2009

High Accuracy Optical Flow Method Based on a Theory for Warping: 3D Extension

Weixin Chen

High Accuracy Optical Flow Method Based on a Theory for Warping: 3D Extension

(Spine title: High Accuracy 3D Optical Flow)

(Thesis format: Monograph)

by

Weixin Chen

Graduate Program in Computer Science

Submitted in partial fulfillment
of the requirements for the degree of
Master of Science

School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario
August, 2009

Abstract

This thesis presents the implementation and the qualitative and quantitative evaluation of a 3D optical flow algorithm, whose derivation is based on the 2D optical flow method published in the European Conference on Computer Vision (ECCV 2004) by Thomas Brox, Andrés Bruhn, Nils Papenburg and Joachim Wieickert. The optical flow minimizes an energy function built with three assumptions: a brightness constancy assumption, a gradient constancy assumption, and a smoothness assumption. They minimize this functional using a robust estimator, which make the functional convex and gaurantees convergence to a single solution. They propose a numerical scheme based on nested fixed points iterations and shows that this scheme implements a coarse-to-fine warping strategy within a 2D hierarchical image pyramid. In our 3D extension, our solution requires the regularization of a 3D functional based on 3D extensions of their assumptions in a 3D hierarchical volume pyramid. We solve the corresponding Euler-Lagrange equations iteratively using nested iterations that uses Cramer's rule, as suggested by Faisal and Barron (ICIAR 2007) rather than Brox et al.'s SOR calculation. We present 3D quantitative results on three sets of 3D sinusoidal data (with and without 3D intensity discontinuities). We also presented qualitative evaluation on a gated MRI cardiac dataset.

Keywords: 2D and 3D optical flow, regularization, brightness constancy assumption,

gradient constancy assumption, Horn and Schunck smoothness assumption, hierarchical image pyramid, hierarchical volume pyramid, warping, Euler-Lagrange equations, quantitative and qualitative error analysis.

Acknowledgements

I would like to thank my supervisor, Dr. John Barron, for his enormous assistance and patience in the completion of my thesis. He devoted countless hours of help towards solving my problems.

I would also like to thank my parents and friends for their love and support. Many thanks to my aunt and uncle who provided much care for me during my grduate studies. Last, but not the least, I would like to thank all other professors who have instructed me during my graduate studies at Western and my friends for moral support.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

This chapter introduces the basic concept of optical flow, outlines the topic for the thesis and summarizes contributions of the thesis. At the end of this chapter, the outline of the coming chapters is provided.

Optical flow is an important research area in image processing and computer vision. It is the distribution of apparent velocities of the movement of brightness patterns in an image sequence [14]. Optical flow shows the relative motions of moving objects and the observer (usually a camera). Therefore, it gives information about the motion of each pixel in an image. In other word, in case of 2D images, it specifies how much each pixel moves between adjacent frames; in case of 3D images, it specifies how much each voxel moves between adjacent volumes in the dataset. Some of its uses include motion estimation, the recovery of observer 3D motion parameters and scene depth maps object tracking and image registration. Optical flow is also referred to as **image velocity**. Figure 2.1 in the next chapter show one image of the Yosemite fly-through sequence plus its correct flow field.

Starting from the pioneering work of Horn and Schunck (1981) [14] and Lucas and Kanade (1981) [18], many methods have been proposed to estimate optical flow.

However, accurate optical flow estimation remains one of the open research areas in Computer Vision. At ECCV conference in 2004, Brox, Bruhn, Papenberg and Weickert [7] presented a variational model for computing optical flow that integrates three assumptions: a brightness constancy assumption, a gradient constancy assumption, and a smoothness constraint. They employed an image warping technique and showed that warping is a theoretically reasonable way to handle large displacements in a multi-resolution scheme. Their experimental results showed that their algorithm gave the "best" published results at that time (for Yosemite image sequence). Later, Papenberg, Bruhn, Brox, Didas and Weickert [23] extended this method and investigated more non-linearized constancy assumptions. In 2007, Faisal and Barron [9] presented two alternative ways of solving the nonlinear system of equations.

The purpose of this thesis is to design and implement a 3D extension of 2D Brox method to see if it can produce as good results on 3D data as it does for 2D images. We quantitatively evaluate the algorithm on three datasets of synthetic sinusoidal data and we also perform qualitative evaluation on two gated MRI cardiac datasets [20], a series of real volumes containing data of a human beating heart.

1.1 Contributions of Thesis

We enumerate the main thesis contributions below:

1. We propose and implement a 3D extension of a variant of Brox et al.'s algorithm [7], as described by Faisal and Barron [9] in MATLAB. Since MATLAB is optimized for matrix operations, we vectorize our programs as much as possible to gain higher efficiency. The Brox et al. algorithm is a 2-frame optical flow method; it requires only two images, but produced the best quantitative results (verified for Yosemite sequence only) as compared to other 2-frame and multi-frame algorithms (see the next chapter for an overview of some of these

algorithms). We implement our 3D algorithm as a 2-volume optical flow method to see how good the result we compute are for the two volume datasets. We also use the same differentiation method as proposed by Brox et al. (namely simple pixel differences for temporal derivatives and 4-point central differences for spatial derivatives).

2. We perform a quantitative analysis using synthetic 3D sinusoidal sequences for our 3D Brox algorithm. The synthetic sequences have both continuous and discontinuous motions and distinct intensity gradients. We know not only the correct optical flow everywhere but also the correct spatio-temporal intensity derivatives everywhere. We also implemented a hierarchical version of 3D Horn and Schunck algorithm [14] using Brox et al.'s pyramid framework in MATLAB and compared these results with those generated by our 3D Brox et al. algorithm. We also implemented a 3D version of Uras et al. local 2D algorithm [28] that uses 1^{st} and 2^{nd} spatio-temporal derivatives to test the correctness of the derivatives we generated.

3. It is interesting to observe how the coarse-to-fine warping strategy influences the accuracy of optical flow between adjacent levels. We tested this by re-sampling and re-scaling the correct velocity field appropriately for each pyramid level and computed the 3D angular error (see Chapter 4 for a full description of this and other experimental tools) for all pyramid levels. We also show the flow fields each level.

4. We test the variation of the parameters for the algorithm. These parameters includes the smoothness parameter which controls the contribution of the smoothness term to the total energy, the gradient parameter which specifies the contribution of the gradient constraint to this term, the number of pyramid levels and the size of a median filter (used to remove outliers caused by interpolation in the warping process).

5. Lastly, we provided a method for generating 3D sinusoidal data, correct 3D optical flow and correct 3D intensity derivatives for the data.

1.2 Outline of Thesis

In this chapter, we introduced optical flow, described the topics of this thesis, and enumerated the contributions of the thesis. The next chapter, Chapter 2, presents a general review of some well-known optical flow algorithms that give good experimental results for the Yosemite Fly-Through synthetic image sequence and gives an introduction to the Euler-Lagrange equations that many of these techniques use. Chapter 3 contains the details of 3D Horn and Schunck algorithm, 3D Brox et al. algorithm and 3D Uras et al. algorithm. Chapter 4 presents the techniques we use to generate 3D sinusoidal image data, the equations to compute correct spatio-temporal derivatives and the method to compute 3D angular error. Chapter 5 gives experimental results for 3D sinusoidal data sequences as produced by the Horn and Schunck algorithm and the Brox et al. algorithm, and qualitative results for gated MRI cardiac data sequences. We summarize the thesis results and outline some future work in Chapter 6.

Chapter 2

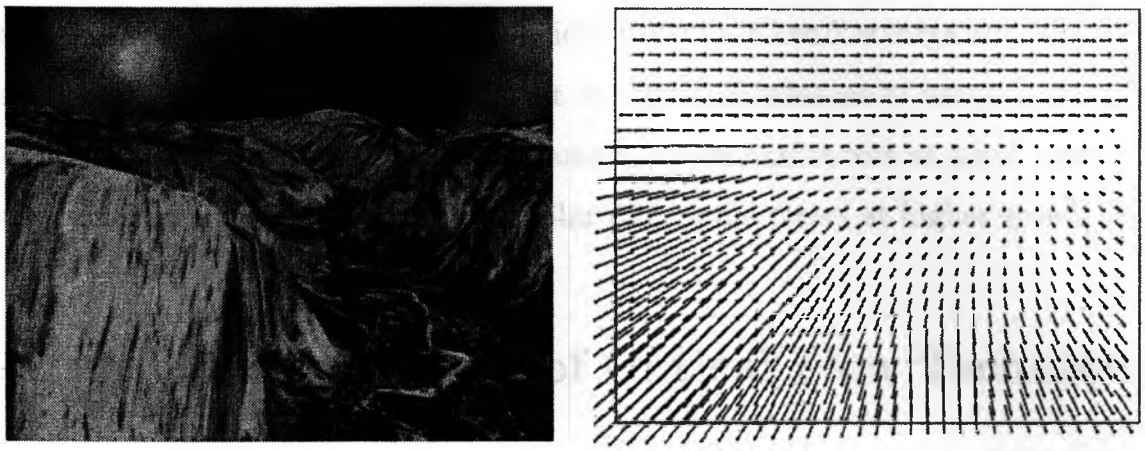
Literature Survey

This chapter provides a general review of some of the recent well known optical flow algorithms and an introduction to the Euler-Lagrange equations used to minimize the energy term used by many of the techniques.

Most of the algorithms discussed in this chapter use the famous **Yosemite Fly-Through** synthetic image sequence, made in 1988 by Lynn Quan at SRI International. These images were made by texture mapping picture of the Yosemite mountain range onto their corresponding depth maps. The clouds are fractal based and move left to right at 2 pixels per frame. This sequence is referred to just as the Yosemite sequence or the **cloudy** Yosemite sequence. Another version of this sequence is the **cloudless** Yosemite sequence, where the clouds have been removed. The reason the clouds are removed is because some researchers argue that clouds are not rigid (but rather deformable) objects and the motion constraint equation does not hold. The clouds indeed do significantly reduce the accuracy of the recovered flow. If the clouds are removed there is no flow in this area.

Since both the depth map and camera motions were known for this image sequence, the correct velocities can be computed. This allows a quantitative evaluation of optical flows generated by different algorithms. Figure 2.1 shows the middle frame

of the Yosemite Fly-Through sequence plus its correct 2D optical flow. The correct flow field is generated using the image velocity equations derived by Longuet-Higgins and Prazdny in 1980 [17].



(a) (b)

Figure 2.1: (a) The middle frame of the Yosemite Fly-Through sequence and (b) its correct flow field.

Since this thesis is concerned with the 3D extension and evaluation of the algorithm presented by Brox et al. [7] and they reported 2D quantitative analysis of the Yosemite sequence, we will restrict our survey to those optical techniques that also quantitatively analyze this sequence.

The comparison is based on an angular measure of error [6]. Image velocity (optical flow) may be written as displacement per time unit as in $\vec{v} = (u, v)$ pixels/frame. We can also write this as a space-time direction vector $(u, v, 1)$ in units of (pixel, pixel, frame). In this case, velocity is obtained from the direction vector by dividing by the third component. When velocity is viewed (and measured) as an orientation in space-time (a space-time direction), it is natural to measure errors as angular deviations from the correct space-time orientation. Therefore, we let velocities $\vec{v} = (u, v)^T$ be represented as 3D direction vectors, the normalized velocity $\hat{v}$ is defined as $\hat{v} \equiv \frac{1}{\sqrt{u^2+v^2+1}}(u, v, 1)^T$. The angular error between the correct velocity $\vec{v}_c$ and an

estimate $\vec{v}_e$ is then:

$$\psi_E = \arccos(\hat{v}_c \cdot \hat{v}_e) \ . \tag{2.1}$$

Both the direction error and magnitude error are taken into account because of normalization. This error measure is convenient because it handles large and very small speeds without the amplification inherent in a relative measure of vector differences. It does have some bias however. For example, directional errors at small speeds do not give as large an angular error as similar directional errors at higher speeds [6].

2.1 A General Review of Optical Flow Techniques

2.1.1 Barron, Fleet and Beauchemin 1994

Barron et al. [6] implemented 9 algorithms from the 1994 contemporary literature and performed quantitative evaluation of these algorithms for a common set of real and synthetic data. The algorithms examined can be classified as:

1. Spatio-temporal intensity derivative methods compute image velocity based on derivatives of image intensity or filtered version of image together with further constraints such as the global smoothness. Four instances were chosen in this paper: Horn and Schunck [14] (1^{st} order derivative with global smoothness), Lucas and Kanade [18] (1^{st} order intensity derivatives fit via a weighted least-squares fit to a constant velocity model in a local neighbourhood), Nagel [21] (2^{nd} order derivative with oriented global smoothness constraint) and Uras, Girosi, Verri and Torre [28] (1^{st} and 2^{nd} order intensity derivatives fit to constant velocity model in a local neighbourhood).

2. Region-based matching methods define velocity as the shift that yields the best fit between image regions at different times by finding the best match matches to maximizing or minimizing certain measures. The techniques considered here

are: Anandan [3] (Laplacian pyramid and coarse-to-fine sum-of-squared difference (SSD) based matching) and Singh [27] (SSD values using three adjacent band-pass filtered images and propagation of the resulting velocity using neighborhood constraints).

3. Energy-based methods use velocity-tuned filters in the Fourier domain and compute the optical flow using the energy of these filters. The method developed by Heeger [13], which was formulated as a least-squares fit of spatio-temporal energy to a plane in frequency space, is reported.

4. Phrase-based techniques define velocity in terms of phase behavior of band-pass filter outputs. The method developed by Waxman, Wu and Bergholm [29], which applies spatio-temporal filters to binary-edge maps to track edges in real-time, was considered. Fleet [11] designed a family of band-pass filters and defined component velocity according to the instantaneous motion normal to level phrase contours.

Their comparisons concentrated on the accuracy, reliability and density of the velocity measurements. They show that performance can differ significantly among the techniques they implemented.

2.1.2 Horn and Schunck 1981

Horn and Schunck's algorithm [14] is pioneering work in the optical area. It is based on two key assumptions: the grayvalue constancy assumption and the global smoothness assumption. The grayvalue constancy assumption assumes that the brightness of a pixel stays constant when it moves from one location to another. So $\frac{dI}{dt} = 0$, which can be expanded using a 1^{st} order Taylor series expansion to derive the motion constraint equation:

$$I_x u + I_y v + I_t = 0, \tag{2.2}$$

where u is the velocity component in the x direction, v is the velocity component in the y direction, and I_x, I_y and I_t are the partial derivatives of image brightness with respect to x, y and t respectively.

