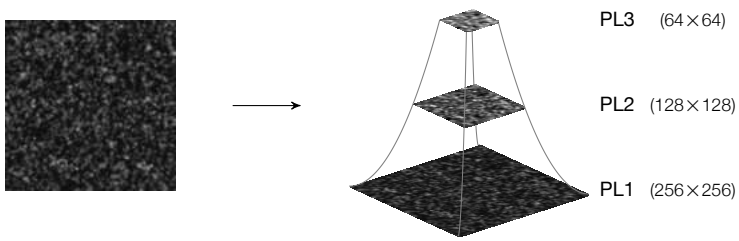


Figure 2.9: Representation of the gaussian pyramid, built from original image (pyramid base) through successive filtering and sub-sampling. Adapted from Ruhnau et al. (2005).

As detailed in Ruhnau et al. (2005), the algorithm starts on the coarsest (topmost) pyramid level, with the most blurred scale level (PL3, SL3 in Figure 2.9)[1]. The estimated flow field of the first calculation $(L+1)$ is passed to the next scale level (L) using the pyramidal (alternatively: multi-scale) Horn and Schunck method in Equation 2.15[2].

$$L = \int_{\Omega} \left[(I_x u_L + I_y v_L + I_t)^2 + \alpha^2 \left(|\nabla (u_L + u_{L+1})|^2 + |\nabla (v_L + v_{L+1})|^2 \right) \right] dx\, dy \quad (2.14)$$

and minimizing,

$$I_x^2 u_L + I_x I_y v_L = \alpha^2 \Delta u_L + \alpha^2 \Delta u_{L+1} - I_x I_t \quad (2.15a)$$

$$I_x I_y u_L + \quad I_y^2 v_L = \alpha^2 \Delta v_L + \alpha^2 \Delta v_{L+1} - I_y I_t \quad (2.15b)$$

[1] It is common to use a zero velocity initial guess for this step, so the first iteration of the pyramid is in effect the original Horn and Schunck algorithm.

[2] Note that a slightly different formulation of Equation 2.15 is used compared to Ruhnau et al. (2005), as OFN calculates the total velocity field at each level, not iterative correction fields that must be combined.

Successive projection and velocity field calculations continue down the scale levels until a jump in pyramid level requires the velocity estimate to be up-sampled. Calculations proceed down the scale levels as before, until the last pyramid and scale level is reached - with the original unfiltered images (PL1, SL1 in Figure 2.10).

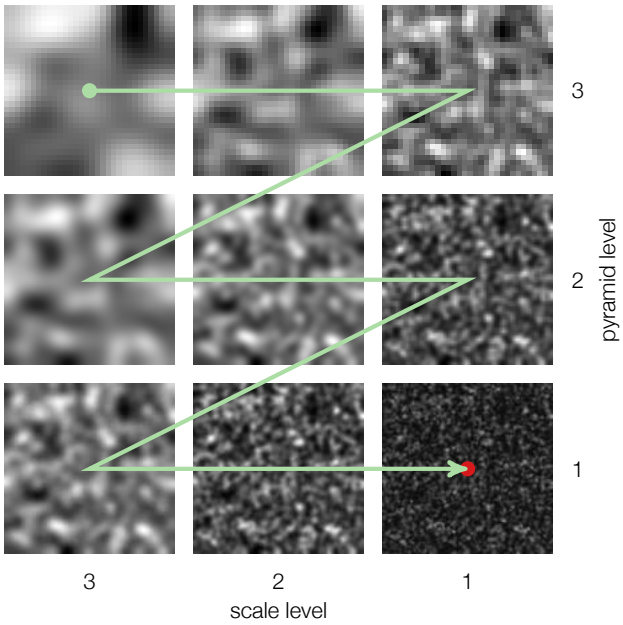


Figure 2.10: Representation of the gaussian pyramid levels (SL1) together with the other scale levels. Calculation begins on the coarsest most blurred level (PL3, SL3) and proceeds until the original input image is reached (PL1, SL1).

Warping

To propagate velocity estimates down the pyramid and scale levels, velocities from the previous level $(L+1)$ are used for the current (L) minimization. For the smoothness term, they are incorporated directly, but for the data term, they are applied in a pre-processing step known as *warping*.

In essence, the velocity field from the previous step is used to distort the image pair, with the current step using these distorted (warped) images to calculate the intensity derivatives. In this way, the estimations from previous levels are used to simplify the motion calculation at the current level, decreasing convergence time and improving accuracy. Common warping mechanisms, forward mapping and inverse mapping, are reviewed below.

Forward Mapping

As described in Wolberg (1994), forward mapping uses the velocity field to reassign the coordinates of each pixel of the original input image to a location in the warped output image, as shown in Figure 2.11. While simple, this method results in output pixels that have no information (holes) or output pixels that have contributions from two donor pixels (overlaps). To overcome these, it is possible to interpolate the irregular output grid using triangulation or distance weighting.

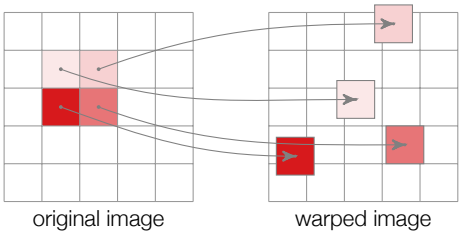


Figure 2.11: Illustration of the forward mapping procedure. The velocity field is used to reassign pixels from the original image to the warped output image.

Alternatively, each pixel can be considered as a deformable region, either a simple quadrilateral, or a higher-order Coon's patch (Coons, 1967) as shown in Figure 2.12. One can imagine a continuous array of deformed pixels all sharing curvilinear boundaries. The intensities of the output image are calculated from weighted contribution of the patches over each pixel.

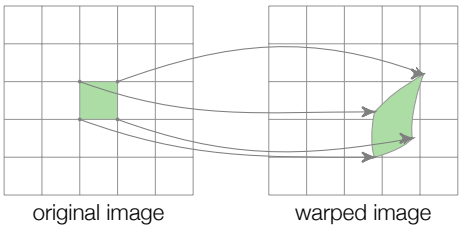


Figure 2.12: Illustration of warping a pixel as a Coon's patch. Each pixel of the warped image shares curvilinear boundaries with its neighbours.

Inverse Mapping

For inverse mapping, the reference frame is reversed. For each pixel of the warped output image, the location of its donor pixel is calculated *backwards* from the velocity components. The intensity is then "fetched" from its place in the original input image, shown in Figure 2.13. This method has the advantage that it produces no holes, and the fetched intensities are interpolated from a regular grid. It is a commonly used method since calculations are simpler and faster than with the forward mapping procedure.

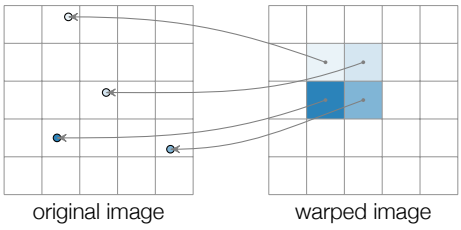


Figure 2.13: Illustration of the inverse mapping procedure. For each pixel of the warped image, the donor pixel is determined by calculated backwards to a location on the original image. The intensities for the warped image are then interpolated from their location on the original image.

Regularization

Let us return for a moment to Equation 2.6, the L2 smoothness term in the energy minimization function, which earns its name from its tendency to penalize velocities differing too much from neighbouring vectors. This "smoothness" also acts in regions where no gradient information is available, smoothly filling in information from neighbouring cells.

Using the constraint of Horn and Schunck, the resulting velocity fields are very smooth, even in the presence of motion boundaries. This is problematic for video analysis, since real-world scenes are full of rigid objects with defined edges moving in front of each other. For this reason, the computer vision community has adopted a modified approach, the so called TV-L1 optical flow, detailed in Sánchez et al. (2012) which is much better at preserving motion boundaries. Having been the focus of most optical flow research of the past two decades, the concept is thoroughly developed.

Of greater concern to the fluid community is the L2 regularizer's enforcement of non-rotational, divergence-free flow fields. To overcome this overly-strict condition, researchers have proposed re-formulations of the regularizer. Beginning with Suter (1994), the L2 smoothness term was decomposed into its divergence and curl components, so called div-curl regularization. Each of the components can be independently weighted to enforce a more divergence or rotationally-free flow field according to Equation 2.16[1].

$$\mathcal{E}_{reg}^2 = \alpha\,\mathrm{div}^2(u,v) + \beta\,\mathrm{curl}^2(u,v) \tag{2.16}$$

where,

$$\mathrm{div}(u,v) = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \qquad \mathrm{curl}(u,v) = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}$$

The div-curl regularizer concept was further developed by Suter (1994) and Gupta et al. (1996), where the second-order extension, Equation 2.17, was proposed. This method is far less restrictive than the first order regularization, penalizing high divergence/rotational *rates*—a much more physical formulation. Unfortunately, the method suffers from implementation issues and numerical workarounds must be used for the high-order terms (Corpetti et al., 2006).

$$\mathcal{E}_{reg}^2 = \alpha\,|\nabla\mathrm{div}(u,v)|^2 + \beta\,|\nabla\mathrm{curl}(u,v)|^2 \tag{2.17}$$

Other, more exotic regularizers have been proposed and also deserve mention. Beginning with Nagel and Enkelmann (1986), anisotropic regularization (imposing smoothness based on direction) was proposed (this idea has been used in the development of a spatially distributed α in this work). Weickert and Schnörr (2001a) built on this idea with their set of of isotropic/anisotropic, data/flow driven regularizers. Further developments of Weickert and Schnörr (2001b) also consider the temporal dimension of image frames with spatio-temporal regularization. While strictly-speaking not a regularization term, a new wavelet approach to solve the optical flow equation has been developed by Dérian et al. (2013) and shows promise for highly vortical flows.

In summary, there are a number of proposed methods to regularize the optical flow equation, some being more physically motivated than others. While recognizing that the original L2 smoothness term of Horn and Schunck might not be the most physically based regularizer, it is the one used exclusively throughout this work for two reasons:

- it is well tested, and its limitations are well known
- it has only one weighting parameter, α, the implications of which are considered next.

[1] If both α and β are equal, the expression is equivalent to the original L2 smoothness term, see Corpetti et al. (2006) for a detailed derivation.

Smoothness Weighting

Up to now, the role of the *scalar* weighting term of Equation 2.8 has not been emphasized, but in practise, this number strongly influences the optical flow algorithm. In an attempt to gain a clearer picture of the effect this term has on the final velocity field, we quote what other authors have said about this constant, α, which can also be found throughout the literature as: α^2, λ, "smoothness term", "smoothness weight", and "weighting factor".

From the original paper of Horn and Schunck (1981):

> *"What should be the relative weight of these two factors [$\mathcal{E}_{dat}$ and $\mathcal{E}_{reg}$]? In practise the image brightness measurements will be corrupted by quantization error and noise so that we cannot expect $\mathcal{E}_{dat}$ to be identically zero. This quantity will tend to have an error magnitude that is proportional to the noise in the measurement. This fact guides us in choosing a suitable weighting factor, denoted by α^2, as will be seen later..."*

and,

> *"...finally we can see that α^2 plays a significant role only for areas where the brightness gradient is small, preventing haphazard adjustments to the estimated flow velocity occasioned by noise in the estimated derivatives. This parameter should be roughly equal to the expected noise in the estimate of $I_x^2 + I_y^2$."*

From Ruhnau (2006):

> *"Parameter λ is either a user-parameter or can be determined as a Lagrange multiplier related to the either of the constraints:*
>
> $$\int_\Omega [I_x u + I_y v + I_t]^2 \; dx\, dy = \alpha$$
>
> $$\int_\Omega \left[|\nabla u|^2 + |\nabla v|^2 \right] \; dx\, dy = \beta$$
>
> *provided either of the numbers α or β is known. The discussion of this interpretation of the regularization parameter is, however, beyond the scope of this manuscript, and we regard λ as a user-parameter. A large value for λ leads to a very smooth flow field, whereas the smoothness decreases for smaller values for λ. At locations with $|\nabla I| \approx 0$ (i.e. untextured regions), no reliable flow can be estimated from the data term. At these locations, the smoothness term solves this problem by filling in information from the neighborhood, leading to a dense flow field."*

From Meinhardt-Llopis and Sánchez (2012):

> *"... α is a parameter to control the weight of the smoothness term compared to the optical flow constraint. α is squared so that its units are units of grey-level. This parameter can be seen as the gradient below which the method discards intensity variations considered as noise."*

From Atcheson et al. (2009):

> *"The Horn-Schunck and Brox algorithms are chiefly controlled by the smoothness parameter α, which damps high frequency spurious errors, as well as turbulent flow. Therefore, α should be made as small as possible to prevent over-smoothing, but beyond a certain (data-dependent) threshold, the effect of the regularizing smoothness term is diminished to the point that [the system] again become underdetermined."*

and,

> *"It is well-known that an appropriate choice of the smoothness weight is crucial for obtaining favourable results."*

From these excerpts, it can be gathered that the smoothness parameter is:

- important in determining the correct velocity field, and
- difficult to choose correctly.

For these reasons, much of the literature presents optical flow results with α selected heuristically to provide acceptable results. Only a handful of publications exist which propose an automated way to select this parameter, with varying levels of complexity and applicability as summarized below.

OPP (Zimmer et al., 2011) - the optimal α predicts the next frame in an image sequence most accurately. Most suitable for frames with constant velocity.

RISK (Ng & Solo, 1997) - the optimal α minimizes a proposed "risk" function. Suitable for image gradients with known, or no noise.

RMS (Tu et al., 2012) - optimal α minimizes the difference between warped frames. Universal.

BAYESIAN (Krajsek & Mester, 2007) - optimal α is selected via a proposed Bayesian framework. Universal.

As none of these methods has gained wide acceptance, and as it is difficult enough to select *one* scalar for the optical flow equation, the reader should be sympathetic with our choice to remain with the original Horn and Schunck formulation. The selection of α remains one of the open problems of optical flow research, and with the aid of physically constrained fluid flows, this thesis attempts to gain further insight into selecting this parameter.

Error Theory

To conclude the theory chapter, common error metrics used by the optical flow community to quantify algorithm performance are reviewed. Since comparing 2D vector fields is not entirely straightforward, the selection of error measure is strongly influenced by the particular underlying phenomenon being compared. The error measures defined below are a summary from the literature on optical flow, and are defined for each pixel of the input images. Pixel indices have been omitted for clarity.

Magnitude Error - L1 norm of the difference between the calculated and reference velocities, see Ruhnau et al. (2005).

$$\text{ME} = \left| \sqrt{u_{calc}^2 + v_{calc}^2} - \sqrt{u_{ref}^2 + v_{ref}^2} \right| \tag{2.18}$$

Average Magnitude Error - magnitude error weighted by the average velocity magnitude, see Kapulla et al. (2011).

$$\text{AME} = \frac{\left| \sqrt{u_{calc}^2 + v_{calc}^2} - \sqrt{u_{ref}^2 + v_{ref}^2} \right|}{\frac{1}{mn} \sum_i^m \sum_j^n \sqrt{u(i,j)_{ref}^2 + v(i,j)_{ref}^2}} \tag{2.19}$$

Angular Error - directional error metric, see Barron et al. (1994), and Baker et al. (2011).

$$\text{AE} = \cos^{-1} \left(\frac{1 + u_{calc} \times u_{ref} + v_{calc} \times v_{ref}}{\sqrt{1 + u_{calc}^2 \ + v_{calc}^2} \cdot \sqrt{1 + u_{ref}^2 + v_{ref}^2}} \right) \tag{2.20}$$

Endpoint Error - error metric combining the magnitude and angular errors. Euclidean distance between vector endpoints, see Baker et al. (2011).

$$\text{EE} = \sqrt{(u_{calc} - u_{ref})^2 + (v_{calc} - v_{ref})^2} \tag{2.21}$$

Average Endpoint Error - similar to the Average Magnitude Error, defined as the Endpoint Error weighted by the average velocity magnitude.

$$\text{AEE} = \frac{\sqrt{(u_{calc} - u_{ref})^2 + (v_{calc} - v_{ref})^2}}{\frac{1}{mn} \sum_i^m \sum_j^n \sqrt{u(i,j)_{ref}^2 + v(i,j)_{ref}^2}} \tag{2.22}$$

Warp RMS - a metric quantifying how similar two images are, see Baker et al. (2011) and Tu et al. (2012).

$$\text{Warp RMS} = \sqrt{\frac{\sum_{i=1}^m \sum_{j=1}^n \left(I_{w1}(i,j) - I_{w2}(i,j) \right)^2}{mn}} \tag{2.23}$$

Proposed Improvements

To complement the theory chapter, modifications to the optical flow PIV algorithm of Ruhnau et al. (2005) performed before and during this thesis are presented. The quantified effect of these modifications on algorithm accuracy is presented in the subsequent Results and Discussion chapter.