The smoothness assumption assumes that neighboring points on the objects have similar velocities and the velocity field of the brightness patterns in the image varies smoothly almost everywhere. One way to express the smoothness constraint is to minimize the square of the magnitude of the gradient of the optical flow velocity:

$$||\nabla u||_2^2 + ||\nabla v||_2^2. \tag{2.3}$$

The problem then becomes minimizing the total energy function:

$$\int\int \left((I_x u + I_y v + I_t)^2 + \alpha^2(||\nabla u||_2^2 + ||\nabla v||_2^2)\right) dxdy, \tag{2.4}$$

where α^2 is a weighting factor (the Lagrange multiplier) between the brightness constraint and smoothness constraint. Typical values of α (determined by trial and error) are 1.0 or 10.0. To minimize Equation 2.4, Horn and Schunck used the Euler-Lagrange equations

$$f_u - \frac{df_{u_x}}{dx} - \frac{df_{u_y}}{dy} = 0 \tag{2.5}$$

$$f_v - \frac{df_{v_x}}{dx} - \frac{df_{v_y}}{dy} = 0, \tag{2.6}$$

to derive:

$$I_x^2 u + I_x I_y v + I_x I_t = \alpha^2 \nabla^2 u \text{ and} \tag{2.7}$$

$$I_x I_y u + I_y^2 v + I_y I_t = \alpha^2 \nabla^2 v. \tag{2.8}$$

Since $\nabla^2 u \approx \bar{u} - u$ and $\nabla^2 v \approx \bar{v} - v$, then by solving u and v using Crammer's rule,

one can obtain a set of iterative equations:

$$u^{n+1} = \bar{u}^n - \frac{I_x \left[I_x \bar{u} + I_y \bar{v} + I_t \right]}{(\alpha^2 + I_x^2 + I_y^2)} \text{ and} \tag{2.9}$$

$$v^{n+1} = \bar{v}^n - \frac{I_y \left[I_x \bar{u} + I_y \bar{v} + I_t \right]}{(\alpha^2 + I_x^2 + I_y^2)}, \tag{2.10}$$

where n denotes the iteration number, u^0 and v^0 denote initial velocity estimates, which are initially set to zero, and $\bar{u}^n$ and $\bar{v}^n$ denote 5×5 neighborhood weighted averages of u^n and v^n.

2.1.3 Horn and Schunck Yosemite Results

For the Yosemite sequence, an implementation [6] of the original 2-frame Horn and Schunck produced $32.43° \pm 30.28°$ of error. When Barron et al. used Simoncelli filters to compute the intensity derivatives (now for a Gaussian standard deviation of 1.5 for the pre-smoothing filter, 15 images are required), the error was reduced to $11.26° \pm 16.41°$. Both flow fields were 100% dense. When thresholding on the spatial gradient was used, $||\nabla I|| \geq 5.0$, the angular error was reduced to $25.41° \pm 28.14°$ with 59.6% density for the original 2-frame algorithm and $5.48° \pm 10.41°$ with 32.9% density for the Simoncelli derivatives.

2.1.4 Lucas and Kanade 1981

Lucas and Kanade's algorithm [18] is also one of the pioneering works in this field. It was initially presented as an image registration technique. The basis of this method also uses the motion constraint equation.

$$I_x u + I_y v + I_t = 0, \tag{2.11}$$

where I_x, I_y and I_t and the spatio-temporal image intensity derivatives and (u, v) are the x and y components of the 2D image velocity. The second assumption of this algorithm assumes that the velocity is the same in some local neighborhood (the local constant velocity model), so two (or more) sets of derivatives can yield a non-singular linear (least squares) system of equations that yield values for (u, v). Barron et al. implemented a weighted least-square fit of local 1^{st} order constraints for (u, v) in each small spatial neighborhood Ω by minimizing:

$$\sum_{(x,y)\in\Omega} \left[w_i^2(\nabla I \cdot \mathbf{v} - I_t)^2\right], \tag{2.12}$$

where w_i is the i^{th} diagonal element of diagonal weight matrix W, whose elements are 2D Gaussian weights which gives derivative values closer to the neighborhood center more heavily weight than values further away. I_{xi}, I_{yi} and I_{ti} are derivative elements of row i in this linear system of equations. The solution is given by

$$A^T W^2 A\mathbf{v} \;=\; A^T W^2 \mathbf{b}\,, \tag{2.13}$$

where, for n points $(x, y) \in \Omega$ at a single time t,

$$\begin{aligned} A_i &= [I_{xi}\;\; I_{yi}], i \in [1, n], \\ W &= \mathrm{diag}[w_0, ..., w_i, ..., w_n], \\ \mathbf{b} &= -(I_{t1}\;\; I_{t2}\;\; ...\;\; I_{tn})^T. \end{aligned} \tag{2.14}$$

The solution to Equation 2.13 is $\mathbf{v} = [A^T W^2 A]^{-1} A^T W^2 \mathbf{b}$, which is solved in closed form when $A^T W^2 A$ is nonsingular, since it is a 2×2 matrix:

$$A^T W^2 A \;=\; \begin{bmatrix} \sum W_i^2 I_{xi}^2 & \sum W_i^2 I_{xi} I_{yi} \\ \sum W_i^2 I_{yi} I_{xi} & \sum W_i^2 I_{yi}^2 \end{bmatrix}, \tag{2.15}$$

where all sums are taken over points in the neighborhood Ω. Spatial neighborhoods Ω are 5×5 pixels.

The smallest eigenvalue of the least squares integration matrix $[A^T W^2 A]$ is used to indicate the reliability of an optical flow calculation. If the eigenvalues are λ_1 and λ_2 ($\lambda_1 \geq \lambda_2 \geq 0$), then if both λ_1 and λ_2 were greater than a threshold τ, then $\vec{v}$ is computed reliably. If $\lambda_1 \geq \tau$ but $\lambda_2 < \tau$, then a normal velocity estimate was computed reliably and if $\lambda_1 < \tau$ no velocity was computed. τ was 1.0.

2.1.5 Lucas and Kanade Yosemite Results

For the Yosemite sequence, when $\lambda_2 \geq 1.0$, Barron et al. reported an error of $4.10° \pm 9.58°$ with a density of 35.1%. When $\lambda_2 \geq 5.0$, the error was $3.05° \pm 7.31°$ with 8.7% density.

2.1.6 Nagel 1983

Nagel [21] was one of the first to use second order derivatives to measure optical flow. He introduced the concept of oriented smoothness constraint to replace Horn and Schunck's smoothness constraint. In his constraint, smoothness perpendicular to the gradient direction is allowed; smoothness tangential to the gradient direction is severely attenuated. The direction of smoothness perpendicular to the intensity gradient is computed using 2^{nd} order derivatives of the grayvalues. The problem is formulated as the minimization of the function:

$$\int\int (\nabla I^T \mathbf{v} + I_t)^2 + \frac{\alpha^2}{||\nabla I||_2^2 + 2\delta}[(u_x I_y - u_y I_x)^2 + (v_x I_y - v_y I_x)^2 + \delta(u_x^2 + u_y^2 + v_x^2 + v_y^2)]dxdy. \quad (2.16)$$

This function is minimized using Euler-Lagrange equations and Gauss-Seidel iterations.

2.1.7 Nagel Yosemite Results

For the Yosemite sequence, Barron et al.'s version of Nagel algorithm produced $11.71° \pm 10.59°$ of error. When thresholding on the spatial gradient was used $||\nabla I|| \geq 5.0$, the angular error was reduced to $6.03° \pm 11.04°$ with 32.9% density.

2.1.8 Uras, Girosi, Verri and Torre 1988

The optical flow algorithm presented by Uras, Girosi, Verri and Torre is a 2^{nd}-order technique based on differentiating the motion constraint equation to obtain:

$$\begin{bmatrix} I_x & I_y \\ I_{xx} & I_{yx} \\ I_{xy} & I_{yy} \end{bmatrix} \begin{pmatrix} u \\ v \end{pmatrix} + \begin{pmatrix} I_t \\ I_{xt} \\ I_{yt} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}. \tag{2.17}$$

Theoretically, this equation can be solved at a single pixel (as there are 2 unknowns in 3 linear equations) but in general it is solved in a local neighbourhood using least squares and assuming constant velocity.

2.1.9 Uras et al. Yosemite Results

Barron et al. report an error of $10.44° \pm 15.00°$ for 100% dense flow and $6.73° \pm 16.01°$ for 14.5% dense flow. The latter result is obtained using a threshold of 1.0 on the determinant of the least squares matrix. Simple 3×2 matrices are solved at every pixel for these results.

2.1.10 Ju, Black and Jepson 1996

Ju, Black and Jepson [15] presented a "Skin and Bones" model to compute optical flow. This model uses a parameterized flow model with a smoothness constraint on the

flow parameters to strike a balance between the flexibility of local dense computations and the robustness and accuracy of global parameterized flow models. "Skin and Bones" refers to the fact that the affine patches can be thought of as rigid "bones" connected by flexible "skin". The image is segmented into fixed rectangular patches, and an affine motion is estimated for each patch. A four level Gaussian pyramid was used for coarse to fine processing. To regularize this layered motion representation a new framework was developed for regularization with transparency.

2.1.11 Ju, Black and Jepson Yosemite Results

For the Yosemite sequence, Ju et al.'s algorithm with bones only (no smoothing) obtained $2.77° \pm 3.4°$ while, Ju et al algorithm with both skin and bones obtained $2.16° \pm 2.0°$.

2.1.12 Lai and Vemuri 1998

Lai and Vemuri [16] proposed two algorithms for computing optical flow. The first is a gradient-based regularization method, and the other is an SSD (sum-of-squared-differences) based regularization method. They first discussed the major problem with the gradient-based approach, that is, approximation errors in the partial derivatives are inevitable due to inaccurate numerical approximation and temporal and spatial aliasing which usually occurs in areas where the local brightness function is highly nonlinear or fast changing. The gradient-based method combines the image flow constraint (the motion constraint equation) and a contour-based flow constraint together into a data constraint term. Each data constraint is normalized to obtain an approximate minimum distance (of the data point to the linear flow equation) constraint instead of the conventional linear flow constraint. In addition, a measure is defined to reject unreliable constraints. To solve the linear system associated with this gradient-based method, they developed a numerical algorithm that is based on the

Constraints	Angular Error ± Standard Deviation
Image constraint	2.29° ± 1.79°
Image constraint+contour constraint	2.29° ± 1.79°
Image constraint+contour constraint+ normalization	2.11° ± 1.57°
Image constraint+contour constraint+ normalization+rejection	1.99° ± 1.41°

Table 2.1: The angle error and standard deviation of Lai and Vemuri algorithm for the cloudless Yosemite sequence with various constraints used.

incomplete Cholesky preconditioned conjugate gradient. The SSD-based regularization method uses a normalized SSD measure as the data constraint in a regularization framework. The resulting energy function is neither quadratic nor convex. The nonlinear conjugate gradient in conjunction with an incomplete Cholesky preconditioning was developed to minimize this energy function.

2.1.13 Lai and Vemuri Yosemite Results

Table 2.1 shows the angular error for the cloudless Yosemite sequence for various constraints. For the SSD-based algorithm, the error was 2.04° ± 1.52°, which is slightly higher that the gradient based results of 1.99° ± 1.41°.

2.1.14 Bab-Hadiashar and Suter 1998

Bab-Hadiashar and Suter [4] formulated the measurement of optical flow as a overdetermined set of simultaneous linear equations. Two optical flow methods are proposed. The first method employs the Least Median of Squares (LMedS) to find the initial solution and to classify these estimates as "inliers" or "outliers" and then uses a weighted least squares (WLS) calculation for inlier estimates. The second method employs the Least Median of Squares Orthogonal Distances (LMSOD) model to identify the outliers and uses weighted total least squares to compute optical flow.

2.1.15 Bab-Hadiashar and Suter Yosemite Results

Bab-Hadiashar and Suter present 2 tables of error results for the Yosemite sequence. With clouds, they obtain a solution of $1.74° \pm 2.37°$ with 72% density and a solution of $2.05° \pm 2.92°$ with 100% density. For the cloudless Yosemite sequence (the top 70 rows were zeroed out) they obtain a best result of $1.97° \pm 1.96o$ with 100% density.

2.1.16 Alvarez, Wickert and Sánchez 2000

Alvarez et al. [1] introduce three improvements over Nagel and Enkelmann's optical flow work:

1. Avoidance of inconsistencies caused by centering the brightness term and the smoothness terms in different images,

2. Use of hierarchical coarse-to-fine processing to avoid convergence to incorrect local minima and to handle larger motions (up to 10 pixels per frame) and

3. Design of an energy functional that is invariant under linear brightness changes.

A gradient descent method is applied to the resulting energy functional which leads to a system of diffusion reaction equations.

2.1.17 Alvarez, Wickert and Sánchez Yosemite Results

For the Yosemite sequence with $\alpha = 1$, $s = 0.1$, $\sigma_0 = 5$, $\sigma_n = 1$ and $\eta = 0.95$ Alvarez et al. obtained 5.53° of error. Here α controls the balance between the motion constraint and the smoothness term, s depends on the cumulative histogram of the image gradient magnitude, σ_0 is the standard deviation of the largest of Gaussian, and *eta* is the rescaling factor. No density was given.

2.1.18 Färneback 2001

Färneback [10] presented an algorithm that has 3 distinct components: estimation of spatio-temporal tensors, estimation of parametric motion models and simultaneous segmentation of the motion field. The estimation of orientation tensors involves spatio-temporal filtering of the volume obtained by stacking the frames of an image sequence onto each other. The motion in the sequences is directly related to oriented structures in the volume. An affine model is employed as estimating parametric motion model. Using simultaneous segmentation and velocity estimation, the algorithm can divide the image into a set of regions with different coherent motions, to avoid coherent motion from spanning the discontinuities in motion field.

2.1.19 Farnebäck Yosemite Results

Farnebäck's algorithm gave an angular error of $1.14° \pm 2.14°$ for a 100% dense flow. When m_0 was varied randomly, an average error of between 1.13° and 1.18° was consistently obtained. Here m_0 is a region size threshold.

2.1.20 Mémin and Pérez 2002

Mémin and Pérez [19] proposed a coarse-to fine energy-based model for optical flow estimation. The energy function includes an intensity constancy constraint and a smoothness constraint. In each level of the hierarchy, the image is segmented into square patches. A particular segmentation is found by minimizing function energy, and the flow is projected to the next lower level where the segmentation can be changed again, based on the minimization of the energy function at that level. This incremental minimization can be regarded as a hierarchical Gauss-Newton minimization of the energy function.

2.1.21 Mémin and Pérez Yosemite Results

For the cloudless Yosemite sequence in their Meḿem and Pérez obtain an error of $1.73° \pm 1.33°$ for regular partitioning (all partitions were fixed as square) and $1.93° \pm 1.33°$ for adaptive partitioning. For the Yosemite sequence with clouds they obtained error of $1.58° \pm 1.21°$ when joint parametric motion estimation and segmentation was used and $2.91° \pm 3.17°$ when segmentation was not used.

2.1.22 Brox, Bruhn, Papenberg and Weickert 2004

Brox et al. [7] proposed an energy function which combines three assumptions: a brightness constancy assumption, a gradient constancy assumption and a discontinuity-preserving spatio-temporal smoothness constraint. The constraints included in this model are:

1. Grayvalue constancy assumption, $I(x, y, t) = I(x + u, y + v, t + 1)$, where $\vec{v} = (u, v)$. This function is quite susceptible to slight changes in brightness, which often appear in natural scenes.

2. Gradient constancy assumption, $\nabla I(x, y, t) = \nabla I(x + u, y + v, t + 1)$. Here $\nabla = (\partial_x, \partial_y)^T$ is the spatial gradient. This function requires that the spatial gradient does not vary due to displacement. This criterion is invariant under grayvalue changes.

3. Smoothness assumption, which is based on the Horn and Schunck [14] smoothness assumption with quadratic penalisers. The spatio-temporal gradient which indicates a spatio-temporal smoothness assumption is involved for applications with more than two images.

4. Pyramid coarse to fine warping is used. Warping (using the computed flow to warp the 2^{nd} image towards the 1^{st} image at each level in the pyramid)

removed the current flow from the image. Thus as processing continues down the pyramid towards the original image the flow between the 1^{st} image and the warped 2^{nd} image becomes smaller and smaller.

The grayvalue and gradient assumptions are measured by the energy:

$$\begin{aligned} E_{Data}(u,v) &= \int_{\Omega} \Psi\left(|I(x+u,y+v,t+1)-I(x,y,t)|^2\right) \\ &+ \gamma\left(|\nabla I(x+u,y+v,t+1)-I(x,y,t)|^2\right) dxdydt. \end{aligned} \tag{2.18}$$

Here Ψ does robust estimation, $\Psi(s^2)=\sqrt{s^2+\epsilon}$. $\epsilon = 0.001$. The smoothness term that models the assumption of piecewise smoothness is:

$$E_{smooth}(u,v) = \int_{\Omega} \Psi\left(|\nabla_3 u|^2 + |\nabla_3 v|^2\right) dxdydt. \tag{2.19}$$

Here $\nabla_3 = (\partial_x, \partial_y, \partial_t)^T$ is the spatio-temporal gradient. The total energy is:

$$E(u,v) = E_{Data} + \alpha E_{smooth}. \tag{2.20}$$

For better readability, they define the following abbreviations:

$$I_x = \partial_x I(\mathbf{x}+\mathbf{w}), \tag{2.21}$$

$$I_y = \partial_y I(\mathbf{x}+\mathbf{w}), \tag{2.22}$$

$$I_z = I(\mathbf{x}+\mathbf{w}) - I(\mathbf{x}), \tag{2.23}$$

$$I_{xx} = \partial_{xx} I(\mathbf{x}+\mathbf{w}), \tag{2.24}$$

$$I_{xy} = \partial_{xy} I(\mathbf{x}+\mathbf{w}), \tag{2.25}$$

$$I_{yy} = \partial_{yy} I(\mathbf{x}+\mathbf{w}), \tag{2.26}$$

$$I_{xz} = \partial_x I(\mathbf{x}+\mathbf{w}) - \partial_x I(\mathbf{x}), \tag{2.27}$$

$$I_{yz} = \partial_y I(\mathbf{x}+\mathbf{w}) - \partial_y I(\mathbf{x}). \tag{2.28}$$

According to the calculus of variations, a minimizer of (2.20) must satisfy the Euler-Lagrange equations:

$$\Psi'\left(I_z^2+\gamma(I_{xz}^2+I_{yz}^2)\right)\left(I_xI_z+\gamma(I_{xx}I_{xz}+I_{xy}I_{yz})\right) \qquad (2.29)$$
$$-\alpha\ \mathbf{Div}\ \Psi'\left(|\nabla_3 u|^2+|\nabla_3 v|^2\right)\nabla_3 u=0$$

and

$$\Psi'\left(I_z^2+\gamma(I_{xz}^2+I_{yz}^2)\right)\left(I_yI_z+\gamma(I_{yy}I_{yz}+I_{xy}I_{xz})\right) \qquad (2.30)$$
$$-\alpha\ \mathbf{Div}\ \Psi'\left(|\nabla_3 u|^2+|\nabla_3 v|^2\right)\nabla_3 v=0$$

A consistent numerical scheme based on two data nested fixed point iterations is presented to compute Equations (2.30) and (2.31). In order to find the global minimum, a multiscale warping strategy is used: One starts with solving a coarse, smoothed version of the problem. Then the coarse solution is used to warp the second image back into the first. This is done by building an image pyramid with certain level number and a downsampling factor η from the original image. The computation of velocities begins from the top level, i.e. the coarsest level. Once a solution is obtained, the velocities are projected down to the adjacent finer level, and are used to warp the second image back to the first one. The construction of image pyramid and the projecting of velocities between adjacent levels are done by bilinear interpolation. If the warping iat a level was well performed and the coarse optical flow was good on the previous level most of the motion between the 2 images will have been removed. Further optical flow calculations and warping as one descends the pyramid can lead to a good final optical flow.

2.1.23 Brox, Bruhn, Papenberg and Weickert Yosemite Results

Brox et al. tested their algorithm on the Yosemite sequence with and without clouds. For the 2D version of their algorithm, they got an error of $2.46° \pm 7.31°$ in Yosemite sequence with clouds and $1.59° \pm 1.39°$ in Yosemite sequence without clouds.For the 3D version , they got an error of $1.94° \pm 6.02°$ in Yosemite sequence with clouds and $0.98° \pm 1.17°$ in Yosemite sequence without clouds.

2.1.24 Bruhn, Weickert and Schnörr 2005

Bruhn et al. [8] presented a combined local-global (CLG) approach that incorporates the advantages of both smoothing effects in local differential method (least-square fit of Lucas and Kanade [18]) and global differential method of Horn and Schunck [14]. It yields dense flow fields and is robust against noise. Spatio-temporal and nonlinear extensions as well as multi-resolution frameworks are proposed for this hybrid method.

2.1.25 Bruhn, Weickert and Schnörr Yosemite Results

Bruhn et al. tested their algorithm on the Yosemite sequence with and without clouds. For the 2D and 3D versions of their CLG algorithm they obtained the results reported in Table 2.2.

	2D		3D	
Sequence	Linear	Nonlinear	Linear	Nonlinear
Yosemite (clouds)	7.14°	6.03°	6.18°	5.18°
Yosemite (no clouds)	2.64°	2.31°	1.79°	1.46°

Table 2.2: Angular errors for the 2D and 3D variants of the CLG method using the Yosemite sequence with and without clouds.

Robust Penalty Functions	Angular Error
Quadratic	2.93°
Charbonnier	1.70°
Charbonnier+Lorentzian	1.76°
FoE+Lorentzian	$1.32° \pm 1.41°$

Table 2.3: The errors for the cloudless Yosemite sequence with various robust penalty functions.

2.1.26 Roth and Black 2005

Roth and Black [25] proposed a method for learning the spatial statistics of optical flow fields from a training database. Training flow fields are constructed using range images of natural scenes and 3D camera motions recovered from hand-held and car-mounted video sequences. The machine learning method introduced here is Fields-of-Experts model which models the prior probability of images using a Markov random field. In computing the optical flow, the learned prior is used as a smoothness term and embedded into the combined local-global method (CLG) proposed by Bruhn et al [8].

2.1.27 Roth and Black Yosemite Results

Table 2.3 shows the angular error for the cloudless Yosemite sequence for various robust penalty functions. Densities are 100%.

2.1.28 Papenberg, Bruhn, Brox, Didas and Weickert 2006

Papenberg et al. [23] extended the Brox et al. method [7] and investigated the following additional non-linearized constancy assumptions:

1. Constancy of the Hessian, $H_2 I(x, y, t) = H_2 I(x+u, y+v, t+1)$, which includes second order derivatives.

2. Constancy of the Laplacian: $\Delta I(x,y,t) = \Delta I(x+u, y+v, t+1)$, where $\Delta = \partial_x^2 + \partial_y^2$, which is the trace of the Hessian and is invariant against directional changes. This paper also includes a brief mention of constancy of the gradient norm, constancy of the norm of the Hessian and constancy of the determinant of the Hessian, but in general these constraints were not very useful.

The same hierarchical framework using nest fixed point iterations as was used by Brox et al. [7] is used here.

2.1.29 Papenberg, Bruhn, Brox, Didas and Weickert Yosemite Results

Papenberg et al. tested their algorithm on the Yosemite sequence with and without clouds. For the 2D version of their algorithm, they got an error of $2.44° \pm 6.9°$ in Yosemite sequence with clouds and $1.64° \pm 1.43°$ in Yosemite sequence without clouds.For the 3D version (they consider time as the 3^{rd} dimension in a 3-frame version of 2D Brox), they got an error of $1.78° \pm 7.00°$ in Yosemite sequence with clouds and $0.98° \pm 1.17°$ in Yosemite sequence without clouds.

2.1.30 Amiaz and Kiryati 2006

In the estimation of optical flow, over-smoothing of flow discontinuities accounts for most of the error. Amiaz and Kiryati [2] embeds the algorithm presented by Brox et al. [7] within a two phase active contour segmentation framework. The level set segmentation approach is used to sharpen optical flow discontinuities. They employed the Mumford-Shah segmentation formulation together with the Vese and Chan segmentation algorithm. The initialization scheme, based on affine flow segmentation, is to obtain the initial parameters using a single application of the original algorithm

of Brox et al., followed by detecting the dominant motion using the algorithm of Borshukov et al.

2.1.31 Amiaz and Kiryati Yosemite Results

Amiaz and Kiryati obtained an error of 1.64°±5.82° in Yosemite sequence with clouds.

2.1.32 Faisal and Barron 2007

Faisal and Barron [9] investigated the Brox et al. algorithm [7] in details and proposed 2 variants on Brox et al's algorithm. In the first variant, they adopt standard Horn and Schunck averaging to solve the smoothness term. In the second variant, they employed Brox's technique in his Ph.D thesis to solve the smoothness term but then use Crammer's rule rather than directly use Gauss-Seidel or SOR (Successive Over-Relaxation) to get convergence.

2.1.33 Faisal and Barron Yosemite Results

For cloudy Yosemite data, Faisal and Barron obtained an error of 5.07° ± 9.90° with 4-points differences and 3.60° ± 10.2° with 7 × 7 averaging. For cloudless Yosemite data, Faisal and Barron obtained an error of 2.63° ± 11.68° with 4-points differences and 0.54° ± 5.39° with 7 × 7 averaging. The 7 × 7 averaging results are better than Brox et al.'s results, suggesting smoothing is very important in Brox et al.'s method.

2.1.34 Nir, Bruckstein and Kimmel 2008

Nir et al's algorithm [22] represented optical flow ($u(x,y,t)$,$v(x,y,t)$) by the general over-parameterized space-time model

$$u(x,y,t) = \sum_{i=1}^{n} A_i(x,y,t)\phi_i(x,y,t), \tag{2.31}$$

$$v(x,y,t) = \sum_{i=1}^{n} A_i(x,y,t)\eta_i(x,y,t), \tag{2.32}$$

where $\eta_i(x,y,t)$ and $\phi_i(x,y,t)$, $i = 1,...,n$ are n basis functions of the flow model, while the A_i are space and time varying coefficients of the model. Models includes the affine over-parameterization model, the rigid motion model, the pure translation motion model and the constant motion model. The solution is gained by minimizing the sum of the data and smoothness terms. For an over-parameterization model with n coefficients, there are n Euler-Lagrange equations. Coarse-to-fine resolution is also employed in this paper.

2.1.35 Nir, Bruckstein and Kimmel Yosemite Results

Nir et al.tested their algorithm on the Yosemite sequence without clouds. They obtain an error of $0.96° \pm 1.25°$ with rigid motion 3D smoothness, $0.91° \pm 1.18°$ with affine 3D smoothness and $0.85° \pm 1.18°$ with pure translation 3D smoothness.

2.2 Introduction of the Euler-Lagrange Equation

All the optical flow method mention above that minimize an energy term (starting with Horn and Schunck) to obtain iterative equations to solve for the flow use the Euler-Lagrange equations. We end this chapter with an introduction to these equations.

In the 1750's, Leonhard Euler and Joseph-Louis Lagrange developed a method, later called the Euler-Lagrange method, to solve functions which extremize a given functional. It is analogous to the result from calculus where a function attains its extreme values when its derivative vanishes. In case of 1D, a system can be written as $F(x, f(x), f'(x))$, where f is a function with continuous first order partial derivatives. Any f that extremizes the cost function,

$$J = \int_a^b F(x, f(x), f'(x))dx, \tag{2.33}$$

must satisfies the ordinary differential equation,

$$\frac{\partial F}{\partial f} - \frac{d}{dx}\frac{\partial F}{\partial f'} = 0. \tag{2.34}$$

The multidimensional version of the Euler-Lagrange equations is:

$$J = \int_\Omega F(f, x_1, ..., x_n, f_{x1}, ..., f_{fn}), \tag{2.35}$$

here we have a function on n variables and Ω being some surface. The function is extremized if and only if f satisfies the partial differential equation:

$$\frac{\partial F}{\partial f} - \sum_{i=1}^{n}\left(\frac{\partial}{\partial x_i}\frac{\partial F}{\partial f_{x_i}}\right) = 0. \tag{2.36}$$

In 2D the ordinary differential equations are given by Equations (2.5a) and (2.6) in Section 2.1.2. In 3D the ordinary differential equations are given by Equations (3.3), (3.4) and (3.5) in Section 3.1 for the Horn and Schunch functional, f, to be minimized with a Horn and Schunck derivation following.