Previous Work

Program modifications which preceded, and became a foundation for the work of this thesis are summarized. While symmetric warping is used for all analysis in this report, intensity normalization is only applied to experimental PIV images, since the step is unnecessary for synthetic images.

Symmetric Warping

Typically when warping images in the gaussian pyramid, the first image is warped towards the second. But, since the velocity between frames is assumed steady-state, the second image could just as easily be warped toward the first. The only practical difference being that the calculated flow field is slightly shifted towards the static image. This slight shift however, introduces errors when analyzing synthetic image pairs, since the velocity field is usually defined at $t = 0$, and images generated at +/- $dt/2$. To exploit this time symmetry, and reduce errors when analyzing synthetic images, the idea of Scarano (2002) is used to implement a scheme where both input images are warped *toward* each other, shown in Figure 3.1.

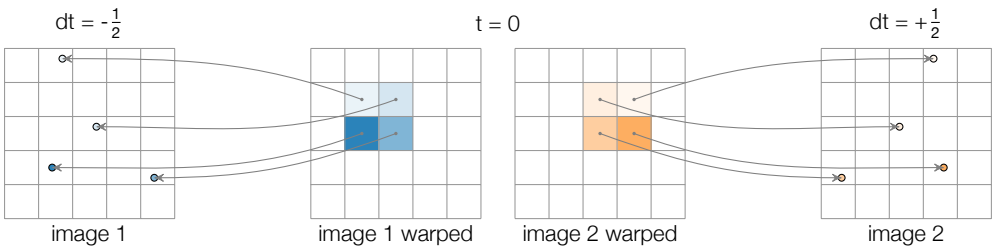


Figure 3.1: Illustration of the symmetric warping technique. Image 1 and Image 2 are warped towards each other using inverse mapping.

This method, referred to as symmetric warping, greatly improves the accuracy when analyzing synthetic images, since it mimics the image generation process. However, warping synthetic images is an Eulerian approximation of a Lagrangian particle advection. Thus, any errors in the warping scheme are compounded when used in a symmetric warp.

Image Intensity Normalization

Due to the gaussian profile of the PIV light sheet and optical aberrations, it is common to have a non-uniformly illuminated imaging plane. Combined with laser light reflections within the test section,

and in the case of GEMIX, high refractive distortion, there is a large amount of intensity variation in a given image. Considering the foundation of optical flow is that a point retains the same brightness in both images (Equation 2.3), it is important to correct these intensity variations. The optical flow community has had success with advecting image gradients (Brox et al., 2004), which are more invariant than raw image intensities. However, when applied to PIV images, this method enlarges the apparent size of particles and distorts the estimated flow field. Instead, the strict sliding maximum filter detailed in Kapulla et al. (2011) is used, which is successful not only in normalizing intensities, but removes much of the characteristic "blotches" of high refractive distortion. An example experimental image before and after filtering is shown in Figure 3.2, with the removal of light variations and increase in particle contrast clearly shown.

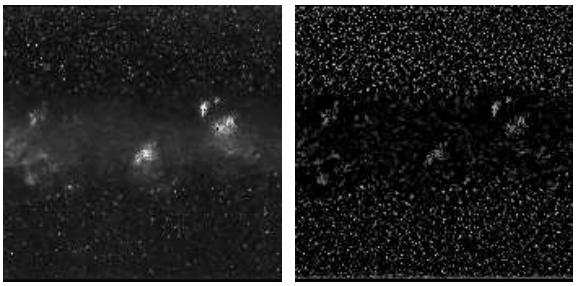


Figure 3.2: Experimental GEMIX image before and after filtering.

Prototypical Algorithm

To give context to the program modifications made in the next sections, a prototypical pyramidal optical flow program structure is shown in Figure 3.3.

```
> load image pair
> normalize image intensities[1,2]
> create image pyramid

for each pyramid level:
    for each scale level:
        > warp image pair[3,4]
        > calculate image gradients
        > create linear system[5]
        > solve
    end
    > scale velocity field for next pyramid level
end

> output velocity field
```

Figure 3.3: Pyramidal optical flow pseudocode. The location of our modifications are indicated with superscripts.
1. Image Intensity Normalization
2. Data-driven Boundary Conditions
3. Symmetric Warping
4. 2D Advection Warping
5. Spatially Varying Alpha

Spatially Varying Alpha

When analyzing synthetic images of channel flow with optical flow PIV, Kapulla et al. (2011) observed that the algorithm did not properly capture the boundary layer. They hypothesized that by using a globally optimum α, the smoothness close to the wall is too high to properly resolve the high velocity gradient. A proposed solution was to keep α high in the middle of the flow, where low gradients were present, and gradually decrease the value approaching the wall, where velocity gradients are higher—a velocity gradient dependent smoothing. To test the idea, the formerly scalar α of Equation 2.15 has been replaced with a 2D flow-driven smoothing function, defined generally in Equation 3.1.

$$\alpha_{i,j} \propto f\left(|\nabla \mathbf{u}_{i,j}|\right) \tag{3.1}$$

As before, $|\cdot|$ denotes the vector norm, and $\mathbf{u} = \sqrt{u^2 + v^2}$ denotes the magnitude of the velocity vector. The subscripts i, j are the discrete indices of the velocity field, indicating the function is defined for each pixel of the input images. To determine the mapping function of the velocity gradient to smoothness, the inverse velocity gradient/smoothness relation of Equation 3.2 is used, which was observed in experimental results using scalar α values, A is a fitting parameter.

$$|\nabla \mathbf{u}| = A \cdot \frac{1}{\sqrt{\alpha}} \tag{3.2}$$

From a simple rearrangement of Equation 3.2, a distributed smoothing function is proposed.

$$\alpha = \frac{\gamma}{|\nabla \mathbf{u}|^2} = \frac{\gamma}{\left(\frac{\partial \mathbf{u}}{\partial x}\right)^2 + \left(\frac{\partial \mathbf{u}}{\partial y}\right)^2} \tag{3.3}$$

The parameter γ is introduced to control the amount of smoothing applied for a given velocity gradient. Its function is similar to that of α, with higher values resulting in smoother flows. Although it might be tempting to equate γ with the coefficient A of Equation 3.2, the two are not interchangeable, since A is calculated over the entire flow field, using a scalar α. In the spirit of Nagel and Enkelmann (1986), the concept is extended to anisotropic, directional smoothing.

$$\alpha_x = \frac{\gamma}{|\nabla_y \mathbf{u}|^2} = \gamma \cdot \left(\frac{\partial \mathbf{u}}{\partial y}\right)^{-2} \tag{3.4}$$

$$\alpha_y = \frac{\gamma}{|\nabla_x \mathbf{u}|^2} = \gamma \cdot \left(\frac{\partial \mathbf{u}}{\partial x}\right)^{-2} \tag{3.5}$$

Analysis of synthetic images using the proposed isotropic and anisotropically distributed smoothness functions is presented in the following Results and Discussion chapter.

Improved Warping

Although successfully applied to rigid body motion common in video sequences, the unphysical nature of the mapping techniques often used in multi-scale optical flow poses a number of problems for optical flow PIV:

1. Commonly used bicubic interpolation does not accurately fit the intensity peaks of particles in PIV images.
2. Interpolation is not strictly valid for image data, recognizing that intensity values are *total photon counts* for the area of the sensor pixel. They are *not* samples of a continuous function, although interpolation treats them as such.
3. Donor pixel coordinates (inverse mapping) or destination pixel coordinates (forward mapping) are usually calculated linearly, disregarding any curved paths a pixel may travel.
4. Image intensity is not conserved, even when using higher-order methods to determine donor/destination pixel coordinates. For forward mapping, holes are filled in with extra data, and for inverse mapping, multiple pixels can have the same donor cell.

Each of these points was addressed in detail throughout the course of this work, and the outcomes are summarized below:

Peak Fitting: Resolution of PIV particle intensity peaks can be improved by replacing the bicubic interpolation function with a spline, as shown in Figure 3.4. Although slightly more difficult to calculate, Sun et al. (2010), and our own results have confirmed this to be a simple and effective way to improve flow field estimates. It should be noted that spline interpolations can lead to spurious overshoots, and when not corrected, can lead to an increasingly poorly-conditioned LES. An effective countermeasure is to clip interpolated values to the original intensity domain i.e. [0:1] or [0:255].

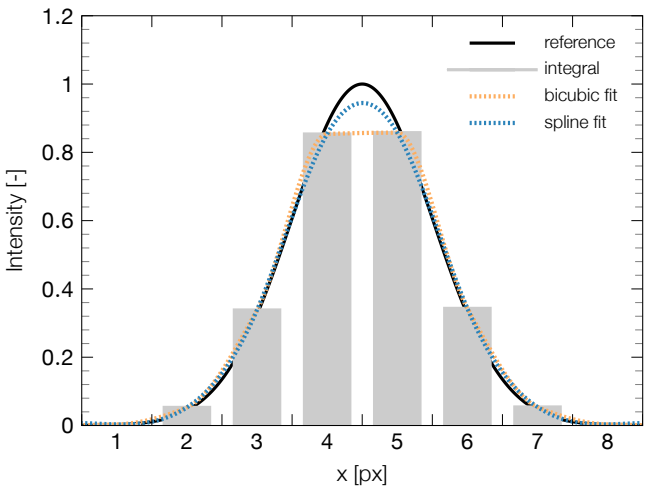


Figure 3.4: Bicubic vs. spline interpolation. The reference gaussian intensity (black) is integrated (i.e. digitized by a CCD sensor, shown in grey). The grey profile is then interpolated with bicubic (yellow) and spline (blue) functions to approximate the reference intensity profile.

Integral Interpolation: As an exercise, the author has programmed a method to fit a spline surface to image intensity data such that the 2D integral of the surface over each pixel is equal to the image intensity. The algorithm uses the highly efficient spline functions of Dierckx (1993) via the `scipy` package (Jones et al., 2001–2013) in a *regula falsi* scheme to determine the surface which reproduces the original image intensities when integrated. Unfortunately, improvements using this method were marginal compared to the computational expense.

Coordinate Calculation: To properly calculate the coordinates of the donor cell (inverse mapping) or destination cell (forward mapping), the integral effect of the velocity field over the entire journey of the pixel must be accounted for. Thus, the linear motion assumption commonly used in image mapping techniques is replaced with a fourth-order Runge-Kutta approximation, detailed for a steady-state function, f, in Equation 3.6.

$$\begin{aligned} x_{n+1} &= x_n + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) \\ t_{n+1} &= t_n + h \end{aligned} \tag{3.6}$$

with,

$$\begin{aligned} k_1 &= h \cdot f\left(x_n\right) \\ k_2 &= h \cdot f\left(x_n + \frac{1}{2}k_1\right) \\ k_3 &= h \cdot f\left(x_n + \frac{1}{2}k_2\right) \\ k_4 &= h \cdot f\left(x_n + k_3\right) \end{aligned}$$

In this case, the coordinates (x) of a pixel are moved iteratively through the velocity field (f). The number of iterations (n) are controlled by the step size (h). At each step, the four increments (k_1 to k_4) are weighted together with the current coordinate (x_n) to produce the next coordinate location (x_{n+1}). The velocity field is interpolated to provide the intermediate velocities used in the increments. An example inverse map coordinate calculation around a vortex is shown in Figure 3.5, to highlight the benefits of this method compared to the linear method.

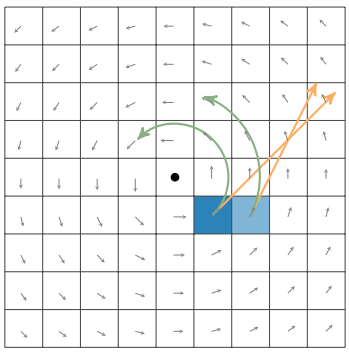


Figure 3.5: Mapping coordinate calculations around a counter-clockwise vortex. Linear coordinate calculations shown in yellow, fourth-order Runge-Kutta calculations shown in green, h=0.25.

The benefits of a higher-order approximation is clearly shown. When linear motion is assumed, the location of the donor cell is completely incorrect, but when a fourth-order approximation is used, the curvilinear nature of the flow field is followed to arrive at the expected location.

Although better than a first-order approximation, Figure 3.5 exposes a problem that even higher-order mapping schemes cannot address—the non-conservation of intensity. The problem for both forward and inverse mapping is illustrated more clearly in Figure 3.6 using the same vortical velocity field as before.

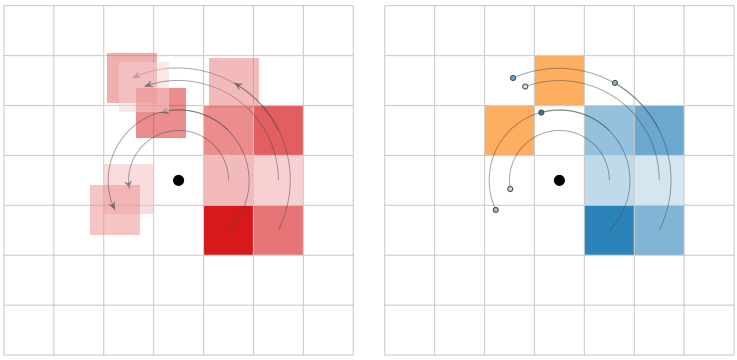


Figure 3.6: Illustration of the problems of forward and inverse mapping. In forward mapping (left), pixel coherence is completely broken up. Inverse mapping (right) improperly samples the data, with one pixel being sampled twice, and two pixels (indicated in yellow) never being sampled.

In the case of forward mapping, the "particle" which would hopefully stay together, is instead broken up. If this field were interpolated, intensity would invariably be added to regions where pixels did not appear. In the case of inverse mapping, there is a similar scenario, where information is misrepresented, some pixels are ignored (shown in yellow), while others are sampled twice. Since the mapping technique does not incorporate any *a priori* information about the object(s) being warped, it is difficult to correct for this information addition/loss. In the end, many attempts were made in an attempt to solve the shortcomings of mapping, but the technique was eventually abandoned in favour of a method showing promise in addressing interpolation and motion reconstruction inclusively.

2D Advection Warping

To overcome the non-conservative nature of traditional warping methods, the procedure has been re-formulated as a conservative scalar advection, formally in Equation 3.7, for a quantity ϕ in the velocity field $\mathbf{u}$.

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\phi \mathbf{u}) = 0 \tag{3.7}$$

For two dimensions, this expands to[1]:

$$\frac{\partial \phi}{\partial t} + u\frac{\partial \phi}{\partial x} + v\frac{\partial \phi}{\partial y} = 0 \tag{3.8}$$

The advected scalar quantity, ϕ, being pixel intensity in our case. As an aid, the scalar quantity (pixel intensities) can be visualized as voxels of fluid, where the height is proportional to the intensity value, as illustrated in Figure 3.7.

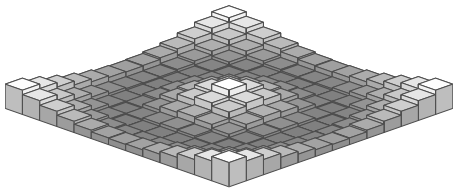


Figure 3.7: Intensity voxels over a regular 2D domain.

Implied from Figure 3.7, is that that pixel intensity is treated as an integral value—which is valid for digital images. Additionally, since intensity is advected around in incompressible quantities, it is inherently conserved, there is no need to resort to any interpolation as with the mapping methods. Furthermore, if the advection is properly implemented, and all voxel motions are taken into account, vortical motions will be implicitly handled, since intensity can freely travel from cell to cell. As an added benefit, the domain is a regular image grid, so in comparison to typical CFD codes solving the advection equation, the implementation is greatly simplified.

However unassuming, Equation 3.7 has proved notoriously difficult to solve in practice. Scientific literature abounds in proposed techniques to elicit acceptable solutions such as high-order differencing, flux limiters, and spectral schemes. For a simple proof of concept, these techniques have been eschewed, and the basic upwind differencing scheme of Equation 3.9 is used. Here, a is the advection velocity in pixels per timestep, and Δx is the grid size. The corresponding computation stencil is illustrated in Figure 3.8.

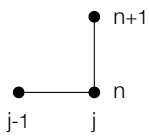


Figure 3.8: Computational stencil for first-order upwind advection. j is the space coordinate, n is the time coordinate.

[1]Note the similarity with the BCCE of Equation 2.5

$$\phi_j^{n+1} = \begin{cases} \phi_j^n - \frac{a\Delta t}{\Delta x}(\phi_j^n - \phi_{j-1}^n) & \text{if } a > 0 \\ \phi_j^n - \frac{a\Delta t}{\Delta x}(\phi_{j+1}^n - \phi_j^n) & \text{if } a < 0 \end{cases} \tag{3.9}$$

It is recognized that the upwind method of Equation 3.9 is highly dissipative, with sharp intensity peaks being eroded and spread out over the advection. But, since symmetric warping is used, the two waves are advected towards each other using the same velocity field. In this way, the dissipation of both peaks is equal, as shown in Figure 3.9, for a CFL = 0.25[1].

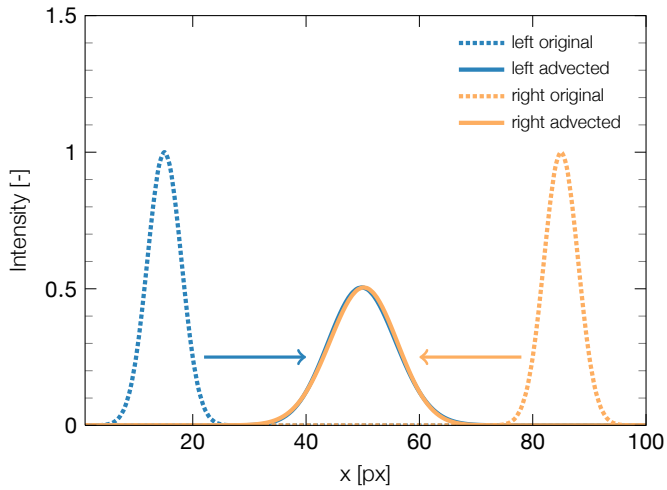


Figure 3.9: 1D upwind advection of two gaussian peaks towards each other, CFL = 0.25.

Of more concern for symmetric warping is the highly non-linear phase error of the upwind scheme which must be corrected (visible as a slight misalignment in the advected curves of Figure 3.9). In our case, phase errors slow down or speed up the apparent motion of advected features. When coupled with an optical flow code, these phase errors cause biased velocity predictions. For example, a phase lag at a low CFL causes advected particles to move more slowly than they should. The optical flow algorithm, in turn, overcompensates with a faster (incorrect) flow field. To further complicate matters, phase error depends on wavelength, with shorter waves incurring larger errors. This is especially problematic for PIV, where the particles are only a few pixels in diameter, corresponding to a short wavelength.

To demonstrate the complexity of the phase error, ϕ_j^{n+1} and ϕ_j^n are replaced with the finite Fourier series of Equations (3.10) and (3.11) and von Neumann stability analysis is performed.

$$\phi_j^{n+1} = A_k e^{ikj\Delta x} \tag{3.10}$$

$$\phi_j^n = e^{ikj\Delta x} \tag{3.11}$$

The solution, ϕ_j^{n+1}, is multiplied with the amplification factor A_k, and $k\Delta x$ is the wavelength as a multiple of the grid step.

[1] CFL $= \frac{a\Delta t}{\Delta x}$ from Courant et al. (1928). In this case, enforces the condition that the advection cannot be faster than 1 pixel per timestep.

Substituting into Equation 3.9, and defining $\mu = \frac{a\Delta t}{\Delta x}$, the CFL number.

$$A_k e^{ikj\Delta x} = (1-\mu)e^{ikj\Delta x} + \mu e^{ik(j-1)\Delta x} \tag{3.12}$$

Simplifying,

$$A_k = (1-\mu) + \mu e^{-ik\Delta x} \tag{3.13}$$

and using Euler's formula,

$$e^{ix} = \cos x + i \sin x \tag{3.14}$$

$$A_k = (1-\mu) + \mu\big(\cos(k\Delta x) - i\sin(k\Delta x)\big) \tag{3.15}$$

The phase (for the numerical scheme, denoted θ_d) is the inverse tangent of the ratio of imaginary to real components of Equation 3.15.

$$\theta_d = \tan^{-1}\left(\frac{\text{Im}}{\text{Re}}\right) = \tan^{-1}\left(\frac{-\mu\sin(k\Delta x)}{1-\mu+\mu\cos(k\Delta x)}\right) \tag{3.16}$$

This is in contrast to the phase of the analytical solution θ_a,

$$\theta_a = -\mu k\Delta x \tag{3.17}$$

Using the ratio of both analytical and numerical phase, the phase error can be computed, and is illustrated in Figure 3.10 relative to the advection speed.

$$\theta_{\mathcal{E}} = \frac{\theta_d}{\theta_a} \tag{3.18}$$

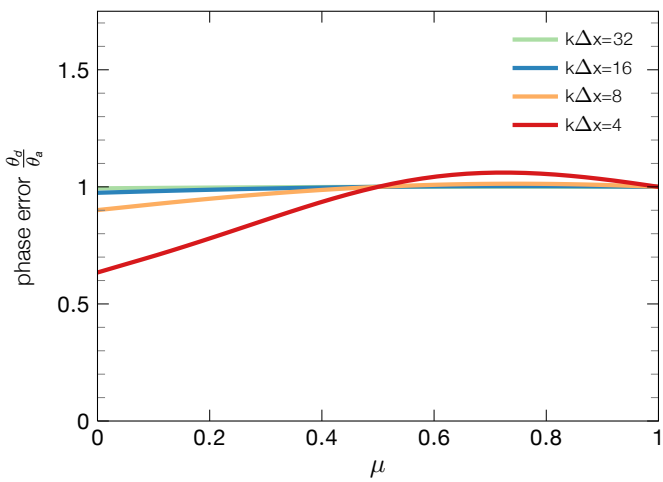


Figure 3.10: Phase error θ_d/θ_a of the upwind scheme vs. μ (CFL number) for different wavelengths.

From Figure 3.10, it can be seen that the wave lags when it travels less than half a grid length per time step (CFL < 0.5), and is accelerated when it travels faster than half a grid length (0.5 < CFL < 1). When the wave travels *exactly* a full or half a grid length, there is no phase error, and the result is exact—the desired behaviour.

While it is difficult to eliminate phase errors of advection schemes completely, an effective technique to reduce them has been proposed by Fromm (1968). By averaging two second-order advection schemes with opposite phase errors, the proposed technique reduces phase errors considerably. Figure 3.11 compares the phase error of Fromm's method with simple second-order and fourth -order schemes for short wavelengths (low $k\Delta x$).

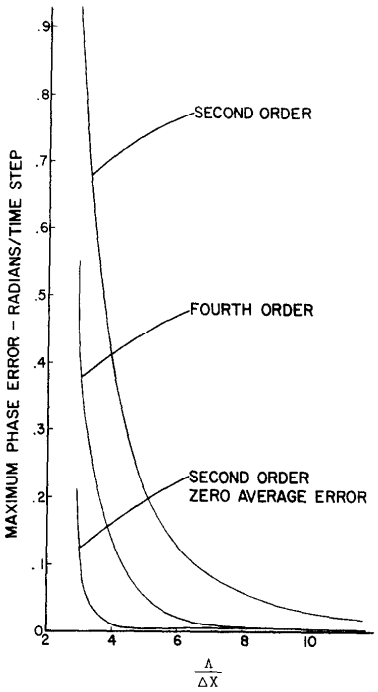


Figure 3.11: Performance of Fromm's method at short wavelengths, Λ = wavelength, Δx = grid step, from Fromm (1968).

The proposed method performs exceptionally well for short wavelengths, and is much better than second or simple fourth-order methods in advecting the small particles present in PIV images. Using this method, a 2D advection warping method for optical flow PIV has been developed. Our proposed implementation is detailed in Appendix C, and is a slight modification of that proposed by Fromm (1968), which for simplicity, calculates u and v components in one step.

During development, a couple of tricks have been used to further improve results. First, the advection is run as fast (with CFL as close to 1) as possible, as recommended by Kim (2003). Second, the input images are super-sampled with a spline surface before being advected. With every doubling of resolution, the wavelength is effectively doubled, and the wave number shifts to the right of Figure 3.11. Care must be taken, however, as super-sampling introduces interpolation errors, and for every doubling of resolution, the number of time steps increases proportionally (to satisfy the CFL condition). It can be, that the accrued phase error over more time steps is larger than that caused by short wavelengths. A simple doubling of resolution has been found to be the best compromise.

Image Replacement

In addition to accurate advection, missing information after warping must be rigorously accounted for. Take for example the symmetrically warped images of a poiseuille flow in Figure 3.12. During the process of warping, parts of the image are warped out of the frame (shown in red), and there is a lack of information following the image warped into the frame. If the missing information is not compensated, incorrect velocities are calculated. The most effective method found thus far to address this issue is to exploit the symmetric nature of the warp, which, (barring corner-cases) ensures the information missing from one image can be copied from the other. This perfect copying fully engaging the smoothing term and in effect, the replaced region becomes a non-divergent, irrotational interpolation of the valid velocity vectors of non-replaced image portions. The advantage of image replacement should be emphasized, as it provides much more accurate and stable results than simply setting these regions to a uniform value.

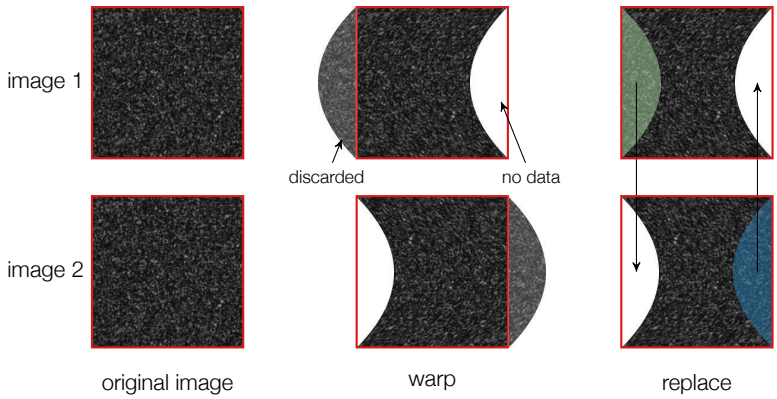


Figure 3.12: Illustration of information loss and replacement during the symmetric warping of a poiseuille flow.

Determining the portion of an image that needs to be replaced, however, is not trivial. Currently, the velocity vector of each pixel is checked to see if it extends outside of the frame, and if it does, that pixel is flagged for replacement. This method has obvious drawbacks for highly curved or divergent flow fields, but is reasonably conservative and fast.

Data-driven Boundary Conditions

In an attempt to improve the optical flow algorithm performance in the boundary layer, a way to impose boundary conditions has been developed. Absent from all literature the author has seen on optical flow, the proposed method is a simple and effective way to impose physical constraints on the velocity field calculation reflecting experimental conditions. The method exploits the algorithm's treatment of motion normal to intensity gradients by adding monolithic white image features to the intensity-normalized image on the edges where the user wishes to impose the conditions. Since this method only affects the data term of the minimization equation, it is referred to as "data-driven".

In the case of a parallel flow boundary, a series of 2 pixel wide white bars are added (Figure 3.13), and for a no-slip (zero velocity) boundary, the author has found the Gray code inspired pattern of Figure 3.14 to perform exceptionally well. The no-slip condition has been designed to be robust for large displacements, with the pattern repeating every 108 pixels. Both patterns are quite compact, adding only 10 pixels to the image on the edge they are applied, and do not significantly affect computation time.

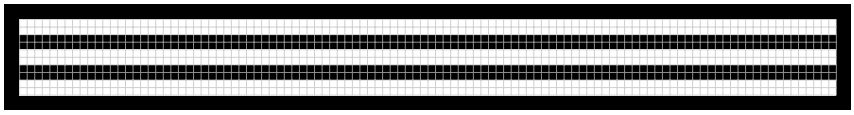


Figure 3.13: Parallel boundary condition shown on a pixel grid.

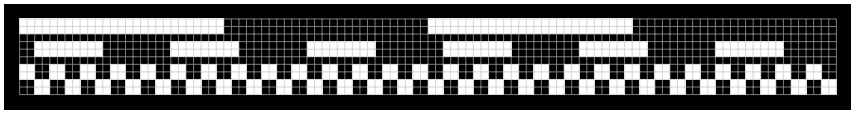


Figure 3.14: No-slip boundary condition shown on a pixel grid.

Although the proposed methods are very effective, it should be kept in mind that optical flow is an energy minimizing method. Given a very dominant smoothing term (i.e. high α), the boundary conditions may not be as strictly enforced as say, modifying the program code. However, it remains a simple technique to apply physically motivated constraints with unmodified optical flow algorithms.

In designing no-slip boundary conditions, the following has been found to be beneficial:

- no overlapping white areas between rows
- unique pattern over many pixels
- compact patterns
- reducing the size of the white pixels near the no-slip edge to 1×1 for flows with high gradients.

Results and Discussion

To quantify the effect of our proposed modifications to the optical flow algorithm, select analyses of synthetic and real image sequences are presented. In all cases symmetric warping was used, on 5 pyramid levels, and 6 scale levels—a combination providing good results with reasonable processing time. Symmetric warping is performed with spline interpolation, and results are calculated from only two images, unless otherwise indicated. Velocities are presented in units of pixels/frame, easily converted to physical units with an appropriate scaling factor for the experimental images.

Image Sets

Most code development and synthetic test results were calculated with the FLUid Image analysis and Description (FLUID) test images (Carlier, 2005). The fundamental nature of these data sets allows us to test the algorithm under varying conditions of translational, vortical, divergent, and viscous flows. The database also contains a turbulent image set generated by a direct numerical simulation (DNS) of experimental results. This physically motivated, complex flow field is exceptionally challenging to analyze, and makes an excellent benchmark for PIV algorithms. An example PIV image, and visualizations of the "turbulent" and "cylinder with circulation" flow fields from this dataset are presented in Figure 4.1.

Since this work is affiliated with GEMIX experiments, an analysis of synthetic images generated from large eddy simulations of the experiment is also included. The images were generated using the EUROPIV Synthetic Image Generator (Lecordier & Westerweel, 2004). Images were generated for time steps range from 0.0025 - 0.02 s, and are 256×256 px in size. Lastly, results are presented from actual GEMIX experiments, along with proof-of-concept analysis of experimental laser-induced fluorescence (LIF), and near infrared (NIR) images. For clarity, in the following sections recurring image sets are referred to with shortened names;

poiseuille	synthetic FLUID image set a1b, frames 1–2
turbulent	synthetic FLUID self-sustaining turbulence images, frames 50980–50990
GEMIX_ref	synthetic GEMIX images calculated with dt = 0.01.