Chapter 3

Theoretical Technique

In this chapter, the 3D Horn and Schunck and 3D Brox et al. algorithms are described in details. We also provide a 3D version of Uras et al. algorithm as a method the verify the correctness of 1^{st}-order and 2^{nd}-order derivatives generated in Chapter 4

3.1 3D Horn and Schunck

Barron [5] extended the 2D Horn and Schunck into 3D. The 3D motion constraint equation can be written as:

$$E_{motion} = \nabla \vec{I}^T \cdot \vec{V} + I_t = 0. \tag{3.1}$$

where $\vec{I} = (I_x, I_y, I_z)$ is the spatial gradient, I_t is the temporal grayvalue derivative and $\vec{v} = (u, v, w)$ is the 3D velocity. We need to minimize the energy functional:

$$f(x, y, z, u, v, w, u_x, u_y, u_z, v_x, v_y, v_z, w_x, w_y, w_z) = (I_x u + I_y v + I_z w + I_t)^2 +$$

$$\alpha^2 \left[\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial u}{\partial z}\right)^2 + \right.$$

$$\left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial z}\right)^2 +$$

$$\left(\frac{\partial w}{\partial x}\right)^2 + \left(\frac{\partial w}{\partial y}\right)^2 + \left(\frac{\partial w}{\partial z}\right)^2\Bigg]. \tag{3.2}$$

The first part of this energy function specifies the 3D brightness constraint, which is based on the assumption that the grayvalue of a pixel will stay the same when it is moved to a new location. The second part of the function specifies the global smoothness term, assuming that neighboring points have similar velocities and velocities varies smoothly almost everywhere in an image. The general 3D Euler-Lagrange equations are:

$$f_u - \frac{df_{u_x}}{dx} - \frac{df_{u_y}}{dy} - \frac{df_{u_z}}{dz} = 0, \tag{3.3}$$

$$f_v - \frac{df_{v_x}}{dx} - \frac{df_{v_y}}{dy} - \frac{df_{v_z}}{dz} = 0, \tag{3.4}$$

$$f_w - \frac{df_{w_x}}{dx} - \frac{df_{w_y}}{dy} - \frac{df_{w_z}}{dz} = 0, \tag{3.5}$$

where:

$$f_u = 2I_x(I_x u + I_y v + I_z w + I_t), \tag{3.6}$$

$$f_v = 2I_y(I_x u + I_y v + I_z w + I_t), \tag{3.7}$$

$$f_w = 2I_z(I_x u + I_y v + I_z w + I_t), \tag{3.8}$$

$$f_{u_x} = 2\alpha^2 u_x, \tag{3.9}$$

$$f_{u_y} = 2\alpha^2 u_y, \tag{3.10}$$

$$f_{u_z} = 2\alpha^2 u_z, \tag{3.11}$$

$$f_{v_x} = 2\alpha^2 v_x, \tag{3.12}$$

$$f_{v_y} = 2\alpha^2 v_y, \tag{3.13}$$

$$f_{v_z} = 2\alpha^2 v_z, \tag{3.14}$$

$$f_{w_x} = 2\alpha^2 w_x, \tag{3.15}$$

$$f_{w_y} = 2\alpha^2 w_y, \tag{3.16}$$

$$f_{w_z} = 2\alpha^2 w_z, \tag{3.17}$$

$$\frac{df_{u_x}}{dx} = 2\alpha^2 u_{xx}, \tag{3.18}$$

$$\frac{df_{u_y}}{dy} = 2\alpha^2 u_{yy}, \tag{3.19}$$

$$\frac{df_{u_z}}{dz} = 2\alpha^2 u_{zz}, \tag{3.20}$$

$$\frac{df_{v_x}}{dx} = 2\alpha^2 v_{xx}, \tag{3.21}$$

$$\frac{df_{v_y}}{dy} = 2\alpha^2 v_{yy}, \tag{3.22}$$

$$\frac{df_{v_z}}{dz} = 2\alpha^2 v_{zz}, \tag{3.23}$$

$$\frac{df_{w_x}}{dx} = 2\alpha^2 w_{xx}, \tag{3.24}$$

$$\frac{df_{w_y}}{dy} = 2\alpha^2 w_{yy}, \tag{3.25}$$

$$\frac{df_{w_z}}{dz} = 2\alpha^2 w_{zz}. \tag{3.26}$$

Using $\nabla^2 u = u_{xx} + u_{yy} + u_{zz}$, $\nabla^2 v = v_{xx} + v_{yy} + v_{zz}$ and $\nabla^2 w = w_{xx} + w_{yy} + w_{zz}$ we can rewrite the Euler-Lagrange equations as:

$$I_x^2 u + I_x I_y v + I_x I_z w + I_x I_t = \alpha^2 \nabla^2 u, \tag{3.27}$$

$$I_x I_y u + I_y^2 v + I_y I_z w + I_y I_t = \alpha^2 \nabla^2 v, \tag{3.28}$$

$$I_x I_z u + I_y I_z v + I_z^2 w + I_z I_t = \alpha^2 \nabla^2 w. \tag{3.29}$$

Using standard averaging to solve the smoothness term, $\nabla^2 u \approx \bar{u} - u$, $\nabla^2 v \approx \bar{v} - v$ and $\nabla^2 w \approx \bar{w} - w$ we get:

$$(\alpha^2 + I_x^2)u + I_x I_y v + I_x I_z w = (\alpha^2 \bar{u} - I_x I_t), \tag{3.30}$$

$$I_x I_y u + (\alpha^2 + I_y^2)v + I_y I_z w = (\alpha^2 \bar{v} - I_y I_t), \tag{3.31}$$

$$I_x I_z u + I_y I_z v + (\alpha^2 + I_z^2)w = (\alpha^2 \bar{w} - I_z I_t), \tag{3.32}$$

or in matrix form:

$$\begin{bmatrix} (\alpha^2 + I_x^2) & I_x I_y & I_x I_z \\ I_x I_y & (\alpha^2 + I_y^2) & I_y I_z \\ I_x I_z & I_y I_z & (\alpha^2 + I_z^2) \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} (\alpha^2 \bar{u} - I_x I_t) \\ (\alpha^2 \bar{v} - I_y I_t) \\ (\alpha^2 \bar{w} - I_z I_t) \end{bmatrix}. \tag{3.33}$$

The solution of this system of equations is:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} (\alpha^2 + I_x^2) & I_x I_y & I_x I_z \\ I_x I_y & (\alpha^2 + I_y^2) & I_y I_z \\ I_x I_z & I_y I_z & (\alpha^2 + I_z^2) \end{bmatrix}^{-1} \begin{bmatrix} (\alpha^2 \bar{u} - I_x I_t) \\ (\alpha^2 \bar{v} - I_y I_t) \\ (\alpha^2 \bar{w} - I_z I_t) \end{bmatrix}. \tag{3.34}$$

Using the Gauss-Seidel method, we obtain the iterative equations:

$$u^{n+1} = \bar{u}^n - \frac{I_x \left[I_x \bar{u} + I_y \bar{v} + I_z \bar{w} + I_t\right]}{(\alpha^2 + I_x^2 + I_y^2 + I_z^2)}, \tag{3.35}$$

$$v^{n+1} = \bar{v}^n - \frac{I_y \left[I_x \bar{u} + I_y \bar{v} + I_z \bar{w} + I_t\right]}{(\alpha^2 + I_x^2 + I_y^2 + I_z^2)}, \tag{3.36}$$

$$w^{n+1} = \bar{w}^n - \frac{I_z \left[I_x \bar{u} + I_y \bar{v} + I_z \bar{w} + I_t\right]}{(\alpha^2 + I_x^2 + I_y^2 + I_z^2)}. \tag{3.37}$$

The value of α was typically set to 1.0 or 10.0, which is based on experiential observation. The number of iterations used was typically either 50, 100 or 200. It may also be a good choice to define certain stopping criteria for iterations, for example, if changes in velocity $\vec{v}$ is smaller than a number τ, i.e. $\|\vec{v_m} - \vec{v_{m+1}}\|_2 < \tau$, then

the iteration stops. Nonetheless, Brox et al. method uses fixed iteration number, for better comparison, we also choose to set fixed iteration number.

In this thesis, we also implement a hierarchical 3D Horn and Schunck algorithm which employs the same warping and interpolation techniques as the 3D Brox et al. algorithm (see Section 3.2.10 and 3.2.11 for details).

3.2 3D Brox, Bruhn, Papenberg and Weickert Algorithm

We seek to extend 2D Brox et al. optical flow method to 3D. The Brox et al. algorithm [7] is composed of several constraints.

3.2.1 The Variational Model

We extend Brox et al.s assumptions to 3D. We describe this in the following subsections.

3.2.1.1 Gray value constancy assumption

Since the beginning of optical flow research, it is widely assumed that the intensity at a particular point (x, y) in the image is constant. That is, the gray value of a pixel does not change when it moves to another location. Therefore, we have:

$$I(x, y, z, t) \;=\; I(x + u, y + v, z + w, t + 1). \tag{3.38}$$

Here $I : \Omega \subset \mathbb{R}^4 \rightarrow \mathbb{R}$ denotes a cubical volume sequence, and $\mathbf{w} := (u, v, w, 1)^\top$ is the searched displacement vector between an image at time t and another image at

time $t+1$. Performing a 1^{st} order Taylor series expansion of $I(x,y,z,t)$, we get:

$$I(x+u,y+v,z+w,t+1) = I(x,y,z,t) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial z}w + \frac{\partial I}{\partial t} + H.O.T., \quad (3.39)$$

where $H.O.T.$ stands for higher order terms. Combining Equations (3.38) and (3.39) yields the optical flow constraint (motion constraint) equation in 3D:

$$\begin{aligned} \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial z}w + \frac{\partial I}{\partial t} &= 0 \text{ or} \\ \Rightarrow \quad I_x u + I_y v + I_z w + I_t &= 0, \end{aligned} \quad (3.40)$$

where subscripts denote partial derivatives. However, this linear equation is only valid under the assumption that the image changes linearly along the displacement, which is usually violated in general cases, especially when there are larger displacements. Therefore, this model uses the original grayvalue constancy assumption given by equation (3.38).

3.2.1.2 Gradient constancy assumption

One big drawback of the grayvalue constancy assumption is that it is quite susceptible to slight changes in brightness, which often appear in natural scenes. Therefore, it is useful to introduce a constraint that is invariant under gray value changes to allow some small variations in the grayvalue. Assuming the gradient of the image gray value is not vary due to the displacement, this will give:

$$\nabla I(x,y,z,t) = \nabla I(x+u,y+v,z+w,t+1). \quad (3.41)$$

Here $\nabla = (\partial_x, \partial_y, \partial_z)^\top$ denotes the spatial gradient. This constraint is particularly helpful for translatory motion.

3.2.1.3 Smoothness assumption

The constraints described in the last 2 subsections only consider displacement of a pixel locally without taking into account the interaction between neighboring pixels. Therefore, if the gradient vanishes somewhere, or if the flow can only be computed in the normal direction to the gradient (aperture problem), this model will not work. Also, it is quite possible to get some outliers in the solution. As a result, we need to include a smoothness assumption to prevent these problems. This constraint can either be applied solely to the spatial domain or in the spatio-temporal domain. In the case we are interested in, there are only two images available, so we employ the constraint in the spatial domain only. If the displacements in a sequence of images are needed, we can apply it to the spatio-temporal domain. Since the optimal displacement field will have discontinuities at the boundaries of objects in the scene, it is reasonable to generalize the smoothness assumption by demanding a piecewise smooth flow field.

3.2.1.4 Multiscale Approach

It is quite likely to get displacements that are larger than one pixel or more. In that case, the minimization algorithm can easily be trapped in a local minimum. In order to find the global minimum, it can be useful to apply multiscale ideas: one starts by solving a coarse, smoothed version of the problem on a smoothed image sequence. The new problem may have a unique minimum, hopefully close to the global minimum of the original problem. The coarse solution is used as initialization for solving refined versions of the problem step by step until the original problem is obtained. Instead of smoothing over and over again, it is sensible to downsample the images respecting the sampling theorem.

3.2.2 The Energy Functional

Now we have all the assumptions we need to construct the 3D energy functional of Brox et al. optical flow algorithm [7]. The grayvalue and gradient assumptions are measured by the energy:

$$\begin{aligned} E_{Data}(u,v,w) &= \int_{\Omega} \left(|I(x+u,y+v,z+w,t+1) - I(x,y,z,t)|^2 \right. \\ &+ \left. \gamma \, |\nabla I(x+u,y+v,z+w,t+1) - \nabla I(x,y,z,t)|^2 \right) dxdydzdt. \end{aligned} \tag{3.42}$$

where γ is a weighting factor between these two assumptions. The 1^{st} term minimizes the displaced image intensities and the 2^{nd} term minimizes the displaced image gradients. Since a quadratic penalize gives outliers too much influence on the estimation, an increasing, nearly linear, concave function $\Psi(s^2)$ which behaves as if a linear function was applied, leading to a robust energy:

$$\begin{aligned} E_{Data}(u,v,w) &= \int_{\Omega} \Psi \left(|I(x+u,y+v,z+w,t+1) - I(x,y,z,t)|^2 \right. \\ &+ \left. \gamma \, |\nabla I(x+u,y+v,z+w,t+1) - \nabla I(x,y,z,t)|^2 \right) dxdydzdt. \end{aligned} \tag{3.43}$$

Here Ψ does robust estimation, $\Psi(s^2) = \sqrt{s^2 + \epsilon}$. Brox et al. [7] use $\epsilon = 0.001$. A smoothness term that models the assumption of piecewise smoothness is:

$$E_{smooth}(u,v,w) = \int_{\Omega} \Psi \left(|\nabla_4 u|^2 + |\nabla_4 v|^2 + |\nabla_4 w|^2 \right) dxdydzdt. \tag{3.44}$$

Here $\nabla_4 = (\partial_x, \partial_y, \partial_z, \partial_t)^T$ is the spatio-temporal gradient. The total energy is the weighted sum between the data term and the smoothness term:

$$E(u,v,w) = E_{Data} + \alpha E_{smooth}, \tag{3.45}$$

with some regularization parameter $\alpha > 0$. Now the goal is to find the functions u, v and w that minimizes this energy.