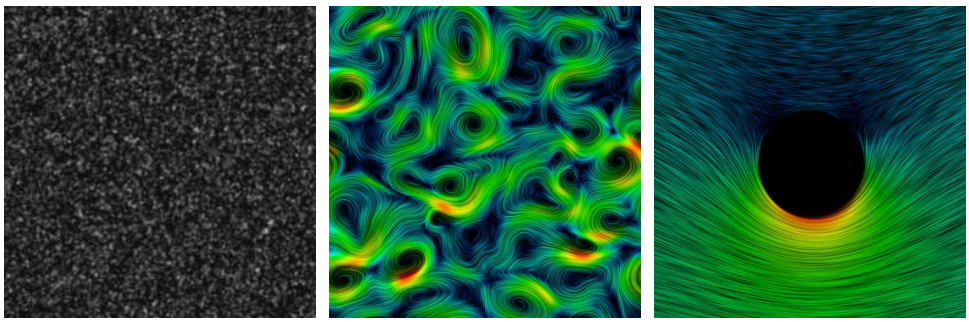


Figure 4.1: Example FLUID synthetic PIV image (left), along with an illustration of the turbulence flow field (middle) and a cylinder with circulation (right).

Determining Optimum Smoothness

To fairly compare the merits of each method in the next sections, results are presented at their optimum α value. This is determined by calculating flow fields starting with a uniformly distributed α of 100, and decreasing down the decades to 0.001 or 0.0001. The flow field with the lowest average endpoint error (AEE, Equation 2.22) is determined to be optimally smoothed. It is entirely justifiable to use another metric such as average magnitude, or angular error to determine the optimally smoothed flow, but the AEE is chosen because it intuitively incorporates both directional and magnitude errors.

Example evolutions of select errors from page 17 over varying α values are presented in Figures 4.2 and 4.3. Values are presented without units, to demonstrate the relative position of the minima. It is interesting to note the behaviour of the minimum warp RMS relative to the other error metrics. While the minimum occurs at a lower α, it always seems to reside at the lower end of an optimal α "plateau".

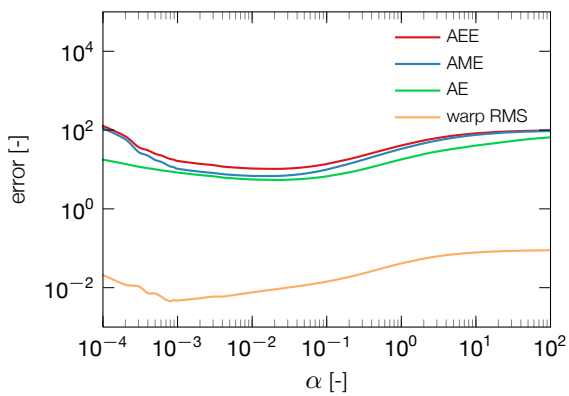


Figure 4.2: Error evolution for the *turbulent* image set at various α values.

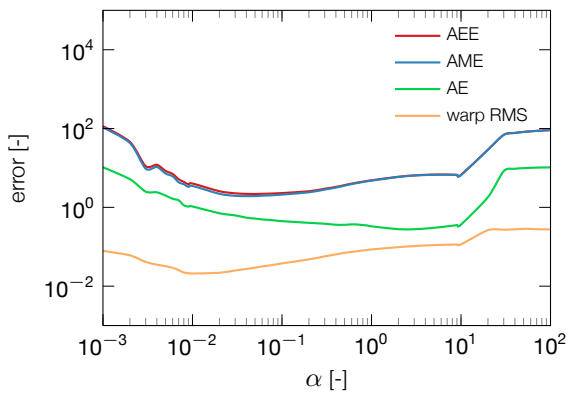


Figure 4.3: Error evolution for the *GEMIX_ref* image set at various α values.

Spatially Varying Alpha

To illustrate how Equation 3.2 (the α distribution function) is obtained, two example α sweeps typical for synthetic images are presented. For each scalar α value, the maximum velocity gradient (MVG) is calculated according to Equation 4.1.

$$\mathrm{MVG} = \max\left(|\nabla \mathbf{u}|\right) = \max\left(\sqrt{\left(\frac{\partial \mathbf{u}}{\partial x}\right)^2 + \left(\frac{\partial \mathbf{u}}{\partial y}\right)^2}\right) \tag{4.1}$$

And a fit is calculated according to Equation 4.2.

$$\mathrm{MVG}_{fit} = A \cdot \frac{1}{\sqrt{\alpha}} \tag{4.2}$$

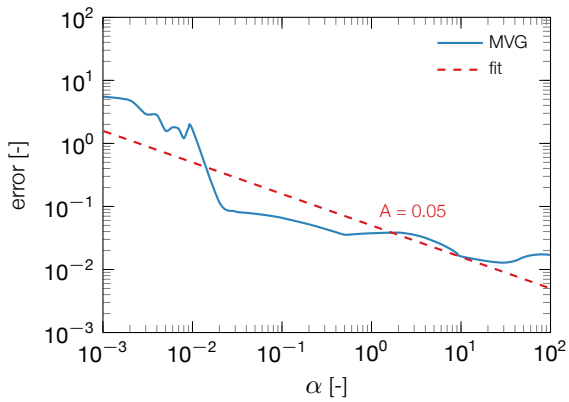


Figure 4.4: Maximum velocity gradient evolution for the *poiseuille* image set.

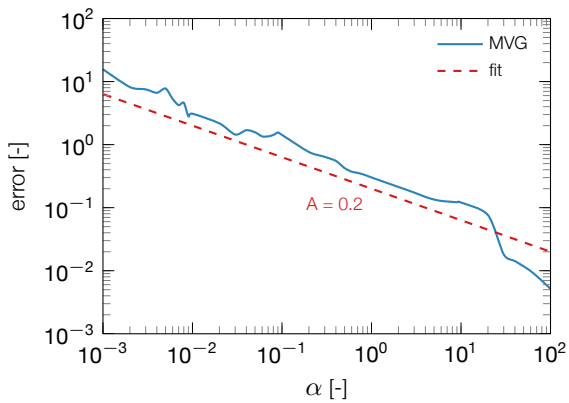


Figure 4.5: Maximum velocity gradient evolution for the *GEMIX_ref* image set.

To demonstrate the effect of spatially varying smoothness, presented in Equations 3.3 to 3.5, results are presented for the *poiseuille*, *GEMIX_ref*, and *turbulent* image sets. The scaling coefficient, γ, in each case is chosen so that the average distributed smoothness function, or "α map" approximately equals the optimum uniform α. In all cases, the α maps have been clipped to an upper and lower bound and then smoothed with a gaussian filter. These steps are necessary to ensure the LES is solvable. The conditioning parameters are shown Table 4.1, and the resulting isotropic α maps are presented in Figures 4.6 to 4.8.

Table 4.1: Parameters used to condition the distributed α fields. In all cases a gaussian filter was used, with $\sigma = 0.039$.

Image Set	Optimal α	Isotropy	γ	lower bound	upper bound
poiseuille	0.2	isotropic	5E-6	5E-6	10
	0.2	anisotropic	1E-7	1E-7	10
GEMIX_ref	0.05	isotropic	1E-5	1E-5	10
	0.05	anisotropic	1E-7	1E-7	10
turbulent	0.02	isotropic	5E-7	1E-7	100
	0.02	anisotropic	5E-7	1E-7	25

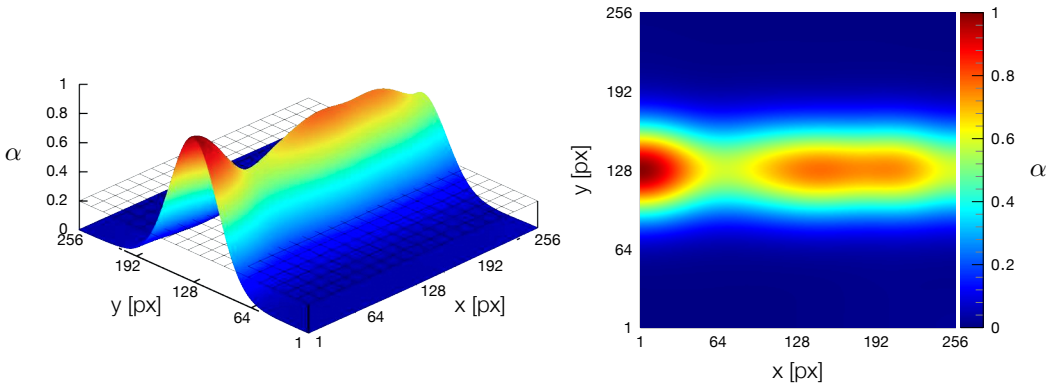


Figure 4.6: Isotropic α distribution for the *poiseuille* image set. The mesh in the 3D plot indicates the optimal uniform value.

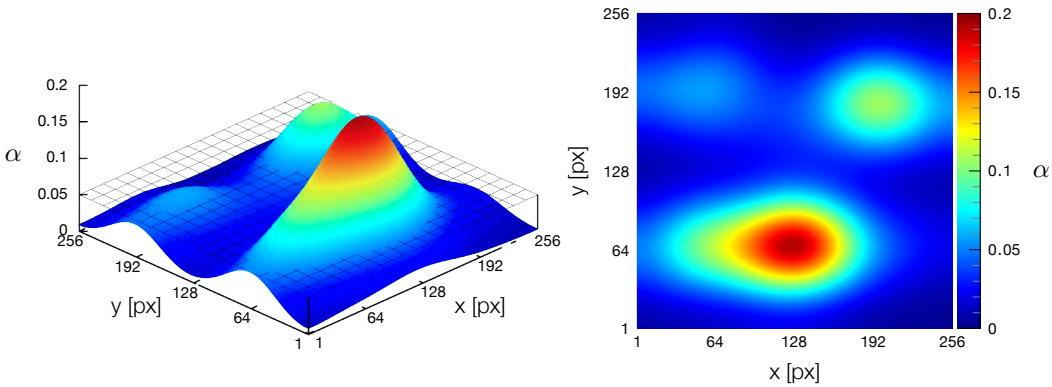


Figure 4.7: Isotropic α distribution for the *GEMIX_ref* image set. The mesh in the 3D plot indicates the optimal uniform value.

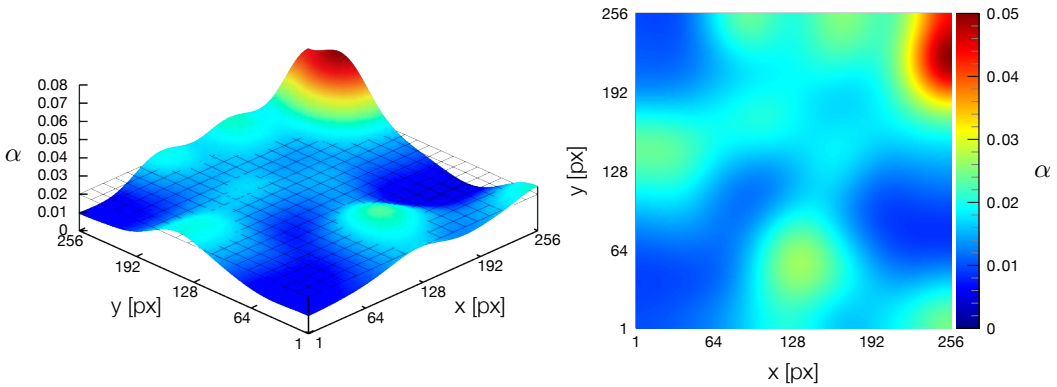


Figure 4.8: Isotropic α distribution for the *turbulent* images set. The mesh in the 3D plot indicates the optimal uniform value.

The average endpoint error (AEE) for each of the α distribution methods is presented in Table 4.2. The u and v velocity profiles taken from a vertical line in the middle of the flow fields are presented for the three image sets in Figures 4.9 to 4.14.

Table 4.2: Summary of average endpoint error for uniformly, isotropically, and anisotropically distributed smoothing. The best performing cases are highlighted in bold.

Image Set	Uniform AEE [%]	Isotropic AEE [%]	Anisotropic AEE [%]
poiseuille	**1.05**	4.82	2.35
GEMIX_ref	2.09	1.78	**1.00**
turbulence	**10.33**	10.39	10.34

From Table 4.2 it can be seen that for the *GEMIX_ref* image set, anisotropically distributing α provides more accurate results than a simple uniform value. Using a distributed smoothing approach relaxes the over-smoothing in the boundary layer, as can be seen in Figures 4.9 and 4.11. However, the effect is marginal, and the velocity profile in the last two pixels still deviates quite strongly from the actual value. For the *poiseuille* and *turbulent* flows, distributing α is actually *less* accurate than simply using a uniform value. This was an unexpected result, and many attempts were made to find an explanation. It wasn't until a way to impose boundary conditions (presented in its own section) to the flow field was developed that the reason became clear. Basically, errors in resolving the boundary region are largely not caused by over-smoothing.

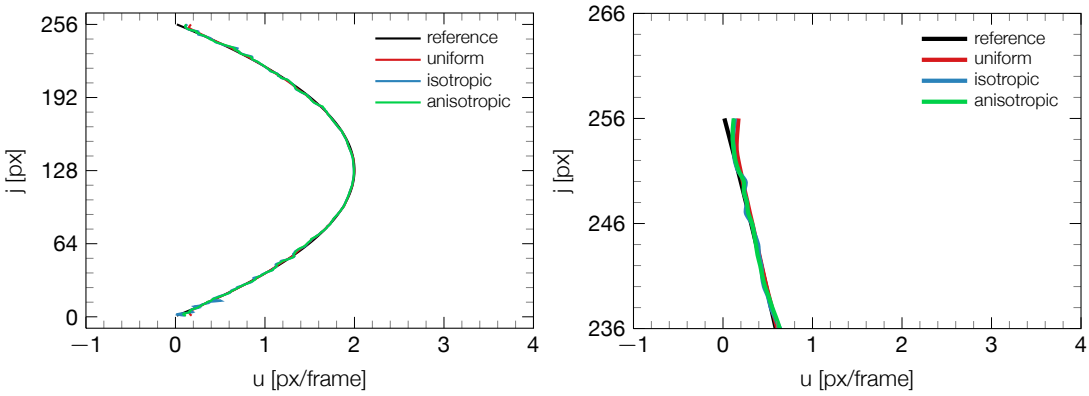


Figure 4.9: u velocity profiles of the *poiseuille* image set with distributed smoothing. Boundary region enlarged on right.

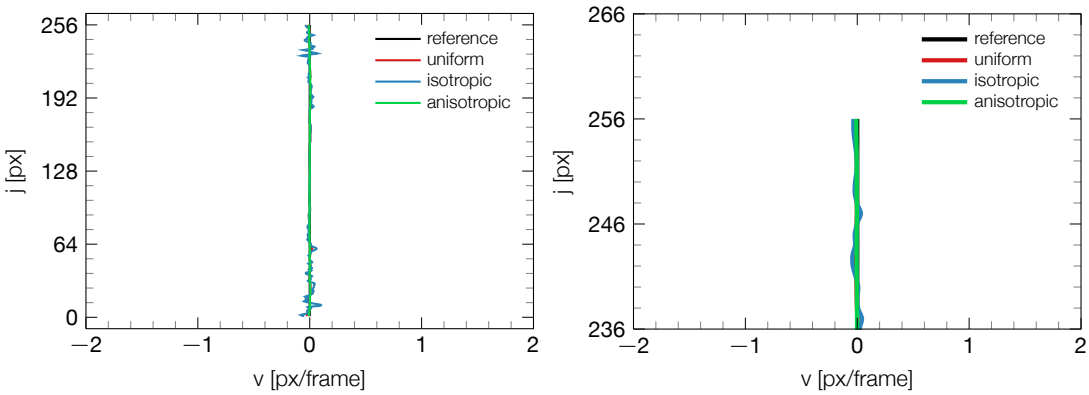


Figure 4.10: v velocity profiles of the *poiseuille* image set with distributed smoothing. Boundary region enlarged on right.

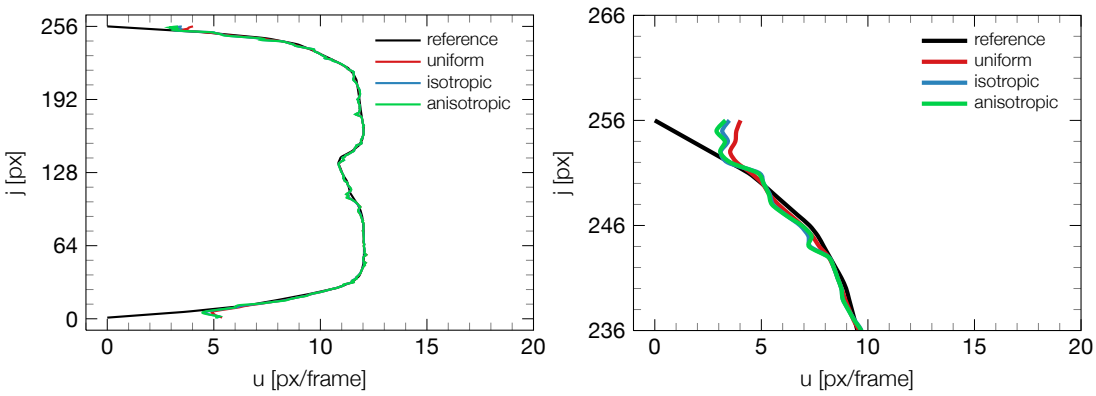


Figure 4.11: u velocity profiles of the *GEMIX_ref* image set with distributed smoothing. Boundary region enlarged on right.

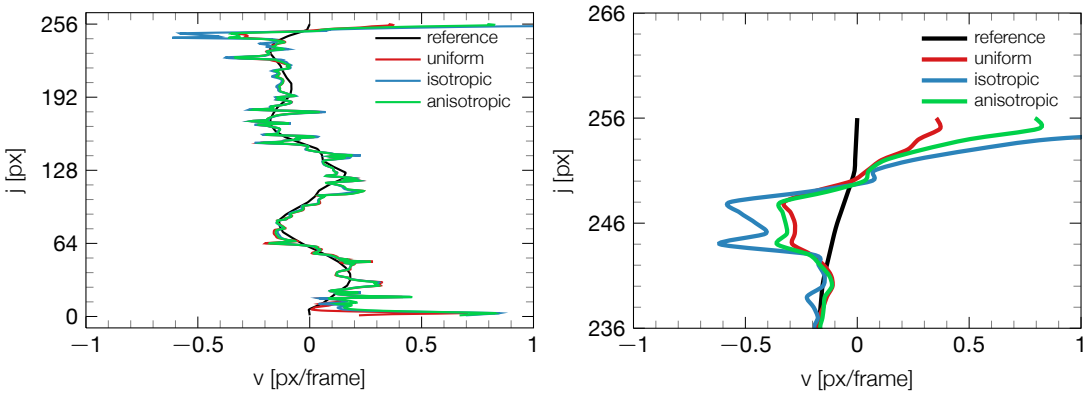


Figure 4.12: v velocity profiles of the *GEMIX_ref* image set with distributed smoothing. Boundary region enlarged on right.

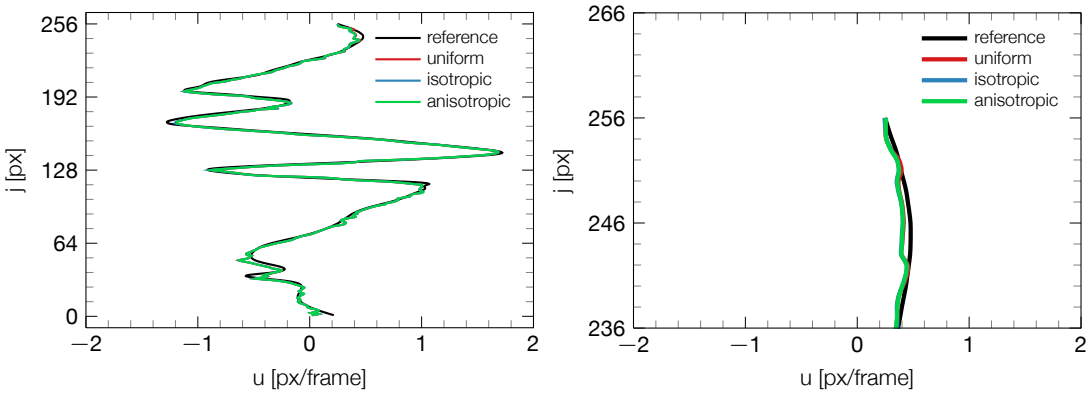


Figure 4.13: u velocity profiles of the *turbulent* image set with distributed smoothing. Boundary region enlarged on right.

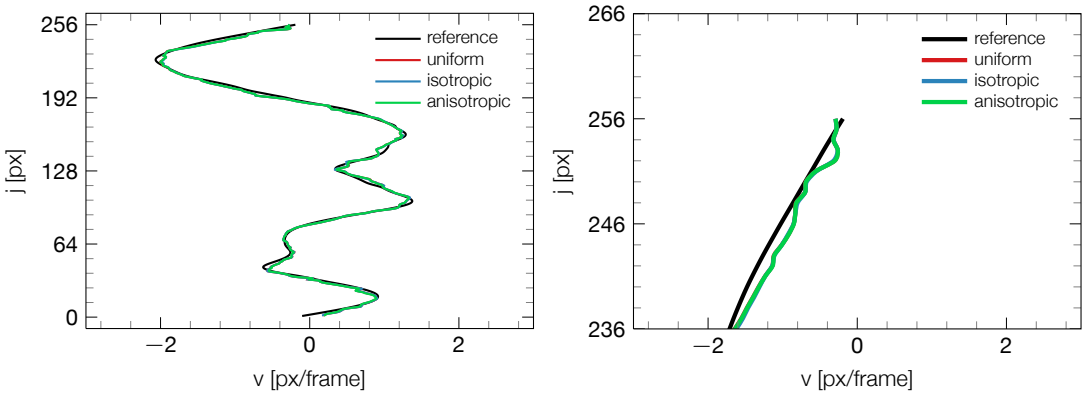


Figure 4.14: v velocity profiles of the *turbulent* image set with distributed smoothing. Boundary region enlarged on right.

Comparison of Warping Methods

To compare the accuracy of different warping methods, the minimum average endpoint error (AEE, Equation 2.22) achievable with each of the three methods: traditional bicubic backward mapping, and the proposed spline backward mapping, and scalar advection is reported. Results are presented in Table 4.3 for the range of synthetic GEMIX and FLUID images.

Table 4.3: Comparison of symmetric warping techniques based on minimum average endpoint error. The best performing cases are highlighted in bold.

			Bicubic		Spline		Advection	
ID	Description	Notes	α	Error [%]	α	Error [%]	α	Error [%]
1	GEMIX 1	dt = 0.0025	0.1	2.07	**0.2**	**1.68**	0.2	1.74
2	GEMIX 2	dt = 0.005	0.07	1.95	**0.1**	**1.64**	0.2	1.83
3	GEMIX 3	dt = 0.0075	0.05	1.96	**0.07**	**1.83**	0.09	2.07
4	GEMIX 4	dt = 0.01 *(GEMIX_ref)*	0.04	2.40	**0.05**	**2.19**	0.08	2.56
5	GEMIX 5	dt = 0.0125	0.07	2.83	**0.07**	**2.66**	0.1	3.00
6	GEMIX 6	dt = 0.015	0.04	3.37	**0.04**	**3.35**	0.06	3.76
7	GEMIX 7	dt = 0.02	0.05	4.78	**0.04**	**4.76**	0.08	5.24
8	FLUID 1	poiseuille flow *(poiseuille)*	0.07	2.69	0.2	1.05	**0.2**	**0.91**
9	FLUID 2	lamb-oseen vortex	6.0	8.42	0.7	2.16	**0.4**	**1.52**
10	FLUID 3	uniform translation	3.0	0.17	**1.0**	**0.12**	0.003	0.69
11	FLUID 4	pure sink	0.2	25.71	0.2	10.15	**0.2**	**9.41**
12	FLUID 5	point vortex	0.2	25.66	0.3	10.36	**0.4**	**9.65**
13	FLUID 6	cylinder w/circulation	0.004	14.95	0.3	11.43	**0.4**	**11.14**
14	FLUID 7	affine transform	4.0	12.90	0.7	3.07	**0.6**	**2.15**
15	FLUID 8	turbulent DNS *(turbulence)*	0.02	11.66	0.02	10.33	**0.02**	**10.22**

In all instances, an improvement over the traditional bicubic warping method is seen. The advection warp developed for this thesis performs very well for the FLUID images, but has an increasingly difficult time with GEMIX images. The degrading performance when analyzing the GEMIX images is attributed to larger pixel displacement, which increases the number of iterations required for each warp and causes compounding phase errors.

Data-driven Boundary Conditions

To demonstrate the effectiveness of applying boundary condition patterns to images, the *poiseuille* and *GEMIX_ref* image sets are used. Parallel and no-slip boundary conditions are applied to both image sets (see Figure 4.15), and the average endpoint errors are compared to the control (no boundary condition) case. A summary is presented in Table 4.4, with α constant for an image set. Flow profiles extracted from the middle of the flow fields are presented in Figures 4.16 to 4.19.

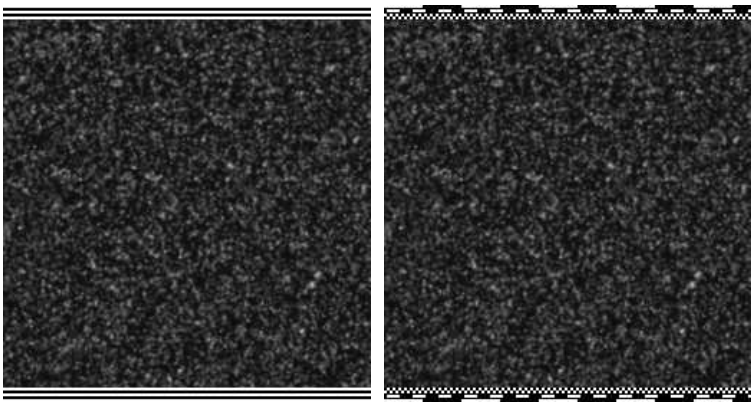


Figure 4.15: Synthetic PIV images with boundary condition features applied. Parallel flow boundary condition on the left, no-slip boundary condition on the right.

Table 4.4: Minimum average endpoint error achieved with different boundary conditions.

Image Set	Boundary	Average Endpoint Error [%]
poiseuille ($\alpha = 0.2$)	none	1.05
	parallel	1.36
	no-slip	**0.73**
GEMIX_ref ($\alpha = 0.05$)	none	2.19
	parallel	2.40
	no-slip	**1.02**

As both ground truth flow fields actually have zero-velocity at the walls, it is not surprising to see the no-slip boundary condition perform so well. It is rather surprising to see the parallel boundary condition actually increase the error, but this is due to a sub-optimal α selection, since α is constant for a given image set. Figures 4.16 and 4.18 show the results in more detail, with the no-slip boundary showing improved accuracy in resolving the boundary. Since the correct velocity profile was extracted using a fixed α, it can be concluded that problems resolving the boundary layer are *not* caused by over-smoothing.

It should be noted that in reality, it is highly unlikely that particles near a wall will be motionless, even if the velocity at the wall is exactly zero. The gradient of the boundary layer will act on the finite diameter of the particle, causing a velocity bias (see Kähler et al. (2012) for a detailed treatment of this, and other near-wall phenomena).

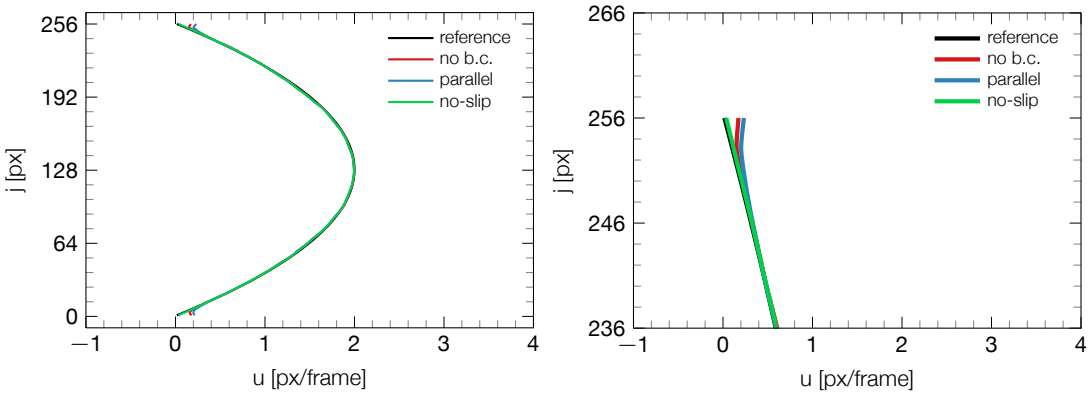


Figure 4.16: u velocity profile of the *poiseuille* flow with various boundary conditions. The boundary layer region is enlarged on the right.

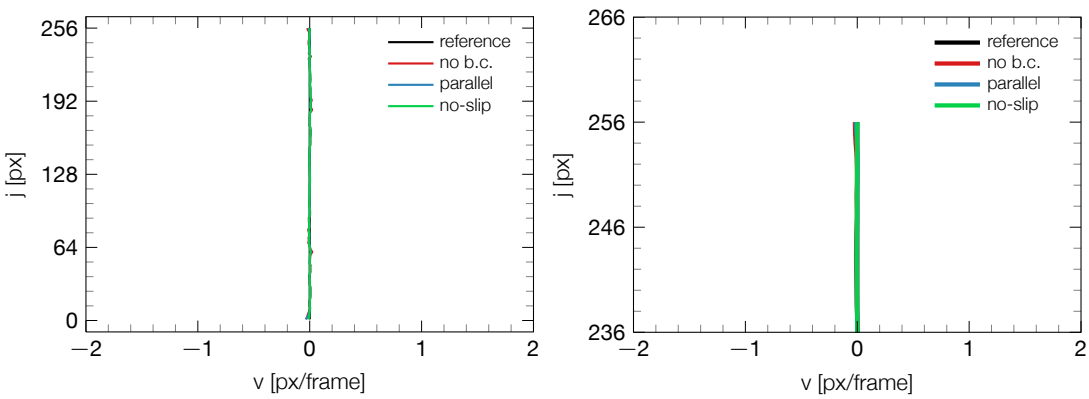


Figure 4.17: v velocity profile of the *poiseuille* flow with various boundary conditions. The boundary layer region is enlarged on the right.

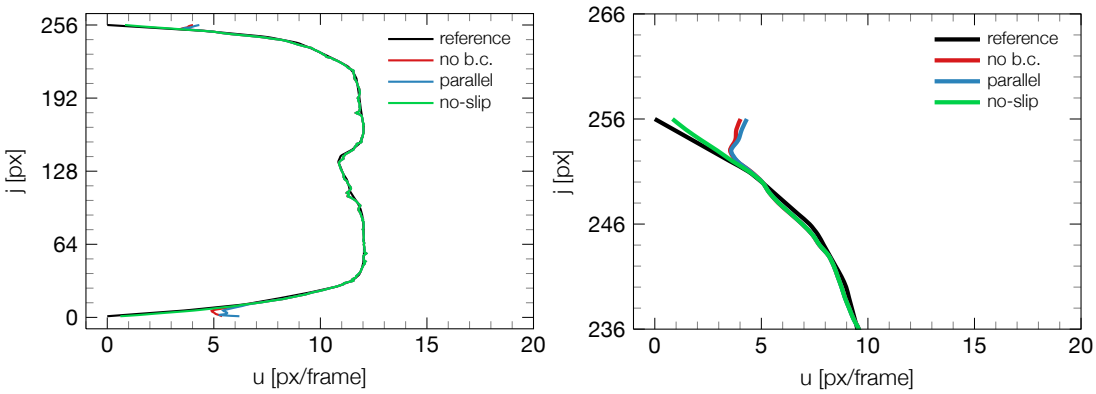


Figure 4.18: u velocity profiles of the *GEMIX_ref* flow with various boundary conditions. The boundary layer region is enlarged on the right.