3.2.3 Derivation of Euler-Lagrange Equations

$E(u, v, w)$ is nonlinear. Notice that we assume symmetric 2^{nd} order derivatives, i.e. $I_{xy} = I_{yx}$, $I_{yz} = I_{zy}$ and $I_{xz} = I_{zx}$. For better readability, we use the following abbreviations (based on those used in [7]), with D meaning difference and not a temporal derivative :

$$I_x = \partial_x I(x+u, y+v, z+w, t+1), \tag{3.46}$$

$$I_y = \partial_y I(x+u, y+v, z+w, t+1), \tag{3.47}$$

$$I_z = \partial_z I(x+u, y+v, z+w, t+1), \tag{3.48}$$

$$I_D = I(x+u, y+v, z+w, t+1) - I(x, y, z, t), \tag{3.49}$$

$$I_{xx} = \partial_{xx} I(x+u, y+v, z+w, t+1), \tag{3.50}$$

$$I_{xy} = \partial_{xy} I(x+u, y+v, z+w, t+1), \tag{3.51}$$

$$I_{xz} = \partial_{xz} I(x+u, y+v, z+w, t+1), \tag{3.52}$$

$$I_{yy} = \partial_{yy} I(x+u, y+v, z+w, t+1), \tag{3.53}$$

$$I_{yz} = \partial_{yz} I(x+u, y+v, z+w, t+1), \tag{3.54}$$

$$I_{zz} = \partial_{zz} I(x+u, y+v, z+w, t+1), \tag{3.55}$$

$$I_{xD} = \partial_x I(x+u, y+v, z+w, t+1) - \partial_x I(x, y, z, t), \tag{3.56}$$

$$I_{yD} = \partial_y I(x+u, y+v, z+w, t+1) - \partial_y I(x, y, z, t), \tag{3.57}$$

$$I_{zD} = \partial_z I(x+u, y+v, z+w, t+1) - \partial_z I(x, y, z, t). \tag{3.58}$$

According to the calculus of variations, a minimizer of (3.45) must fulfill the Euler-Lagrange equations derived from the energy functional. The general 3D Euler-

Lagrange equations are:

$$f_u - \frac{df_{u_x}}{dx} - \frac{df_{u_y}}{dy} - \frac{df_{u_z}}{dz} = 0, \tag{3.59}$$

$$f_v - \frac{df_{v_x}}{dx} - \frac{df_{v_y}}{dy} - \frac{df_{v_z}}{dz} = 0, \tag{3.60}$$

$$f_w - \frac{df_{w_x}}{dx} - \frac{df_{w_y}}{dy} - \frac{df_{w_z}}{dz} = 0, \tag{3.61}$$

where f is the 3D extension of Brox et al. function to be minimized:

$$f = \Psi_{Data} + \alpha\Psi_{Smooth}. \tag{3.62}$$

Let $I(\mathbf{x}+\mathbf{w})$ denote $I(x+u, y+v, z+w, t+1)$ and $I(\mathbf{x})$ denote $I(x, y, z, t)$. Then:

$$f = \Psi\left(|I(\mathbf{x}+\mathbf{w}) - I(\mathbf{x})|^2 + \gamma(|\nabla I(\mathbf{x}+\mathbf{w}) - \nabla I(\mathbf{x})|^2\right) +$$

$$\alpha\Psi\left(|\nabla_3 u|^2 + |\nabla_3 v|^2 + |\nabla_3 w|^2\right). \tag{3.63}$$

Follow the abbreviations in Equations (3.49), (3.56), (3.57) and (3.58), we have:

$$f = \Psi\left(I_D^2 + \gamma(I_{xD}^2 + I_{yD}^2 + I_{zD}^2)\right) + \alpha\Psi\left(u_x^2 + u_y^2 + u_z^2 + v_x^2 + v_y^2 + v_z^2 + w_x^2 + w_y^2 + w_z^2\right). \tag{3.64}$$

Note that $|x, y, z|$ is the magnitude of vector (x, y, z), i.e. $|x, y, z|^2 = \left(\sqrt{x^2+y^2+z^2}\right)^2$ $= x^2 + y^2 + z^2$. ∇_3 is the spatial gradient only, $\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right)$, when we are using only 2 images $\mathbf{x}$ is (x, y, z, t) and $\mathbf{w}$ is $(u, v, w, 1)$. u , v and w are 2-volume optical flow values and $\delta t = 1$ for 2 adjacent frames in an image sequence. Differentiating f in Equation (3.64) with respect to u, v and w requires the evaluation of the following

derivatives:

$$\frac{|I(\mathbf{x}+\mathbf{w})-I(\mathbf{x})|^2}{\partial u} = 2\left(|I(\mathbf{x}+\mathbf{w})-I(\mathbf{x})|\right)\frac{\partial I(\mathbf{x}+\mathbf{w})}{\partial u} = 2I_D\frac{\partial I(\mathbf{x}+\mathbf{w})}{\partial u}, \tag{3.65}$$

$$\begin{aligned}\frac{|\nabla I(\mathbf{x}+\mathbf{w})-\nabla I(\mathbf{x})|^2}{\partial u} &= 2|(I_x(\mathbf{x}+\mathbf{w}),I_y(\mathbf{x}+\mathbf{w}),I_z(\mathbf{x}+\mathbf{w}))-(I_x(\mathbf{x}),I_y(\mathbf{x}),I_z(\mathbf{x}))| \\ &\quad \frac{\partial(I_x(\mathbf{x}+\mathbf{w}),I_y(\mathbf{x}+\mathbf{w}),I_z(\mathbf{x}+\mathbf{w}))}{\partial u} \\ &= 2(I_{xD},I_{yD},I_{zD})\frac{\partial(I_x(\mathbf{x}+\mathbf{w}),I_y(\mathbf{x}+\mathbf{w}),I_z(\mathbf{x}+\mathbf{w}))}{\partial u},\end{aligned} \tag{3.66}$$

$$\frac{|I(\mathbf{x}+\mathbf{w})-I(\mathbf{x})|^2}{\partial v} = 2\left(|I(\mathbf{x}+\mathbf{w})-I(\mathbf{x})|\right)\frac{\partial I(\mathbf{x}+\mathbf{w})}{\partial v} = 2I_D\frac{\partial I(\mathbf{x}+\mathbf{w})}{\partial v}, \tag{3.67}$$

$$\begin{aligned}\frac{|\nabla I(\mathbf{x}+\mathbf{w})-\nabla I(\mathbf{x})|^2}{\partial v} &= 2|(I_x(\mathbf{x}+\mathbf{w}),I_y(\mathbf{x}+\mathbf{w}),I_z(\mathbf{x}+\mathbf{w}))-(I_x(\mathbf{x}),I_y(\mathbf{x}),I_z(\mathbf{x}))| \\ &\quad \frac{\partial(I_x(\mathbf{x}+\mathbf{w}),I_y(\mathbf{x}+\mathbf{w}),I_z(\mathbf{x}+\mathbf{w}))}{\partial v} \\ &= 2(I_{xD},I_{yD},I_{zD})\frac{\partial(I_x(\mathbf{x}+\mathbf{w}),I_y(\mathbf{x}+\mathbf{w}),I_z(\mathbf{x}+\mathbf{w}))}{\partial v},\end{aligned} \tag{3.68}$$

$$\frac{|I(\mathbf{x}+\mathbf{w})-I(\mathbf{x})|^2}{\partial w} = 2\left(|I(\mathbf{x}+\mathbf{w})-I(\mathbf{x})|\right)\frac{\partial I(\mathbf{x}+\mathbf{w})}{\partial w} = 2I_D\frac{\partial I(\mathbf{x}+\mathbf{w})}{\partial w}, \tag{3.69}$$

$$\begin{aligned}\frac{|\nabla I(\mathbf{x}+\mathbf{w})-\nabla I(\mathbf{x})|^2}{\partial w} &= 2|(I_x(\mathbf{x}+\mathbf{w}),I_y(\mathbf{x}+\mathbf{w}),I_z(\mathbf{x}+\mathbf{w}))-(I_x(\mathbf{x}),I_y(\mathbf{x}),I_z(\mathbf{x}))| \\ &\quad \frac{\partial(I_x(\mathbf{x}+\mathbf{w}),I_y(\mathbf{x}+\mathbf{w}),I_z(\mathbf{x}+\mathbf{w}))}{\partial w} \text{ and} \\ &= 2(I_{xD},I_{yD},I_{zD})\frac{\partial(I_x(\mathbf{x}+\mathbf{w}),I_y(\mathbf{x}+\mathbf{w}),I_z(\mathbf{x}+\mathbf{w}))}{\partial w}.\end{aligned} \tag{3.70}$$

Note that in Equations (3.65) to (3.70) we used the definitions in Equations (3.46) to (3.58). Using the chain rule, we can evaluate the remaining derivatives in Equations (3.65) and (3.66) as:

$$\begin{aligned}\frac{\partial I(\mathbf{x}+\mathbf{w})}{\partial u} &= \frac{\partial I(\mathbf{x}+\mathbf{w})}{\partial x}\frac{\partial(x+u)}{\partial u} \\ &+ \frac{\partial I(\mathbf{x}+\mathbf{w})}{\partial y}\frac{\partial(y+v)}{\partial u} \\ &+ \frac{\partial I(\mathbf{x}+\mathbf{w})}{\partial z}\frac{\partial(z+w)}{\partial u} \\ &+ \frac{\partial I(\mathbf{x}+\mathbf{w})}{\partial t}\frac{\partial(t+1)}{\partial u} \\ &= \frac{\partial I(\mathbf{x}+\mathbf{w})}{\partial x} = I_x(\mathbf{x}+\mathbf{w}),\end{aligned} \tag{3.71}$$

$$\begin{aligned}\frac{\partial I_x(\mathbf{x}+\mathbf{w})}{\partial u} &= \frac{\partial I_x(\mathbf{x}+\mathbf{w})}{\partial x}\frac{\partial (x+u)}{\partial u} \\ &+ \frac{\partial I_x(\mathbf{x}+\mathbf{w})}{\partial y}\frac{\partial (y+v)}{\partial u} \\ &+ \frac{\partial I_x(\mathbf{x}+\mathbf{w})}{\partial z}\frac{\partial (z+w)}{\partial u} \\ &+ \frac{\partial I_x(\mathbf{x}+\mathbf{w})}{\partial t}\frac{\partial (t+1)}{\partial u} \\ &= \frac{\partial I_x(\mathbf{x}+\mathbf{w})}{\partial x} = I_{xx}(\mathbf{x}+\mathbf{w}), \end{aligned} \tag{3.72}$$

$$\begin{aligned}\frac{\partial I_y(\mathbf{x}+\mathbf{w})}{\partial u} &= \frac{\partial I_y(\mathbf{x}+\mathbf{w})}{\partial x}\frac{\partial (x+u)}{\partial u} \\ &+ \frac{\partial I_y(\mathbf{x}+\mathbf{w})}{\partial y}\frac{\partial (y+v)}{\partial u} \\ &+ \frac{\partial I_y(\mathbf{x}+\mathbf{w})}{\partial z}\frac{\partial (z+w)}{\partial u} \\ &+ \frac{\partial I_y(\mathbf{x}+\mathbf{w})}{\partial t}\frac{\partial (t+1)}{\partial u} \\ &= \frac{\partial I_y(\mathbf{x}+\mathbf{w})}{\partial x} = I_{xy}(\mathbf{x}+\mathbf{w}), \end{aligned} \tag{3.73}$$

and

$$\begin{aligned}\frac{\partial I_z(\mathbf{x}+\mathbf{w})}{\partial u} &= \frac{\partial I_z(\mathbf{x}+\mathbf{w})}{\partial x}\frac{\partial (x+u)}{\partial u} \\ &+ \frac{\partial I_z(\mathbf{x}+\mathbf{w})}{\partial y}\frac{\partial (y+v)}{\partial u} \\ &+ \frac{\partial I_z(\mathbf{x}+\mathbf{w})}{\partial z}\frac{\partial (z+w)}{\partial u} \\ &+ \frac{\partial I_z(\mathbf{x}+\mathbf{w})}{\partial t}\frac{\partial (t+1)}{\partial u} \\ &= \frac{\partial I_z(\mathbf{x}+\mathbf{w})}{\partial x} = I_{xz}(\mathbf{x}+\mathbf{w}). \end{aligned} \tag{3.74}$$

Again, using the chain rule, we can evaluate the remaining derivatives in Equations (3.67) and (3.68) as:

$$\begin{aligned}\frac{\partial I(\mathbf{x}+\mathbf{w})}{\partial v} &= \frac{\partial I(\mathbf{x}+\mathbf{w})}{\partial x}\frac{\partial (x+u)}{\partial v} \\ &+ \frac{\partial I(\mathbf{x}+\mathbf{w})}{\partial y}\frac{\partial (y+v)}{\partial v} \\ &+ \frac{\partial I(\mathbf{x}+\mathbf{w})}{\partial z}\frac{\partial (z+w)}{\partial v} \\ &+ \frac{\partial I(\mathbf{x}+\mathbf{w})}{\partial t}\frac{\partial (t+1)}{\partial v} \\ &= \frac{\partial I(\mathbf{x}+\mathbf{w})}{\partial y} = I_y(\mathbf{x}+\mathbf{w}),\end{aligned} \tag{3.75}$$

$$\begin{aligned}\frac{\partial I_x(\mathbf{x}+\mathbf{w})}{\partial v} &= \frac{\partial I_x(\mathbf{x}+\mathbf{w})}{\partial x}\frac{\partial (x+u)}{\partial v} \\ &+ \frac{\partial I_x(\mathbf{x}+\mathbf{w})}{\partial y}\frac{\partial (y+v)}{\partial v} \\ &+ \frac{\partial I_x(\mathbf{x}+\mathbf{w})}{\partial z}\frac{\partial (z+w)}{\partial v} \\ &+ \frac{\partial I_x(\mathbf{x}+\mathbf{w})}{\partial t}\frac{\partial (t+1)}{\partial v} \\ &= \frac{\partial I_x(\mathbf{x}+\mathbf{w})}{\partial y} = I_{xy}(\mathbf{x}+\mathbf{w}),\end{aligned} \tag{3.76}$$

$$\begin{aligned}\frac{\partial I_y(\mathbf{x}+\mathbf{w})}{\partial v} &= \frac{\partial I_y(\mathbf{x}+\mathbf{w})}{\partial x}\frac{\partial (x+u)}{\partial v} \\ &+ \frac{\partial I_y(\mathbf{x}+\mathbf{w})}{\partial y}\frac{\partial (y+v)}{\partial v} \\ &+ \frac{\partial I_y(\mathbf{x}+\mathbf{w})}{\partial z}\frac{\partial (z+w)}{\partial v} \\ &+ \frac{\partial I_y(\mathbf{x}+\mathbf{w})}{\partial t}\frac{\partial (t+1)}{\partial v} \\ &= \frac{\partial I_y(\mathbf{x}+\mathbf{w})}{\partial y} = I_{yy}(\mathbf{x}+\mathbf{w}),\end{aligned} \tag{3.77}$$

and

$$\begin{aligned}\frac{\partial I_z(\mathbf{x}+\mathbf{w})}{\partial v} &= \frac{\partial I_z(\mathbf{x}+\mathbf{w})}{\partial x}\frac{\partial (x+u)}{\partial v} \\ &+ \frac{\partial I_z(\mathbf{x}+\mathbf{w})}{\partial y}\frac{\partial (y+v)}{\partial v} \\ &+ \frac{\partial I_z(\mathbf{x}+\mathbf{w})}{\partial z}\frac{\partial (z+w)}{\partial v} \\ &+ \frac{\partial I_z(\mathbf{x}+\mathbf{w})}{\partial t}\frac{\partial (t+1)}{\partial v} \\ &= \frac{\partial I_z(\mathbf{x}+\mathbf{w})}{\partial y} = I_{yz}(\mathbf{x}+\mathbf{w}). \end{aligned} \tag{3.78}$$

The remaining derivatives in Equations (3.69) and (3.70) are as follows:

$$\begin{aligned}\frac{\partial I(\mathbf{x}+\mathbf{w})}{\partial w} &= \frac{\partial I(\mathbf{x}+\mathbf{w})}{\partial x}\frac{\partial (x+u)}{\partial w} \\ &+ \frac{\partial I(\mathbf{x}+\mathbf{w})}{\partial y}\frac{\partial (y+v)}{\partial w} \\ &+ \frac{\partial I(\mathbf{x}+\mathbf{w})}{\partial z}\frac{\partial (z+w)}{\partial w} \\ &+ \frac{\partial I(\mathbf{x}+\mathbf{w})}{\partial t}\frac{\partial (t+1)}{\partial w} \\ &= \frac{\partial I(\mathbf{x}+\mathbf{w})}{\partial z} = I_z(\mathbf{x}+\mathbf{w}), \end{aligned} \tag{3.79}$$

$$\begin{aligned}\frac{\partial I_x(\mathbf{x}+\mathbf{w})}{\partial w} &= \frac{\partial I_x(\mathbf{x}+\mathbf{w})}{\partial x}\frac{\partial (x+u)}{\partial w} \\ &+ \frac{\partial I_x(\mathbf{x}+\mathbf{w})}{\partial y}\frac{\partial (y+v)}{\partial w} \\ &+ \frac{\partial I_x(\mathbf{x}+\mathbf{w})}{\partial z}\frac{\partial (z+w)}{\partial w} \\ &+ \frac{\partial I_x(\mathbf{x}+\mathbf{w})}{\partial t}\frac{\partial (t+1)}{\partial w} \\ &= \frac{\partial I_x(\mathbf{x}+\mathbf{w})}{\partial z} = I_{xz}(\mathbf{x}+\mathbf{w}), \end{aligned} \tag{3.80}$$

$$\begin{aligned}\frac{\partial I_y(\mathbf{x}+\mathbf{w})}{\partial w} &= \frac{\partial I_y(\mathbf{x}+\mathbf{w})}{\partial x}\frac{\partial (x+u)}{\partial w} \\ &+ \frac{\partial I_y(\mathbf{x}+\mathbf{w})}{\partial y}\frac{\partial (y+v)}{\partial w} \\ &+ \frac{\partial I_y(\mathbf{x}+\mathbf{w})}{\partial z}\frac{\partial (z+w)}{\partial w} \\ &+ \frac{\partial I_y(\mathbf{x}+\mathbf{w})}{\partial t}\frac{\partial (t+1)}{\partial w} \\ &= \frac{\partial I_y(\mathbf{x}+\mathbf{w})}{\partial z} = I_{yz}(\mathbf{x}+\mathbf{w}), \end{aligned} \tag{3.81}$$

and

$$\begin{aligned}\frac{\partial I_z(\mathbf{x}+\mathbf{w})}{\partial w} &= \frac{\partial I_z(\mathbf{x}+\mathbf{w})}{\partial x}\frac{\partial (x+u)}{\partial w} \\ &+ \frac{\partial I_z(\mathbf{x}+\mathbf{w})}{\partial y}\frac{\partial (y+v)}{\partial w} \\ &+ \frac{\partial I_z(\mathbf{x}+\mathbf{w})}{\partial z}\frac{\partial (z+w)}{\partial w} \\ &+ \frac{\partial I_z(\mathbf{x}+\mathbf{w})}{\partial t}\frac{\partial (t+1)}{\partial w} \\ &= \frac{\partial I_z(\mathbf{x}+\mathbf{w})}{\partial z} = I_{zz}(\mathbf{x}+\mathbf{w}). \end{aligned} \tag{3.82}$$

$\Psi(s^2)$ is defined as $\sqrt{s^2+\epsilon}$. Then the derivative of $\Psi(s^2)$ is given as:

$$\Psi\,'(s^2) = \frac{1}{2\sqrt{s^2+\epsilon}}. \tag{3.83}$$

Using Equations (3.64), (3.65), (3.66), (3.71), (3.72), (3.73) and (3.74), we can rewrite the equation for f_u as:

$$\begin{aligned} f_u &= \Psi\,'(I_D^2+\gamma(I_{xD}^2+I_{yD}^2+I_{zD}^2))\cdot(\frac{|I(\mathbf{x}+\mathbf{w})-I(\mathbf{x})|^2}{\partial u}+\frac{|\nabla I(\mathbf{x}+\mathbf{w})-\nabla I(\mathbf{x})|^2}{\partial u}) \\ &= \Psi\,'(I_D^2+\gamma(I_{xD}^2+I_{yD}^2+I_{zD}^2))\cdot(2I_D\frac{\partial I(\mathbf{x}+\mathbf{w})}{\partial u} \\ &+ 2(I_{xD},I_{yD},I_{zD})\frac{\partial (I_x(\mathbf{x}+\mathbf{w}),I_y(\mathbf{x}+\mathbf{w}),I_z(\mathbf{x}+\mathbf{w}))}{\partial u}) \\ &= \Psi\,'(I_D^2+\gamma(I_{xD}^2+I_{yD}^2+I_{zD}^2))\cdot(I_xI_D+\gamma(I_{xx}I_{xD}+I_{xy}I_{yD}+I_{xz}I_{zD})), \end{aligned} \tag{3.84}$$

where we have used Equation (3.83) for Ψ'. We have also used the fact that $\frac{\partial I(\mathbf{x})}{\partial u}$, $\frac{\partial I_x(\mathbf{x})}{\partial u}$, $\frac{\partial (y+v)}{\partial u}$, $\frac{\partial (z+w)}{\partial u}$ and $\frac{\partial (t+1)}{\partial u}$ are all 0 in Equations (3.65), (3.66), (3.71), (3.72) and (3.73). We can derive f_v in an analogous way:

$$\begin{aligned} f_v &= \Psi'(I_D^2 + \gamma(I_{xD}^2 + I_{yD}^2 + I_{zD}^2)) \cdot (\frac{|I(\mathbf{x}+\mathbf{w}) - I(\mathbf{x})|^2}{\partial v} + \frac{|\nabla I(\mathbf{x}+\mathbf{w}) - \nabla I(\mathbf{x})|^2}{\partial v}) \\ &= \Psi'(I_D^2 + \gamma(I_{xD}^2 + I_{yD}^2 + I_{zD}^2)) \cdot (2I_D \frac{\partial I(\mathbf{x}+\mathbf{w})}{\partial v} \\ &+ 2(I_{xD}, I_{yD}, I_{zD}) \frac{\partial (I_x(\mathbf{x}+\mathbf{w}), I_y(\mathbf{x}+\mathbf{w}), I_z(\mathbf{x}+\mathbf{w}))}{\partial v}) \\ &= \Psi'(I_D^2 + \gamma(I_{xD}^2 + I_{yD}^2 + I_{zD}^2)) \cdot (I_y I_D + \gamma(I_{yx} I_{xD} + I_{yy} I_{yD} + I_{zy} I_{zD})), \end{aligned} \tag{3.85}$$

where we have again used Equation (3.83). We have also used the fact that $\frac{\partial I(\mathbf{x})}{\partial v}$, $\frac{\partial I_x(\mathbf{x})}{\partial v}$, $\frac{\partial (x+u)}{\partial v}$, $\frac{\partial (z+w)}{\partial v}$ and $\frac{\partial (t+1)}{\partial v}$ are all 0 in Equations (3.67), (3.68), (3.75), (3.76) and (3.77). And also f_w:

$$\begin{aligned} f_w &= \Psi'(I_D^2 + \gamma(I_{xD}^2 + I_{yD}^2 + I_{zD}^2)) \cdot (\frac{|I(\mathbf{x}+\mathbf{w}) - I(\mathbf{x})|^2}{\partial w} + \frac{|\nabla I(\mathbf{x}+\mathbf{w}) - \nabla I(\mathbf{x})|^2}{\partial w}) \\ &= \Psi'(I_D^2 + \gamma(I_{xD}^2 + I_{yD}^2 + I_{zD}^2)) \cdot (2I_D \frac{\partial I(\mathbf{x}+\mathbf{w})}{\partial w} \\ &+ 2(I_{xD}, I_{yD}, I_{zD}) \frac{\partial (I_x(\mathbf{x}+\mathbf{w}), I_y(\mathbf{x}+\mathbf{w}), I_z(\mathbf{x}+\mathbf{w}))}{\partial w}) \\ &= \Psi'(I_D^2 + \gamma(I_{xD}^2 + I_{yD}^2 + I_{zD}^2)) \cdot (I_z I_D + \gamma(I_{zx} I_{xD} + I_{zy} I_{yD} + I_{zz} I_{zD})). \end{aligned} \tag{3.86}$$

We can write f_{u_x}, f_{u_y}, f_{u_z}, f_{v_x}, f_{v_y}, f_{v_z}, f_{w_x}, f_{w_y} and f_{w_z} as:

$$
\begin{aligned}
f_{u_x} &= \Psi'\left(u_x^2+u_y^2+u_z^2+v_x^2+v_y^2+v_z^2+w_x^2+w_y^2+w_z^2\right)2u_x, && (3.87)\\
f_{u_y} &= \Psi'\left(u_x^2+u_y^2+u_z^2+v_x^2+v_y^2+v_z^2+w_x^2+w_y^2+w_z^2\right)2u_y, && (3.88)\\
f_{u_z} &= \Psi'\left(u_x^2+u_y^2+u_z^2+v_x^2+v_y^2+v_z^2+w_x^2+w_y^2+w_z^2\right)2u_z, && (3.89)\\
f_{v_x} &= \Psi'\left(u_x^2+u_y^2+u_z^2+v_x^2+v_y^2+v_z^2+w_x^2+w_y^2+w_z^2\right)2v_x, && (3.90)\\
f_{v_y} &= \Psi'\left(u_x^2+u_y^2+u_z^2+v_x^2+v_y^2+v_z^2+w_x^2+w_y^2+w_z^2\right)2v_y, && (3.91)\\
f_{v_z} &= \Psi'\left(u_x^2+u_y^2+u_z^2+v_x^2+v_y^2+v_z^2+w_x^2+w_y^2+w_z^2\right)2v_z, && (3.92)\\
f_{w_x} &= \Psi'\left(u_x^2+u_y^2+u_z^2+v_x^2+v_y^2+v_z^2+w_x^2+w_y^2+w_z^2\right)2w_x, && (3.93)\\
f_{w_y} &= \Psi'\left(u_x^2+u_y^2+u_z^2+v_x^2+v_y^2+v_z^2+w_x^2+w_y^2+w_z^2\right)2w_y, && (3.94)\\
f_{w_z} &= \Psi'\left(u_x^2+u_y^2+u_z^2+v_x^2+v_y^2+v_z^2+w_x^2+w_y^2+w_z^2\right)2w_z. && (3.95)
\end{aligned}
$$