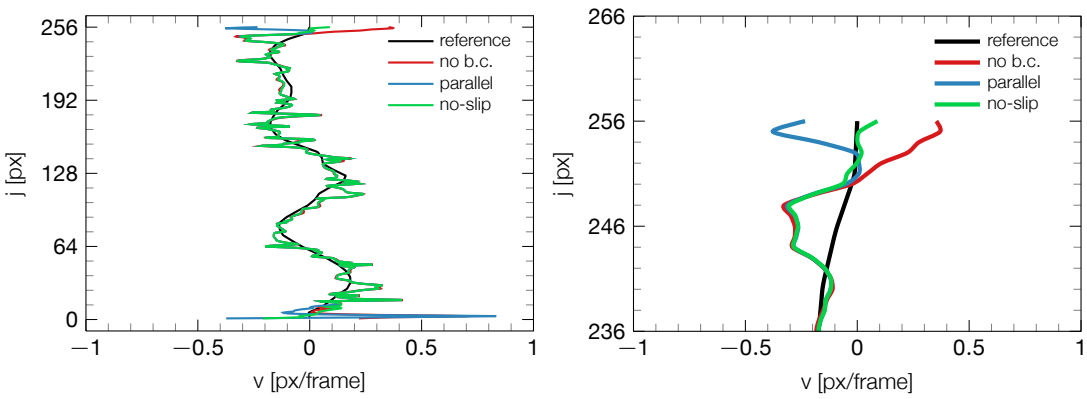


Figure 4.19: v velocity profiles of the *GEMIX_ref* flow with various boundary conditions. The boundary layer region is enlarged on the right.

Experimental Results

One last challenge remains before it is possible to analyze experimental GEMIX images—selecting the appropriate smoothness term. Since the ground truth velocity field is unknown, it is not possible to calculate the minimum endpoint error as before. Facing a seemingly intractable problem, inspiration is taken from the non-weighted warp RMS error method of Tu et al. (2012), and we propose a method to select an appropriate α.

1. Perform an α sweep of a single image pair, calculating the warp RMS error (Equation 2.23) and maximum velocity gradient (MVG, Equation 4.1) at each step.
2. Plot the curves with respect to α, as shown in Figure 4.21.
3. If the maximum velocity gradient is known, either analytically or from CFD simulations, convert it to units of pixels, and select the appropriate α from the plot. It is usually best to be conservative and choose a slightly lower value than estimated.
4. Otherwise, choose α at the minimum warp RMS error. While this may not be low enough to capture the strongest gradients in the flow, it is the lowest α that can be used before the algorithm becomes under-determined for the given images.
5. Check that the velocity field appears reasonable. Especially with experimental images, noise can cause very obvious errors at image edges. If there are problems, choose a slightly larger α and re-check the calculated velocity field.
6. With the chosen α value, process the entire image series.
7. As a final check, ensure Equation 4.3 is true by computing the maximum velocity gradient for each image pair and average the values over the series. Ensure there is a sufficient margin between this value and the maximum velocity gradient calculated from the average velocity field.

$$\frac{1}{n}\sum_{i}^{n}\max|\nabla\mathbf{u}|_i \;>\; \max\left|\nabla\left(\frac{1}{n}\sum_{i}^{n}|\mathbf{u}|_i\right)\right| \tag{4.3}$$

To demonstrate the effectiveness of this method, it is applied to a series of the most challenging experimental GEMIX images (series N325), with an average velocity of 1 m/s (5-6 pixels/time step), and a density difference of 10%. As can be seen from Figure 4.20, refractive distortion is extremely high, with large intensity variations and severe blurring in the mixing region.

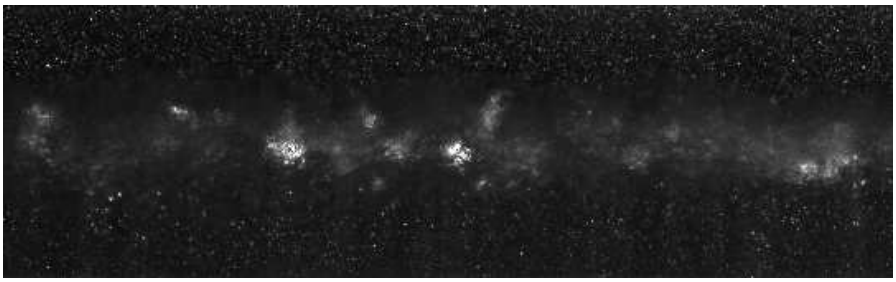


Figure 4.20: GEMIX experimental PIV image from series N325.

As before, symmetric spline warping is used, with intensity normalization also applied. No-slip boundary conditions are added on the top and bottom of the image (where the channel walls are), to mirror experimental conditions. This has an added benefit of stabilizing the images, allowing the use of a lower α, which enables the capture of larger velocity gradients, as shown in Figure 4.21. From the minimum warp RMS value, $\alpha = 0.01$ is selected, and the velocity fields are checked.

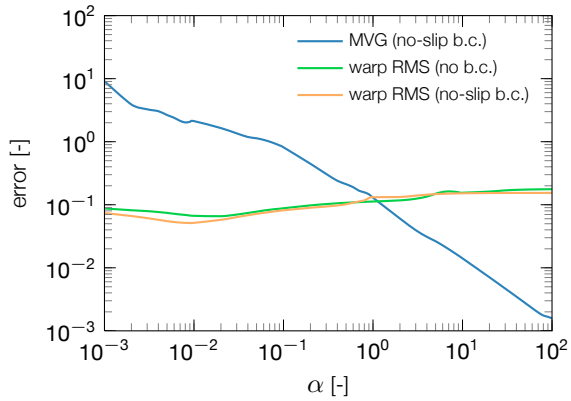


Figure 4.21: Series N325 α sweep for a single image pair.

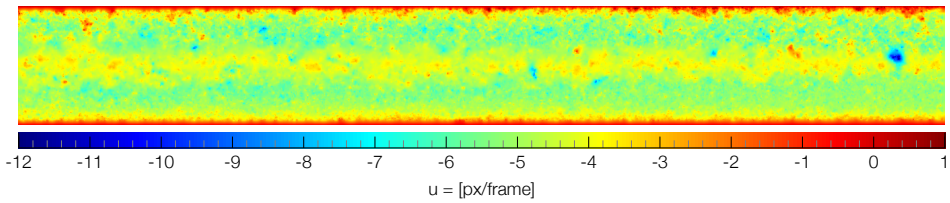


Figure 4.22: u velocity field from an image pair of series N325. No-slip boundary condition, $\alpha = 0.01$.

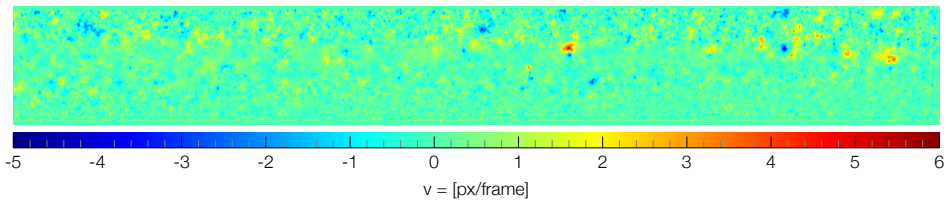


Figure 4.23: v velocity field from an image pair of series N325. No-slip boundary condition, $\alpha = 0.01$.

The fields are noisy, but do not exhibit edge errors, so this α value is satisfactory. For comparison, the u velocity field from the same image pair with no boundary condition is shown in Figure 4.24. Unrealistically large velocities on the right of the velocity field can be seen.

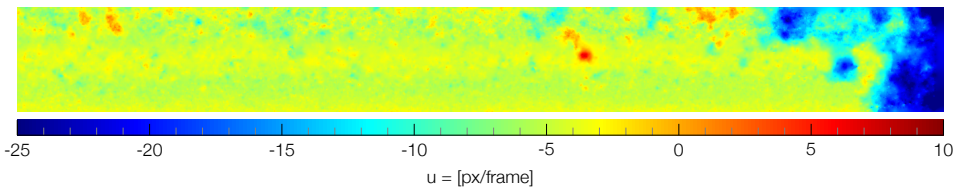


Figure 4.24: u velocity field from an image pair of series N325. No boundary condition, $\alpha = 0.01$.

The series of 1024 images are processed with the selected α. In post processing, the mean velocity (Figures 4.25 and 4.26), and velocity RMS fields (Figures 4.27 and 4.28) are calculated.

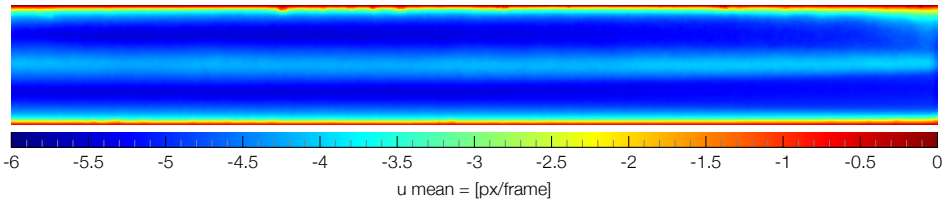


Figure 4.25: Mean u velocity field for series N325 image pairs averaged over 1024 frames.

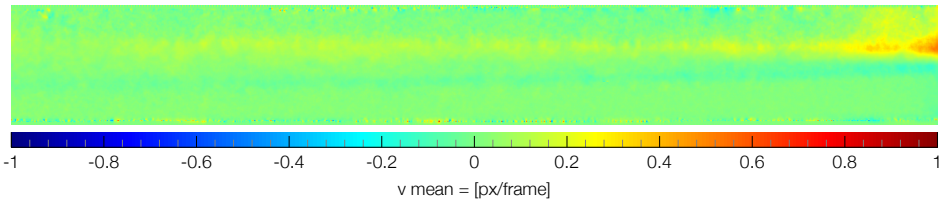


Figure 4.26: Mean v velocity field for series N325 image pairs averaged over 1024 frames.

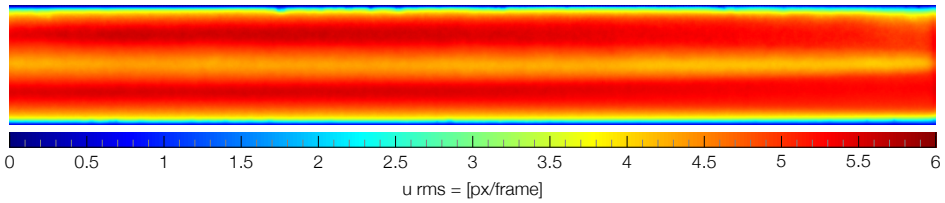


Figure 4.27: u RMS for series N325 image pairs averaged over 1024 frames.

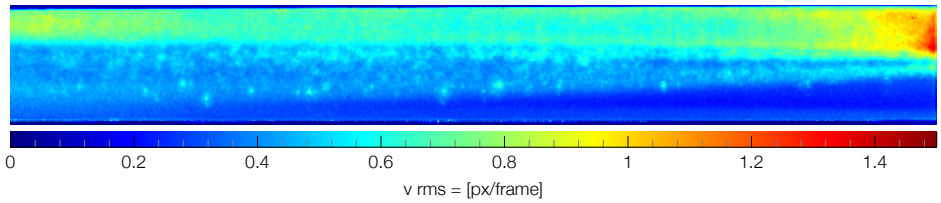


Figure 4.28: v RMS for series N325 image pairs averaged over 1024 frames.

The smoothness of the series averaged results is impressive. The noise present in the analysis of single pairs is completely eliminated, and there are only small anomalies in the velocity field near the channel edges. The over-smoothing visible at the left and right edge of the u mean and RMS plots is due to image replacement in this region. The regularizer fills in this area with an overly smooth velocity field.

As a final check, the condition from Equation 4.3 is calculated. Since $2.34 > 0.66$, the condition is satisfied and it can be reasonably concluded that with this α value, it was possible to capture the gradients present in the flow. For a more robust analysis, the use of a more refined metric to quantify the confidence interval is encouraged, for example, checking that the two values are larger than a couple standard deviations apart.

Additional Applications

A feature of our optical flow PIV algorithm, besides calculating dense velocity fields, is the ability to analyze blurry, or otherwise continuous images. Thus, the algorithm should be equally capable of extracting velocity information from LIF images as traditional PIV images. And indeed, that is the case. In fact, where dye is present, there is more gradient information, and calculated flow fields appear smoother and better determined than those calculated from PIV images, as seen in Figure 4.30, calculated from the left image Figure 4.29.

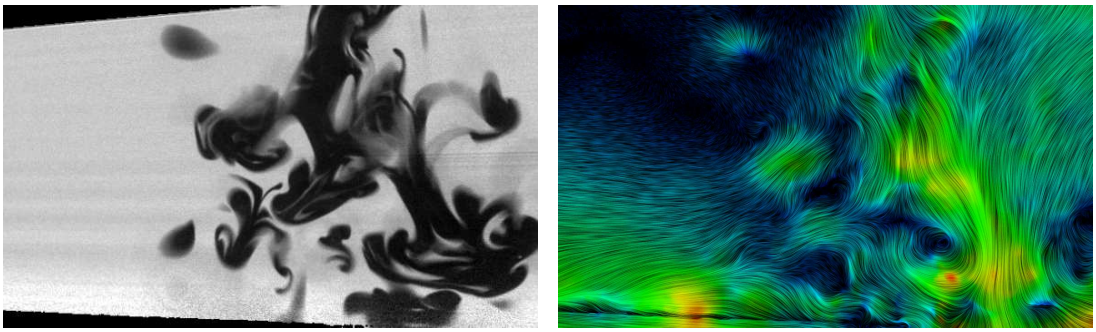


Figure 4.29: LIF image from the FLUID database (Snarf-1) (left) together with a visualization of the velocity field calculated with optical flow (right).

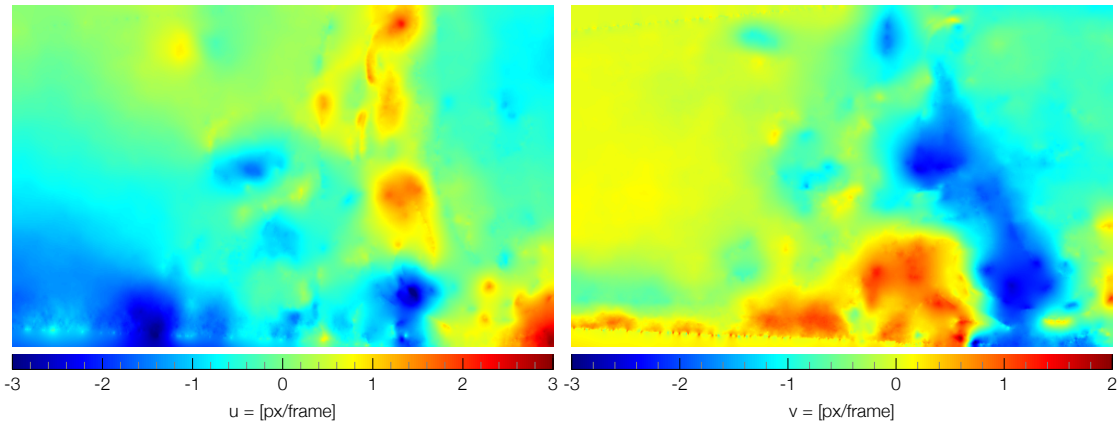


Figure 4.30: u (left) and v (right) components of the calculated velocity field, $\alpha = 0.1$.

Unfortunately, optical flow LIF will not replace PIV any time soon, due to the method's inability to resolve motion perpendicular to the dye gradient, and where there is uniform pigment concentration. For example, in a region saturated with dye, there could be a vortex, and since it would not produce intensity variations, the structure would not be present in the velocity field. In this regard, the discrete particles of PIV are superior. That being said, coupling optical flow and LIF would allow a single experimental setup to capture both concentration and velocity information, without the need for extra cameras, filters, and alignment.