We can compute $\frac{df_{u_x}}{dx}$, $\frac{df_{u_y}}{dy}$ and $\frac{df_{u_z}}{dz}$ as:

$$
\begin{aligned}
\frac{df_{u_x}}{dx} &= \alpha u_{xx}\Psi'\left(u_x^2+u_y^2+u_z^2+v_x^2+v_y^2+v_z^2+w_x^2+w_y^2+w_z^2\right)\\
&= \alpha u_{xx}\Psi'_{Smooth}, && (3.96)\\
\frac{df_{u_y}}{dy} &= \alpha u_{yy}\Psi'\left(u_x^2+u_y^2+u_z^2+v_x^2+v_y^2+v_z^2+w_x^2+w_y^2+w_z^2\right)\\
&= \alpha u_{yy}\Psi'_{Smooth}, && (3.97)\\
\frac{df_{u_z}}{dz} &= \alpha u_{zz}\Psi'\left(u_x^2+u_y^2+u_z^2+v_x^2+v_y^2+v_z^2+w_x^2+w_y^2+w_z^2\right)\\
&= \alpha u_{zz}\Psi'_{Smooth}. && (3.98)
\end{aligned}
$$

and $\frac{df_{v_x}}{dx}$, $\frac{df_{v_y}}{dy}$ and $\frac{df_{v_z}}{dz}$ as:

$$\begin{aligned}
\frac{df_{v_x}}{dx} &= \alpha v_{xx}\Psi'\left(u_x^2+u_y^2+u_z^2+v_x^2+v_y^2+v_z^2+w_x^2+w_y^2+w_z^2\right) \\
&= \alpha v_{xx}\Psi'_{Smooth}, \qquad (3.99)\\
\frac{df_{v_y}}{dy} &= \alpha v_{yy}\Psi'\left(u_x^2+u_y^2+u_z^2+v_x^2+v_y^2+v_z^2+w_x^2+w_y^2+w_z^2\right) \\
&= \alpha v_{yy}\Psi'_{Smooth}, \qquad (3.100)\\
\frac{df_{v_z}}{dz} &= \alpha v_{zz}\Psi'\left(u_x^2+u_y^2+u_z^2+v_x^2+v_y^2+v_z^2+w_x^2+w_y^2+w_z^2\right) \\
&= \alpha v_{zz}\Psi'_{Smooth}, \qquad (3.101)
\end{aligned}$$

and $\frac{df_{w_x}}{dx}$, $\frac{df_{w_y}}{dy}$ and $\frac{df_{w_z}}{dz}$ as:

$$\begin{aligned}
\frac{df_{w_x}}{dx} &= \alpha w_{xx}\Psi'\left(u_x^2+u_y^2+u_z^2+v_x^2+v_y^2+v_z^2+w_x^2+w_y^2+w_z^2\right) \\
&= \alpha w_{xx}\Psi'_{Smooth}, \qquad (3.102)\\
\frac{df_{w_y}}{dy} &= \alpha w_{yy}\Psi'\left(u_x^2+u_y^2+u_z^2+v_x^2+v_y^2+v_z^2+w_x^2+w_y^2+w_z^2\right) \\
&= \alpha w_{yy}\Psi'_{Smooth}, \qquad (3.103)\\
\frac{df_{w_z}}{dz} &= \alpha w_{zz}\Psi'\left(u_x^2+u_y^2+u_z^2+v_x^2+v_y^2+v_z^2+w_x^2+w_y^2+w_z^2\right) \\
&= \alpha w_{zz}\Psi'_{Smooth}. \qquad (3.104)
\end{aligned}$$

The divergence operator for a vector A can be written as

$$\mathbf{DIV}(A) = \partial_x A + \partial_y A + \partial_z A, \qquad (3.105)$$

The **Div** of $\nabla_3 u$ is $u_{xx}+u_{yy}+u_{zz}$. Similarly, the **Div** of $\nabla_3 v$ is $v_{xx}+v_{yy}+v_{zz}$ and the **Div** of $\nabla_3 w$ is $w_{xx}+w_{yy}+w_{zz}$. Again using Equation (3.83) to replace Ψ_{Smooth} with Ψ'_{Smooth} we can rewrite Equations (3.96) to (3.104) as:

$$\frac{df_{u_x}}{dx}+\frac{df_{u_y}}{dy}+\frac{df_{u_z}}{dz} = \alpha\ \mathbf{Div}\ \nabla_3 u\ \Psi'\left(|\nabla_3 u|^2+|\nabla_3 v|^2+|\nabla_3 w|^2\right), \qquad (3.106)$$

$$\frac{df_{v_x}}{dx} + \frac{df_{v_y}}{dy} + \frac{df_{v_z}}{dz} = \alpha \, \mathbf{Div} \, \nabla_3 v \, \Psi' \left(|\nabla_3 u|^2 + |\nabla_3 v|^2 + |\nabla_3 w|^2 \right), \tag{3.107}$$

$$\frac{df_{w_x}}{dx} + \frac{df_{w_y}}{dy} + \frac{df_{w_z}}{dz} = \alpha \, \mathbf{Div} \, \nabla_3 w \, \Psi' \left(|\nabla_3 u|^2 + |\nabla_3 v|^2 + |\nabla_3 w|^2 \right). \tag{3.108}$$

Combining Equations (3.84) and (3.106), Equations (3.85) and (3.107) and Equations (3.86) and (3.108) and rearranging some terms gives the final Euler-Lagrange equations as in Brox et al. 2004 [7] as:

$$\Psi'(I_D^2 + \gamma(I_{xD}^2 + I_{yD}^2 + I_{zD}^2)) \cdot (I_x I_D + \gamma(I_{xx} I_{xD} + I_{xy} I_{yD} + I_{xz} I_{zD})) \\ -\alpha \, \mathbf{Div}(\Psi'(|\nabla_3 u|^2 + |\nabla_3 v|^2 + |\nabla_3 w|^2) \nabla_3 u) = 0, \tag{3.109}$$

$$\Psi'(I_D^2 + \gamma(I_{xD}^2 + I_{yD}^2 + I_{zD}^2)) \cdot (I_y I_D + \gamma(I_{yx} I_{xD} + I_{yy} I_{yD} + I_{zy} I_{zD})) \\ -\alpha \, \mathbf{Div}(\Psi'(|\nabla_3 u|^2 + |\nabla_3 v|^2 + |\nabla_3 w|^2) \nabla_3 v) = 0, \tag{3.110}$$

$$\Psi'(I_D^2 + \gamma(I_{xD}^2 + I_{yD}^2 + I_{zD}^2)) \cdot (I_z I_D + \gamma(I_{zx} I_{xD} + I_{zy} I_{yD} + I_{zz} I_{zD})) \\ -\alpha \, \mathbf{Div}(\Psi'(|\nabla_3 u|^2 + |\nabla_3 v|^2 + |\nabla_3 w|^2) \nabla_3 w) = 0. \tag{3.111}$$

3.2.4 Numerical Implemententation

Equations (3.109) to (3.111) are nonlinear in argument $\mathbf{w} = (u, v, w, 1)$. We employ fixed point iterations to deal with this nonlinearity. A hierarchical approach is combined with these fixed point iterations by a downsampling strategy. a will be used as an index to the pyramid level. Instead of the standard downsampling factor of 0.5 on each level as used in a Gaussian pyramid, Brox et al. suggested using an arbitrary factor $\eta \in (0, 1)$, which allows smoothers transitions between adjacent scales. Given

a solution at level a as $\mathbf{w}^a$, the solution for level $a+1$, $\mathbf{w}^{a+1}$ is found as:

$$\begin{aligned}
&\Psi\,'((I_D^{a+1})^2+\gamma((I_{xD}^{a+1})^2+(I_{yD}^{a+1})^2+(I_{zD}^{a+1})^2)\cdot\\
&\qquad(I_x^aI_D^{a+1}+\gamma(I_{xx}^aI_{xD}^{a+1}+I_{xy}^aI_{yD}^{a+1}+I_{xz}^aI_{zD}^{a+1}))\\
&-\alpha\ \mathbf{Div}(\Psi'(|\nabla_3u|^2+|\nabla_3v|^2+|\nabla_3w|^2)\nabla_3u)=0,
\end{aligned}\tag{3.112}$$

$$\begin{aligned}
&\Psi\,'((I_D^{a+1})^2+\gamma((I_{xD}^{a+1})^2+(I_{yD}^{a+1})^2+(I_{zD}^{a+1})^2)\cdot\\
&\qquad(I_y^aI_D^{a+1}+\gamma(I_{yx}^aI_{xD}^{a+1}+I_{yy}^aI_{yD}^{a+1}+I_{zy}^aI_{zD}^{a+1}))\\
&-\alpha\ \mathbf{Div}(\Psi'(|\nabla_3u|^2+|\nabla_3v|^2+|\nabla_3w|^2)\nabla_3v)=0,
\end{aligned}\tag{3.113}$$

$$\begin{aligned}
&\Psi\,'((I_D^{a+1})^2+\gamma((I_{xD}^{a+1})^2+(I_{yD}^{a+1})^2+(I_{zD}^{a+1})^2)\cdot\\
&\qquad(I_z^aI_D^{a+1}+\gamma(I_{zx}^aI_{xD}^{a+1}+I_{zy}^aI_{yD}^{a+1}+I_{zz}^aI_{zD}^{a+1}))\\
&-\alpha\ \mathbf{Div}(\Psi'(|\nabla_3u|^2+|\nabla_3v|^2+|\nabla_3w|^2)\nabla_3w)=0.
\end{aligned}\tag{3.114}$$

This system of equations is still nonlinear because of $\Psi\,'$ and variables of the form I_*^{a+1}. To remove this nonlinearity we use 1^{st} order Taylor series expansions:

$$I_D^{a+1} \approx I_D^a+I_x^adu^a+I_y^adv^a+I_z^adw^a, \tag{3.115}$$

$$I_{xD}^{a+1} \approx I_{xD}^a+I_{xx}^adu^a+I_{xy}^adv^a+I_{xz}^adw^a, \tag{3.116}$$

$$I_{yD}^{a+1} \approx I_{yD}^a+I_{yx}^adu^a+I_{yy}^adv^a+I_{yz}^adw^a, \tag{3.117}$$

$$I_{zD}^{a+1} \approx I_{zD}^a+I_{zx}^adu^a+I_{zy}^adv^a+I_{zz}^adw^a, \tag{3.118}$$

where $u^{a+1}=u^a+du^a$, $v^{a+1}=v^a+dv^a$ and $w^{a+1}=w^a+dw^a$. Now the unknowns u^{a+1}, v^{a+1} and w^{a+1} are now in terms of the known solution u^a, v^a and w^a and the

unknowns du^a, dv^a and dw^a. We can write:

$$\begin{aligned}(\Psi')^a_{Data} &= \Psi'((I^a_D + I^a_x du^a + I^a_y dv^a + I^a_z dw^a)^2 \\ &+ \gamma((I^a_{xD} + I^a_{xx} du^a + I^a_{xy} dv^a + I^a_{xz} dw^a)^2 + \\ &+ (I^a_{yD} + I^a_{yx} du^a + I^a_{yy} dv^a + I^a_{yz} dw^a)^2 + \\ &+ (I^a_{zD} + I^a_{zx} du^a + I^a_{zy} dv^a + I^a_{zz} dw^a)^2) \end{aligned} \tag{3.119}$$

and

$$(\Psi')^a_{Smooth} = \Psi'(|\nabla_3(u^a + du^a)|^2 + |\nabla_3(v^a + dv^a)|^2 + |\nabla_3(w^a + dw^a)|^2). \tag{3.120}$$

With these equations, Equations (3.112) to (3.114) can be rewritten as:

$$\begin{aligned} 0 &= (\Psi')^a_{Data} \cdot (I^a_x(I^a_D + I^a_x du^a + I^a_y dv^a + I^a_z dw^a)) + \gamma(\Psi')^a_{Data} \cdot \\ &\quad (I^a_{xx}(I^a_{xD} + I^a_{xx} du^a + I^a_{xy} dv^a + I^a_{xz} dw^a) \\ &+ I^a_{xy}(I^a_{yD} + I^a_{xy} du^a + I^a_{yy} dv^a + I^a_{yz} dw^a) \\ &+ I^a_{xz}(I^a_{zD} + I^a_{xz} du^a + I^a_{zy} dv^a + I^a_{zz} dw^a)) \\ &- \alpha\, \mathbf{Div}((\Psi')^a_{Smooth} \nabla_3(u^a + du^a)), \end{aligned} \tag{3.121}$$

$$\begin{aligned} 0 &= (\Psi')^a_{Data} \cdot (I^a_y(I^a_D + I^a_x du^a + I^a_y dv^a + I^a_z dw^a)) + \gamma(\Psi')^a_{Data} \cdot \\ &\quad (I^a_{yx}(I^a_{xD} + I^a_{xx} du^a + I^a_{xy} dv^a + I^a_{xz} dw^a) \\ &+ I^a_{yy}(I^a_{yD} + I^a_{xy} du^a + I^a_{yy} dv^a + I^a_{yz} dw^a) \\ &+ I^a_{yz}(I^a_{zD} + I^a_{xz} du^a + I^a_{zy} dv^a + I^a_{zz} dw^a)) \\ &- \alpha\, \mathbf{Div}((\Psi')^a_{Smooth} \nabla_3(v^a + dv^a)), \end{aligned} \tag{3.122}$$

$$
\begin{aligned}
0 &= (\Psi')^a_{Data} \cdot (I^a_z(I^a_D + I^a_x du^a + I^a_y dv^a + I^a_z dw^a)) + \gamma(\Psi')^a_{Data} \cdot \\
&\quad (I^a_{zx}(I^a_{xD} + I^a_{xx} du^a + I^a_{xy} dv^a + I^a_{xz} dw^a) \\
&+ I^a_{zy}(I^a_{yD} + I^a_{xy} du^a + I^a_{yy} dv^a + I^a_{yz} dw^a) \\
&+ I^a_{zz}(I^a_{zD} + I^a_{xz} du^a + I^a_{zy} dv^a + I^a_{zz} dw^a)) \\
&- \alpha\, \mathbf{Div}((\Psi')^a_{Smooth} \nabla_3 (w^a + dw^a)).
\end{aligned} \tag{3.123}
$$