Similarly, Dupont et al. (2013) have developed an optical liquid film sensor based on near infrared absorption in water (NIR). Basically, the thicker a water film is, the more near-infrared light is absorbed, and the darker it appears. The technique uses a high-speed infrared camera to record the evolution of the fluid film over time. An example image is shown in Figure 4.31.

Figure 4.31: NIR image of water film flowing down a plate, 300×80 px.

Coupled with our method, simultaneous measurements of film thickness and phase velocity (Figure 4.32) is possible without modification to the experimental setup. It should be noted however, that NIR images have large amounts of occlusion, different wavelengths travelling at different speeds which may be different than the average phase velocity. Thus, optical flow methods may not be well suited to calculating mass flow, or velocity spectra.

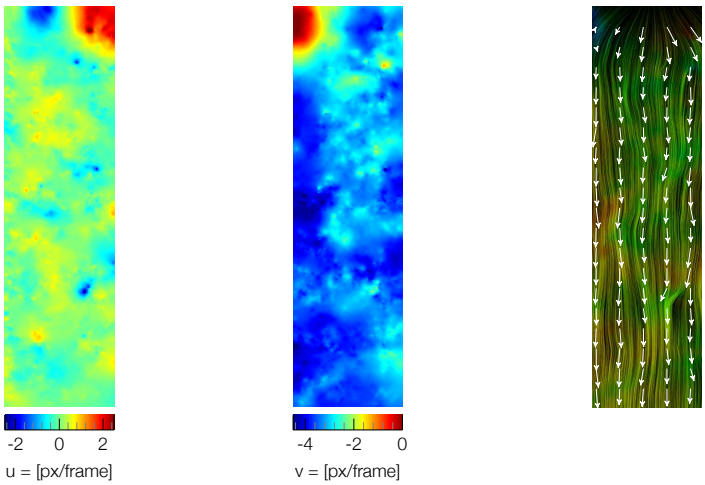


Figure 4.32: u (left) and v (middle) velocity fields calculated from a NIR image set. A visualization of the entire flow field is shown on the right.

Conclusion

In this work, five techniques to improve the accuracy of optical flow PIV are proposed:

Symmetric spline warping: a simple and effective way to improve velocity estimate accuracy compared to traditional bicubic backward mapping.

Scalar advection warping: a promising new method which uses a CFD technique to produce better velocity estimates for small displacements.

Spatially varying α: a way to lower the smoothness in regions of high velocity gradient.

Warp RMS sweep: currently the best method for choose the smoothing parameter. Optimum smoothness is considered to be at the lowest warp RMS.

Data-driven Boundary Conditions: effective at resolving no-slip boundary layers, and improves algorithm stability with noisy experimental images.

All but the warp RMS sweep have been motivated by the physical nature of the continuous, non-rigid velocity fields of fluid flow. This is in contrast to traditional optical flow research, which focuses on resolving discontinuous rigid body motion. Given the unique requirements of optical flow PIV, some of the underlying assumptions were revisited. In return, parts of the problem took on physical significance and it was possible to incorporate fluid dynamics analogs in the optical flow algorithm.

The ability of each of the proposed methods has been demonstrated on a set of synthetic images, to objectively measure the accuracy against the true velocity field. In all cases, symmetric spline warping produces results superior to traditional bicubic methods, and is recommended as the default warping technique. For images with small displacements, or vortical structures, the newly developed advection scheme performs even better than spline warping. Its physical formulation lends itself to analyzing fluid flows, and shows great future promise in optical flow PIV.

Results obtained with a spatially varying α are inconclusive, even after exhaustive modifications of the distribution method and scaling parameters. While the method has conceptual merit, it appears that problems resolving the boundary layer of channel flow are not strongly linked with over-smoothing. This conclusion is based on the use of data-driven boundary conditions, which are very effective in resolving the correct velocity field without needing to change α.

To objectively select an appropriate α for experimental images, an intensity RMS minimization technique previously used with TV-L1 optical flow analysis of video sequences is used. The optimal α is determined brute-force by finding the minimum warp RMS for a series of smoothness values.

Bringing together the concepts developed in this work, symmetric spline warping, no-slip boundary conditions, and warp RMS minimization are used together with intensity normalization to analyze an experimental PIV image series. The particular data set was chosen for its high velocity, large amounts of mixing, and heavy refractive distortion to demonstrate the robustness of the algorithm under the most unfavourable conditions. The calculated flow field is not only reasonable, but is of a resolution unmatched by traditional cross-correlation methods.

To demonstrate the flexibility of optical flow PIV, it was applied unmodified to experimental LIF and NIR images, with very encouraging results.

Future Work

To further increase the accuracy of optical flow PIV, the following improvements can be made:

Higher-order Advection Scheme

The last remaining challenge in replacing inverse mapping with 2D advection is the accumulated phase errors over multiple time steps. By combining:

1. Time extended stencils (Kim, 2003)
2. 2D stencils on regular grids (Kim, 1997)
3. Phase-error cancelling (Fromm, 1968)

It should be possible to significantly decrease both advection dispersion and dissipation. As a proof of concept, a fourth-order upwind leapfrog scheme from Kim (2003) initialized with two second-order Fromm time steps has been implemented with excellent results. But, as shown in Kim (1997), advection diagonal to the grid introduces severe phase errors which need to be addressed before the technique is of practical use. Alternatively, spectral-based advection methods could be used for ultra-low phase error advection.

Accurate Image Replacement

The current image-replacement implementation does not properly account for fractional pixel motion, or curvilinear motion, which causes errors on the interface where image replacement occurs. These errors are propagated towards the edge by the smoothness term. A more sophisticated replacement method, i.e. marker advection with a strict flux limiter should help eliminate this problem.

Speedup

As the OFN code base reaches maturity and undergoes fewer changes, it would be advantageous to implement it in a fast, non-commercial language such as C/C++. The code would lose the flexibility of being implemented in an interpreted language with a large mathematical toolbox, but would be easier to use on multiple computers, and much faster to run. By using open-source code such as OpenCV, or that provided by Meinhardt-Llopis and Sánchez (2012), the translation effort would be greatly reduced. If C++ was chosen as the new language, the GPU implementation summarized in Appendix A could be used directly.

Regularization

To be able to resolve increasingly complex flows, it would be beneficial to replace the regularizer with a more physical div-curl or second-order div-curl. However, these methods introduce a second weighting parameter which must be chosen appropriately.

References

Literature

Atcheson, B., Heidrich, W., & Ihrke, I. (2009). An evaluation of optical flow algorithms for background oriented schlieren imaging. *Experiments in Fluids*, 46 (3), 467–476.

Baker, S., Scharstein, D., Lewis, J., Roth, S., Black, M., & Szeliski, R. (2011). A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92 (1), 1–31.

Barron, J. L., Fleet, D. J., & Beauchemin, S. S. (1994). Performance of optical flow techniques. *International Journal of Computer Vision*, 12 (1), 43–77.

Brox, T., Bruhn, A., Papenberg, N., & Weickert, J. (2004). High accuracy optical flow estimation based on a theory for warping. In *Proceedings of the European Conference on Computer Vision* (Vol. 3024, pp. 25–36). Prague, Czech Republic: Springer.

Coons, S. A. (1967). *Surfaces for computer-aided design of space forms* (Tech. Rep.). Cambridge, MA, USA.

Corpetti, T., Heitz, D., Arroyo, G., Mémin, E., & Santa-Cruz, A. (2006). Fluid experimental flow estimation based on an optical-flow scheme. *Experiments in Fluids*, 40 (1), 80–97. (10.1007/s00348-005-0048-y)

Corpetti, T., Memin, E., & Perez, P. (2002). Dense estimation of fluid flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24 (3), 365–380.

Courant, R., Friedrichs, K., & Lewy, H. (1928). Über die partiellen differenzengleichungen der mathematischen physik. *Mathematische Annalen*, 100 (1), 32-74.

Dérian, P., Héas, P., Herzet, C., & Memin, E. (2013). Wavelets and optical flow motion estimation. *Numerical Mathematics: Theory, Methods and Applications*, 6 (1), 116–137.

Dierckx, P. (1993). *Curve and surface fitting with splines*. Oxford University Press, Inc.

Doshi, A., & Bors, A. G. (2007). Navier-stokes formulation for modelling turbulent optical flow. In *Proceedings of the British Machine Vision Conference* (pp. 97.1–97.10). BMVA Press. (doi:10.5244/C.21.97)

Dupont, J., Mignot, G., & Prasser, H.-M. (2013). Near infrared liquid water film thickness measurement technique and 2-D mapping (submitted). In *Proceedings of the International Topical Meeting on Nuclear Reactor Thermalhydraulics*.

Enkelmann, W. (1988). Investigations of multigrid algorithms for the estimation of optical flow fields in image sequences. *Computer Vision, Graphics, and Image Processing*, 43 (2), 150–177.

Fitzpatrick, J. M. (1988). The existence of geometrical density-image transformations corresponding to object motion. *Computer Vision, Graphics, and Image Processing*, 44 (2), 155 – 174.

Fromm, J. E. (1968). A method for reducing dispersion in convective difference schemes. *Journal of Computational Physics*, 3 (2), 176 – 189.

Gupta, S., Gupta, E. N., & Prince, J. L. (1996). On div-curl regularization for motion estimation in 3-D volumetric imaging. In *Proceedings of the IEEE International Conference on Image Processing* (pp. 929–932).

Hackbusch, W. (1994). *Iterative solution of large sparse systems of equations*. Springer-Verlag.

Horn, B. K., & Schunck, B. G. (1981). Determining optical flow. *Artificial Intelligence*, 17 (1), 185 – 203.

Kähler, C., Scharnowski, S., & Cierpka, C. (2012). On the uncertainty of digital PIV and PTV near walls. *Experiments in Fluids*, 52 (6), 1641–1656.

Kapulla, R., Hoang, P., Szijarto, R., & Fokken, J. (2011). Parameter sensitivity of optical flow applied to PIV images. In *Proceedings of the Lasermethoden in der Strömungsmeßtechnik*. Ilmenau.

Kim, C. (1997). *Multi-dimensional upwind leapfrog schemes and their applications*. Unpublished doctoral dissertation, The University of Michigan.

Kim, C. (2003). Accurate multi-level schemes for advection. *International Journal for Numerical Methods in Fluids*, 41 (5), 471–494.

Krajsek, K., & Mester, R. (2007). Bayesian model selection for optical flow estimation. In *Proceedings of the Conference on Pattern Recognition* (Vol. 4713, pp. 142–151). Springer Berlin Heidelberg.

Lecordier, B., & Westerweel, J. (2004). The EUROPIV synthetic image generator (S.I.G.). In *Particle Image Velocimetry: Recent Improvements* (pp. 145–161). Springer Berlin Heidelberg.

Meinhardt-Llopis, E., & Sánchez, J. (2012). Horn-Schunck optical flow with a multi-scale strategy. *Image Processing On Line*, (pre-print), 1–18.

Morse, P. M., & Feshbach, H. (1953). *Methods of theoretical physics*. McGraw-Hill.

Nagel, H.-H., & Enkelmann, W. (1986). An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8 (5), 565–593.

Ng, L., & Solo, V. (1997). A data-driven method for choosing smoothing parameters in optical flow problems. In *Proceedings of the International Conference on Image Processing* (Vol. 3, pp. 360–363).

Quénot, G. M., Pakleza, J., & Kowalewski, T. A. (1998). Particle image velocimetry with optical flow. *Experiments in Fluids*, 25 (3), 177–189.

Raffel, M., Willert, C. E., Wereley, S. T., & Kompenhans, J. (2007). *Particle image velocimetry: A practical guide*. Springer Berlin Heidelberg.

Ruhnau, P. (2006). *Variational fluid motion estimation with physical priors*. Unpublished doctoral dissertation, Universität Mannheim.

Ruhnau, P., Kohlberger, T., Schnörr, C., & Nobach, H. (2005). Variational optical flow estimation for particle image velocimetry. *Experiments in Fluids*, 38 (1), 21–32.

Sánchez, J., Meinhardt-Llopis, E., & Facciolo, G. (2012). TV-L1 optical flow estimation. *Image Processing On Line*, (pre-print), 1–13.

Scarano, F. (2002). Iterative image deformation methods in PIV. *Measurement Science and Technology*, 13 (1), R1–R19.

Scarano, F., & Riethmuller, M. L. (2000). Advances in iterative multigrid PIV image processing. *Experiments in Fluids (Supplement)*, 29 (1-supplement), S051–S060.

Stanislas, M., Okamoto, K., & Kähler, C. (2003). Main results of the first international PIV challenge. *Measurement Science and Technology*, 14 (10), 63-89.

Su, L. K., & Dahm, W. J. A. (1996a). Scalar imaging velocimetry measurements of the velocity gradient tensor field in turbulent flows. I. Assessment of errors. *Physics of Fluids*, 8 (7), 1869–1882.

Su, L. K., & Dahm, W. J. A. (1996b). Scalar imaging velocimetry measurements of the velocity gradient tensor field in turbulent flows. II. Experimental results. *Physics of Fluids*, 8 (7), 1883–1906.

Sun, D., Roth, S., & Black, M. (2010). Secrets of optical flow estimation and their principles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2432–2439).

Suter, D. (1994). Motion estimation and vector splines. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (pp. 939–942).

Tu, Z., Xie, W., Hürst, W., Xiong, S., & Qin, Q. (2012). Weighted root mean square approach to select the optimal smoothness parameter of the variational optical flow algorithms. *Optical Engineering*, 51 (3), 037202-1–037202-9.

Weickert, J., & Schnörr, C. (2001a). A theoretical framework for convex regularizers in PDE-based computation of image motion. *International Journal of Computer Vision*, 45 (3), 245–264.

Weickert, J., & Schnörr, C. (2001b). Variational optic flow computation with a spatio-temporal smoothness constraint. *Journal of Mathematical Imaging and Vision*, 14 (3), 245–255.

Westerweel, J. (1993). *Digital particle image velocimetry—theory and application*. Unpublished doctoral dissertation, Delft University of Technology, Delft.

Wildes, R. P., Amabile, M. J., Lanzillotto, A.-M., & Leu, T.-S. (2000). Recovering estimates of fluid flow from image sequence data. *Computer Vision and Image Understanding*, 80 (2), 246–266.

Wolberg, G. (1994). *Digital image warping* (1st ed.). Los Alamitos, CA, USA: IEEE Computer Society Press.

Zimmer, H., Bruhn, A., & Weickert, J. (2011). Optic flow in harmony. *International Journal of Computer Vision*, 93 (3), 368–388.

Online

Bell, N., & Garland, M. (2013). *Cusp: Generic parallel algorithms for sparse matrix and graph computations*. Retrieved from `http://cusp-library.googlecode.com`

Carlier, J. (2005). *Second set of fluid mechanics image sequences*. European Project 'Fluid image analysis and description' (FLUID). Retrieved from `http://www.fluid.irisa.fr/`

Jones, E., Oliphant, T., Peterson, P., et al. (2001–2013). *SciPy: Open source scientific tools for Python*. Retrieved from `http://www.scipy.org/`

OpenCV. (2013). Retrieved from `http://opencv.org/`

Code Speedup

Although written in the interpreted `Matlab` language, the code for OFN is extremely optimized, fully exploiting vectorization to increase speed as much as possible. Just the same, when the code was analyzed with the built in `profiler`, two functions were found to be responsible for over 50% of the computation time, the LES builder and the built-in `bicgstab` biconjugate gradient stabilized LES solver.

A long-standing project goal had been to speed up these highly parallel, computationally expensive operations with the on-board graphics processing unit (GPU). The sparse, banded nature of the LES increased the possible speedup margin over a dense matrix, as efficient sparse solvers could be used. After searching for possible commercial solutions, it was decided to program the `Matlab` – GPU solver interface in-house using the open source `cusp` library (Bell & Garland, 2013) for the following reasons:

Flexibility Different solvers, pre-conditioners, and container formats are included, making optimization very convenient.

Maintainability Bugs and support for new cards is handled by the community.

Future-proof Code is open source.

Reusability If OFN is ever translated to C++, the function can be dropped in directly.

Cost Free.

Speed Optimized for sparse matrix calculations, and kernels are optimized to the current hardware setup. It would be very difficult to program faster kernels than those supplied.

Integrating `cusp` and `Matlab` produces the following call hierarchy:

```
Matlab > MEX > C++ > cusp > CUDA > GPU
```

The banded LES is built as a dense matrix in the `DIA`[1] format in `Matlab`. This is faster than building the equivalent `CSR`[2] formatted matrix, has less overhead when passed to the GPU since no coordinates need to be included, and does not need to be converted when it arrives. Once on the GPU, the system is solved using a `diagonal` pre-conditioner, together with the `krylov::bicgstab` solver. The solution is passed back to `Matlab` as a vector, incurring very little bandwidth penalty.

[1] DIAagonal sparse matrix

[2] Compressed Sparse Row

To give an idea of the speedups possible, timing results obtained on the current hardware setup are presented:

System	2× 4 core Intel Xenon E5620 processors, 2.40 GHz 24 GB DDR3 RAM
GPU	NVIDIA GeForce GTX 560 Ti 1GB RAM 128 GB/s bus speed 384 cores 2.1 compute capability
Software	Windows 7, 64 bit Matlab 2010b CUDA 5.0 cusp 0.3.1

Results are presented for input image sizes ranging from 128, 256, 512, to 1024×1024 px. The details of the corresponding system matrices are summarized in Table A.1.

Table A.1: LES characteristics for different image sizes.

Image Size [px]	LES Dimensions [elements]	Number of non-zero elements	Memory footprint (double) [MB]
64×64	8 192 × 8 192	79 888	0.6
128×128	32 768 × 32 768	323 600	2.5
256×256	131 072 × 131 072	1 302 544	9.9
512×512	524 288 × 524 288	5 226 512	39.9
1024×1024	2 097 152 × 2 097 152	20 938 768	159.8

The speedup of the GPU over the `Matlab` LES construction (build) and solving (solve) implementations for increasing image sizes are summarized in Table A.2. Time breakdowns are presented in Figures A.1 to A.3. Note that for the GPU implementation, the LES is still built using `Matlab`, the only difference is the `DIA` format is used instead of `Matlab`'s default `CSR`. GPU timing results have been averaged over 50 runs to smooth anomalies.

Table A.2: Speedups of GPU vs. the `Matlab` implementations.

Image size	Build	Solve	Build + Solve
64×64	30×	18×	21×
128×128	57×	39×	44×
256×256	24×	100×	71×
512×512	16×	174×	82×
1024×1024	14×	154×	75×

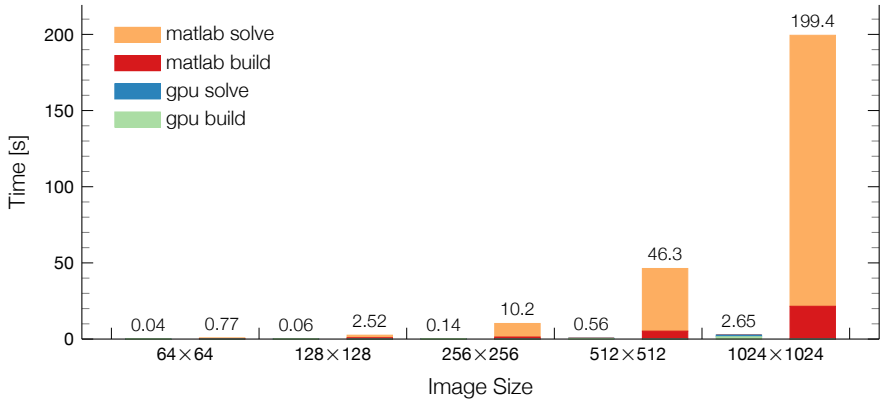


Figure A.1: LES building and solving with `Matlab` vs. GPU (lower is better).

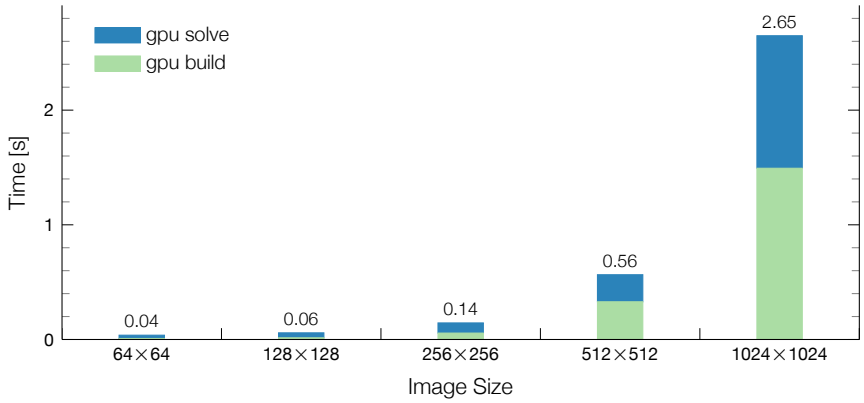


Figure A.2: LES building and solving on the GPU (lower is better).

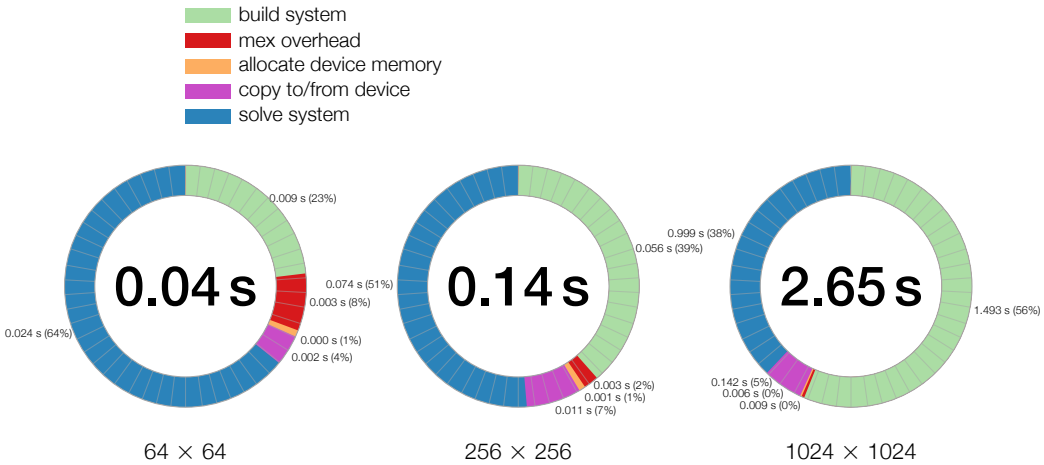


Figure A.3: GPU timing breakdown of LES building and solving.

Derivation of the Optical Flow Euler-Lagrange Equations

To bridge the gap between the total error equation and the final minimization of Horn and Schunck (1981), the detailed derivation is provided here.

$$L = \int_{\Omega} \left[\mathcal{E}_{dat}^2 + \alpha^2 \mathcal{E}_{reg}^2 \right] dx\, dy \tag{B.1}$$

substituting terms,

$$L = \int_{\Omega} \left[(I_x u + I_y v + I_t)^2 + \alpha^2 \left(|\nabla u|^2 + |\nabla v|^2 \right) \right] dx\, dy \tag{B.2}$$

expanding,

$$\begin{aligned} L = \int_{\Omega} \Bigg[& \left(I_x^2 u^2 + I_y^2 v^2 + I_t^2 + 2 I_x I_y u v + 2 I_y I_t v + 2 I_x I_t u \right)^2 \\ & + \alpha^2 \left(\left(\frac{\partial u}{\partial x} \right)^2 + \left(\frac{\partial u}{\partial y} \right)^2 + \left(\frac{\partial v}{\partial x} \right)^2 + \left(\frac{\partial v}{\partial y} \right)^2 \right) \Bigg] dx\, dy \end{aligned} \tag{B.3}$$

To minimize Equation B.3, the calculus of variations is employed to take the derivative of the integral. For simplicity, Equation B.4 is used, the general Euler-Lagrange formula for r functionals of s variables from Morse and Feshbach (1953).

$$\frac{\partial L}{\partial \psi_r} - \sum_{s=1}^{m} \frac{\partial}{\partial x_s} \left(\frac{\partial L}{\partial \psi_{rs}} \right) = 0 \tag{B.4}$$

In this case $r = 2$ and $s = 2$, so the formula yields two expressions,

$$\frac{\partial L}{\partial \psi_1} - \frac{\partial}{\partial x_1} \left(\frac{\partial L}{\partial \psi_{11}} \right) - \frac{\partial}{\partial x_2} \left(\frac{\partial L}{\partial \psi_{12}} \right) = 0$$

$$\frac{\partial L}{\partial \psi_2} - \frac{\partial}{\partial x_1} \left(\frac{\partial L}{\partial \psi_{21}} \right) - \frac{\partial}{\partial x_2} \left(\frac{\partial L}{\partial \psi_{22}} \right) = 0$$

converting variables,

$$\frac{\partial L}{\partial u} - \frac{\partial}{\partial x}\left(\frac{\partial L}{\partial u_x}\right) - \frac{\partial}{\partial y}\left(\frac{\partial L}{\partial u_y}\right) = 0$$

$$\frac{\partial L}{\partial v} - \frac{\partial}{\partial x}\left(\frac{\partial L}{\partial v_x}\right) - \frac{\partial}{\partial y}\left(\frac{\partial L}{\partial v_y}\right) = 0$$

the components being,

$$\frac{\partial L}{\partial u} = 2I_x^2 u + 2I_x I_y v + 2I_x I_t$$

$$\frac{\partial L}{\partial u_x} = 2\alpha^2 \frac{\partial u}{\partial x}; \quad \frac{\partial}{\partial x}\left(\frac{\partial L}{\partial u_x}\right) = 2\alpha^2 \frac{\partial^2 u}{\partial x^2}$$

$$\frac{\partial L}{\partial u_y} = 2\alpha^2 \frac{\partial u}{\partial y}; \quad \frac{\partial}{\partial y}\left(\frac{\partial L}{\partial u_y}\right) = 2\alpha^2 \frac{\partial^2 u}{\partial y^2}$$

$$\frac{\partial L}{\partial v} = 2I_y^2 v + 2I_x I_y u + 2I_y I_t$$

$$\frac{\partial L}{\partial v_x} = 2\alpha^2 \frac{\partial v}{\partial x}; \quad \frac{\partial}{\partial x}\left(\frac{\partial L}{\partial v_x}\right) = 2\alpha^2 \frac{\partial^2 v}{\partial x^2}$$

$$\frac{\partial L}{\partial v_y} = 2\alpha^2 \frac{\partial v}{\partial y}; \quad \frac{\partial}{\partial y}\left(\frac{\partial L}{\partial v_y}\right) = 2\alpha^2 \frac{\partial^2 v}{\partial y^2}$$

substituting,

$$I_x^2 u + I_x I_y v + I_x I_t - \alpha^2\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = 0$$

$$I_x I_y u + \quad I_y^2 v + I_y I_t - \alpha^2\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) = 0$$

finally, rearranging and simplifying.

$$I_x^2 u + I_x I_y v = \alpha^2 \Delta u - I_x I_t$$

$$I_x I_y u + \quad I_y^2 v = \alpha^2 \Delta v - I_y I_t$$

Zero-Phase Error Advection

The expansion of the 2D advection equation (Equation 4 from Fromm (1968)) used in our advection warping scheme is shown in detail. The computational stencil is illustrated in Figure C.1, along with the directional shorthand in Figure C.2.

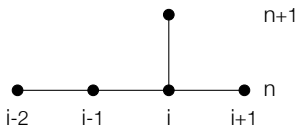


Figure C.1: Computational stencil for $a > 0$, "E" in the figure below.

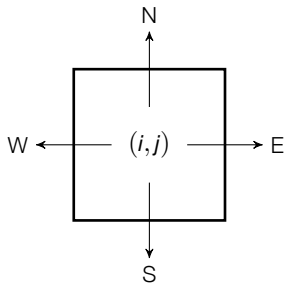


Figure C.2: Directional shorthand for computational cell i, j.

$$a = \frac{u\Delta t}{\Delta x} \tag{C.1}$$

$$b = \frac{v\Delta t}{\Delta x} \tag{C.2}$$

$$\phi_{i,j}^{n+1} = \begin{cases} E_{i,j}^{n+1} + N_{i,j}^{n+1} & \text{if } a_{i,j} > 0, b_{i,j} > 0 \\ W_{i,j}^{n+1} + N_{i,j}^{n+1} & \text{if } a_{i,j} < 0, b_{i,j} > 0 \\ W_{i,j}^{n+1} + S_{i,j}^{n+1} & \text{if } a_{i,j} < 0, b_{i,j} < 0 \\ E_{i,j}^{n+1} + S_{i,j}^{n+1} & \text{if } a_{i,j} > 0, b_{i,j} < 0 \end{cases} \tag{C.3}$$

$$\begin{aligned} E_{i,j}^{n+1} = \phi_{i,j}^{n} &+ \frac{a_{i,j}^{n}}{4}(\phi_{i-1,j}^{n} - \phi_{i+1,j}^{n} + \phi_{i-2,j}^{n} - \phi_{i,j}^{n}) \\ &+ \frac{(a_{i,j}^{n})^2}{4}(\phi_{i-1,j}^{n} - 2\phi_{i,j}^{n} + \phi_{i+1,j}^{n}) \\ &+ \frac{(a_{i,j}^{n})^2 - 2a_{i,j}^{n}}{4}(\phi_{i-2,j}^{n} - 2\phi_{i-1,j}^{n} + \phi_{i,j}^{n}) \end{aligned} \tag{C.4}$$

$$\begin{aligned} W_{i,j}^{n+1} = \phi_{i,j}^{n} &+ \frac{a_{i,j}^{n}}{4}(\phi_{i-1,j}^{n} - \phi_{i+1,j}^{n} + \phi_{i,j}^{n} - \phi_{i+2,j}^{n}) \\ &+ \frac{(a_{i,j}^{n})^2}{4}(\phi_{i-1,j}^{n} - 2\phi_{i,j}^{n} + \phi_{i+1,j}^{n}) \\ &+ \frac{(a_{i,j}^{n})^2 - 2a_{i,j}^{n}}{4}(\phi_{i,j}^{n} - 2\phi_{i+1,j}^{n} + \phi_{i+2,j}^{n}) \end{aligned} \tag{C.5}$$

$$\begin{aligned} N\phi_{i,j}^{n+1} = \phi_{i,j}^{n} &+ \frac{b_{i,j}^{n}}{4}(\phi_{i,j-1}^{n} - \phi_{i,j+1}^{n} + \phi_{i,j-2}^{n} - \phi_{i,j}^{n}) \\ &+ \frac{(b_{i,j}^{n})^2}{4}(\phi_{i,j-1}^{n} - 2\phi_{i,j}^{n} + \phi_{i,j+1}^{n}) \\ &+ \frac{(b_{i,j}^{n})^2 - 2b_{i,j}^{n}}{4}(\phi_{i,j-2}^{n} - 2\phi_{i,j-1}^{n} + \phi_{i,j}^{n}) \end{aligned} \tag{C.6}$$

$$\begin{aligned} S_{i,j}^{n+1} = \phi_{i,j}^{n} &+ \frac{b_{i,j}^{n}}{4}(\phi_{i,j-1}^{n} - \phi_{i,j+1}^{n} + \phi_{i,j}^{n} - \phi_{i,j+2}^{n}) \\ &+ \frac{(b_{i,j}^{n})^2}{4}(\phi_{i,j-1}^{n} - 2\phi_{i,j}^{n} + \phi_{i,j+1}^{n}) \\ &+ \frac{(b_{i,j}^{n})^2 - 2b_{i,j}^{n}}{4}(\phi_{i,j}^{n} - 2\phi_{i,j+1}^{n} + \phi_{i,j+2}^{n}) \end{aligned} \tag{C.7}$$