This is still a nonlinear system of equations because of Ψ'. A second, inner, fixed point iteration is used to remove the nonlinearity in the increments. The nonlinearity in Ψ is handled because Ψ was chosen as a convex function there is always a unique minimum solution. let b denotes the inner iteration in level a. We set $du^{a,0} = 0$, $dv^{a,0} = 0$ and $dw^{a,0} = 0$ to initialize the increments and denote $du^{a,b}$, $dv^{a,b}$ and $dw^{a,b}$ as the increment values at inner iteration b. $(\Psi')^{a,b}_{Data}$ and $(\Psi')^{a,b}_{Smooth}$ are defined by Equations (3.119) and (3.120) for some a and b. We can obtain a linear system of equations in terms of $du^{a,b}$, $dv^{a,b}$ and $dw^{a,b}$:

$$
\begin{aligned}
0 &= (\Psi')^{a,b}_{Data} \cdot (I^a_x(I^a_D + I^a_x du^{a,b+1} + I^a_y dv^{a,b+1} + I^a_z dw^{a,b+1})) + \gamma(\Psi')^{a,b}_{Data} \cdot \\
&\quad (I^a_{xx}(I^a_{xD} + I^a_{xx} du^{a,b+1} + I^a_{xy} dv^{a,b+1} + I^a_{xz} dw^{a,b+1}) \\
&+ I^a_{xy}(I^a_{yD} + I^a_{xy} du^{a,b+1} + I^a_{yy} dv^{a,b+1} + I^a_{yz} dw^{a,b+1}) \\
&+ I^a_{xz}(I^a_{zD} + I^a_{xz} du^{a,b+1} + I^a_{zy} dv^{a,b+1} + I^a_{zz} dw^{a,b+1})) \\
&- \alpha\, \mathbf{Div}((\Psi')^{a,l}_{Smooth} \nabla_3 (u^a + du^{a,b+1})),
\end{aligned} \tag{3.124}
$$

$$
\begin{aligned}
0 &= (\Psi')^{a,b}_{Data} \cdot (I^a_y(I^a_D + I^a_x du^{a,b+1} + I^a_y dv^{a,b+1} + I^a_z dw^{a,b+1})) + \gamma(\Psi')^{a,b}_{Data} \cdot \\
&\quad (I^a_{yx}(I^a_{xD} + I^a_{xx} du^{a,b+1} + I^a_{xy} dv^{a,b+1} + I^a_{xz} dw^{a,b+1}) \\
&+ I^a_{yy}(I^a_{yD} + I^a_{xy} du^{a,b+1} + I^a_{yy} dv^{a,b+1} + I^a_{yz} dw^{a,b+1}) \\
&+ I^a_{yz}(I^a_{zD} + I^a_{xz} du^{a,b+1} + I^a_{zy} dv^{a,b+1} + I^a_{zz} dw^{a,b+1})) \\
&- \alpha\, \mathbf{Div}((\Psi')^{a,b}_{Smooth} \nabla_3 (v^a + dv^{a,b+1})),
\end{aligned} \tag{3.125}
$$

$$
\begin{aligned}
0 &= (\Psi')^{a,b}_{Data} \cdot (I^a_z(I^a_D + I^a_x du^{a,b+1} + I^a_y dv^{a,b+1} + I^a_z dw^{a,b+1})) + \gamma(\Psi')^{a,b}_{Data} \cdot \\
&\quad (I^a_{zx}(I^a_{xD} + I^a_{xx} du^{a,b+1} + I^a_{xy} dv^{a,b+1} + I^a_{xz} dw^{a,b+1}) \\
&+ I^a_{zy}(I^a_{yD} + I^a_{xy} du^{a,b+1} + I^a_{yy} dv^{a,b+1} + I^a_{yz} dw^{a,b+1}) \\
&+ I^a_{zz}(I^a_{zD} + I^a_{xz} du^{a,b+1} + I^a_{zy} dv^{a,b+1} + I^a_{zz} dw^{a,b+1})) \\
&- \alpha\, \mathbf{Div}((\Psi')^{a,b}_{Smooth} \nabla_3(w^a + dw^{a,b+1})). \qquad (3.126)
\end{aligned}
$$

3.2.5 3×3 Crammer's Rule

As suggested by Faisal and Barron [9] because they couldn't get Brox et al.'s SOR to work, we use Crammer's rule to solve our 3×3 equations. Suppose we have a system of three linear equations:

$$
\begin{bmatrix} A & B & C \\ D & E & F \\ G & H & I \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} J \\ K \\ L \end{bmatrix}. \qquad (3.127)
$$

In matrix format the solution of X, Y and Z is:

$$
\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} A & B & C \\ D & E & F \\ G & H & I \end{bmatrix}^{-1} \begin{bmatrix} J \\ K \\ L \end{bmatrix}, \qquad (3.128)
$$

which yields:

$$
X = \frac{\begin{vmatrix} J & B & C \\ K & E & F \\ L & H & I \end{vmatrix}}{\begin{vmatrix} A & B & C \\ D & E & F \\ G & H & I \end{vmatrix}}, \qquad (3.129)
$$

$$Y = \frac{\begin{vmatrix} A & J & C \\ D & K & F \\ G & L & I \end{vmatrix}}{\begin{vmatrix} A & B & C \\ D & E & F \\ G & H & I \end{vmatrix}} \tag{3.130}$$

and

$$Z = \frac{\begin{vmatrix} A & B & J \\ D & E & K \\ G & H & L \end{vmatrix}}{\begin{vmatrix} A & B & C \\ D & E & F \\ G & H & I \end{vmatrix}}. \tag{3.131}$$

Using cofactor expansion of 3×3 determinant, we have:

$$X = \frac{J\begin{vmatrix} E & F \\ H & I \end{vmatrix} - K\begin{vmatrix} B & C \\ H & I \end{vmatrix} + L\begin{vmatrix} B & C \\ E & F \end{vmatrix}}{A\begin{vmatrix} E & F \\ H & I \end{vmatrix} - D\begin{vmatrix} B & C \\ H & I \end{vmatrix} + G\begin{vmatrix} B & C \\ E & F \end{vmatrix}}$$

$$= \frac{J(EI - HF) - K(BI - HC) + L(BF - CE)}{A(EI - HF) - D(BI - HC) + G(BF - CE)}, \tag{3.132}$$

$$Y = \frac{-J\begin{vmatrix} D & F \\ G & I \end{vmatrix} + K\begin{vmatrix} A & C \\ G & I \end{vmatrix} - L\begin{vmatrix} A & C \\ D & F \end{vmatrix}}{B\begin{vmatrix} D & F \\ G & I \end{vmatrix} - E\begin{vmatrix} A & C \\ G & I \end{vmatrix} + H\begin{vmatrix} A & C \\ D & F \end{vmatrix}}$$

$$= \frac{J(DI - GF) - K(AI - GC) + L(AF - CD)}{B(DI - GF) - E(AI - GC) + H(AF - CD)}, \tag{3.133}$$

and

$$Z = \frac{J\begin{vmatrix} D & E \\ G & H \end{vmatrix} - K\begin{vmatrix} A & J \\ G & L \end{vmatrix} + L\begin{vmatrix} A & B \\ D & E \end{vmatrix}}{C\begin{vmatrix} D & E \\ G & H \end{vmatrix} - F\begin{vmatrix} A & B \\ G & H \end{vmatrix} + I\begin{vmatrix} A & B \\ D & E \end{vmatrix}}$$

$$= \frac{J(DH - EG) - K(AH - BG) + L(AE - DB)}{C(DH - EG) - F(AH - BG) + I(AE - BD)}. \tag{3.134}$$

For Equations (3.124), (3.125) and (3.126), we have three linear equation of the form:

$$A(du^{a,b+1}) + B(dv^{a,b+1}) + C(dw^{a,b+1}) = J, \tag{3.135}$$

$$D(du^{a,b+1}) + E(dv^{a,b+1}) + F(dw^{a,b+1}) = K, \tag{3.136}$$

$$G(du^{a,b+1}) + H(dv^{a,b+1}) + I(dw^{a,b+1}) = L. \tag{3.137}$$

Here, A through L can be written as:

$$A = (\Psi')^{a,b+1}_{Data}(I^a_x I^a_x + \gamma(I^a_{xx} I^a_{xx} + I^a_{xy} I^a_{xy} + I^a_{zy} I^a_{xz})), \tag{3.138}$$

$$B = (\Psi')^{a,b+1}_{Data}(I^a_x I^a_y + \gamma(I^a_{xx} I^a_{xy} + I^a_{xy} I^a_{yy} + I^a_{zy} I^a_{zy}), \tag{3.139}$$

$$C = (\Psi')^{a,b+1}_{Data}(I^a_x I^a_z + \gamma(I^a_{xx} I^a_{xz} + I^a_{xy} I^a_{yz} + I^a_{zy} I^a_{zz}), \tag{3.140}$$

$$D = (\Psi')^{a,b+1}_{Data}(I^a_y I^a_x + \gamma(I^a_{yx} I^a_{xx} + I^a_{yy} I^a_{xy} + I^a_{yz} I^a_{xz})), \tag{3.141}$$

$$E = (\Psi')^{a,b+1}_{Data}(I^a_y I^a_y + \gamma(I^a_{yx} I^a_{yx} + I^a_{yy} I^a_{yy} + I^a_{yz} I^a_{zy})), \tag{3.142}$$

$$F = (\Psi')^{a,b+1}_{Data}(I^a_y I^a_z + \gamma(I^a_{yx} I^a_{xz} + I^a_{yy} I^a_{yz} + I^a_{yz} I^a_{zz})), \tag{3.143}$$

$$G = (\Psi')^{a,b+1}_{Data}(I^a_z I^a_x + \gamma(I^a_{zx} I^a_{xx} + I^a_{zy} I^a_{xy} + I^a_{zz} I^a_{xz})), \tag{3.144}$$

$$H = (\Psi')^{a,b+1}_{Data}(I^a_z I^a_y + \gamma(I^a_{zx} I^a_{xy} + I^a_{zy} I^a_{yy} + I^a_{zz} I^a_{zy})), \tag{3.145}$$

$$I = (\Psi')^{a,b+1}_{Data}(I^a_z I^a_z + \gamma(I^a_{zx} I^a_{xz} + I^a_{zy} I^a_{yz} + I^a_{zz} I^a_{zz})), \tag{3.146}$$

$$J = \alpha\ \mathbf{Div}((\Psi')^{a,b}_{Smooth} \nabla_4(u^a + du^{a,b+1}))-$$

$$(\Psi')^{a,b}_{Data}(I^a_x I^a_D + \gamma(I^a_{xx} I^a_{xD} + I^a_{xy} I^a_{yD} + I^a_{zy} I^a_{zD})), \tag{3.147}$$

$$K = \alpha\ \mathbf{Div}((\Psi')^{a,b}_{Smooth} \nabla_4(v^a + dv^{a,b+1}))-$$

$$(\Psi')^{a,b}_{Data}(I^a_y I^a_D + \gamma(I^a_{yx} I^a_{xD} + I^a_{yy} I^a_{yD} + I^a_{yy} I^a_{zD})), \tag{3.148}$$

$$L = \alpha\ \mathbf{Div}((\Psi')^{a,b}_{Smooth} \nabla_4(w^a + dw^{a,b+1}))-$$

$$(\Psi')^{a,b}_{Data}(I^a_z I^a_D + \gamma(I^a_{zx} I^a_{xD} + I^a_{zy} I^a_{yD} + I^a_{zz} I^a_{zD})). \tag{3.149}$$

Since **Div** is $\frac{\partial}{\partial x} + \frac{\partial}{\partial y} + \frac{\partial}{\partial z}$ we can take Ψ' out of the parentheses in the equations for J, K and L:

$$J = \alpha\ (\Psi')^{a,b}_{Smooth} \mathbf{Div}(\nabla_4(u^a + du^{a,b+1}))-$$

$$(\Psi')^{a,b}_{Data}(I^a_x I^a_D + \gamma(I^a_{xx} I^a_{xD} + I^a_{xy} I^a_{yD} + I^a_{zy} I^a_{zD})), \tag{3.150}$$

$$K = \alpha\ (\Psi')^{a,b}_{Smooth} \mathbf{Div}(\nabla_4(v^a + dv^{a,b+1}))-$$

$$(\Psi')^{a,b}_{Data}(I^a_y I^a_D + \gamma(I^a_{yx} I^a_{xD} + I^a_{yy} I^a_{yD} + I^a_{yy} I^a_{zD})), \tag{3.151}$$

$$L = \alpha\ (\Psi')^{a,b}_{Smooth} \mathbf{Div}(\nabla_4(w^a + dw^{a,b+1}))-$$

$$(\Psi')^{a,b}_{Data}(I^a_z I^a_D + \gamma(I^a_{zx} I^a_{xD} + I^a_{zy} I^a_{yD} + I^a_{zz} I^a_{zD})). \tag{3.152}$$

3.2.6 $7 \times 7 \times 7$ Averaging

In Equations (3.150), (3.151) and (3.152), due to the **DIV** operator, we need to compute $(u^a + du^{a,b+1})_{xx}$, $(u^a + du^{a,b+1})_{yy}$, $(u^a + du^{a,b+1})_{zz}$, $(v^a + dv^{a,b+1})_{xx}$, $(v^a + dv^{a,b+1})_{yy}$, $(v^a + dv^{a,b+1})_{zz}$, $(w^a + dw^{a,b+1})_{xx}$, $(w^a + dw^{a,b+1})_{yy}$ and $(w^a + dw^{a,b+1})_{zz}$.

We can rearrange the terms as:

$$(u^a + du^{a,b+1})_{xx} + (u^a + du^{a,b+1})_{yy} + (u^a + du^{a,b+1})_{zz}$$
$$= (u^a_{xx} + u^a_{yy} + u^a_{zz}) + (du^{a,b+1}_{xx} + du^{a,b+1}_{yy} + du^{a,b+1}_{zz}), \tag{3.153}$$

$$(v^a + dv^{a,b+1})_{xx} + (v^a + dv^{a,b+1})_{yy} + (v^a + dv^{a,b+1})_{zz}$$
$$= (v^a_{xx} + v^a_{yy} + v^a_{zz}) + (dv^{a,b+1}_{xx} + dv^{a,b+1}_{yy} + dv^{a,b+1}_{zz}), \tag{3.154}$$

and

$$(w^a + dw^{a,b+1})_{xx} + (w^a + dw^{a,b+1})_{yy} + (w^a + dw^{a,b+1})_{zz}$$
$$= (w^a_{xx} + w^a_{yy} + w^a_{zz}) + (dw^{a,b+1}_{xx} + dw^{a,b+1}_{yy} + dw^{a,b+1}_{zz}). \tag{3.155}$$

We use the approximation $X_{xx} + X_{yy} + X_{zz} \approx \overline{X} - X$ for some scalar X, as suggested by Horn and Schunck [14] to express J, K and L as:

$$J = \alpha\ (\Psi\ ')^{a,b}_{Smooth}\left((\overline{u^a} - u^a) + (\overline{du^{a,b+1}} - du^{a,b+1})\right) - j, \tag{3.156}$$

where

$$j = (\Psi')^{a,b}_{Data}(I^a_x I^a_D + \gamma(I^a_{xx} I^a_{xD} + I^a_{xy} I^a_{yD} + I^a_{zy} I^a_{zD})), \tag{3.157}$$

$$K = \alpha\ (\Psi\ ')^{a,b}_{Smooth}\left((\overline{v^a} - v^a) + (\overline{dv^{a,b+1}} - dv^{a,b+1})\right) - k, \tag{3.158}$$

where

$$k = (\Psi')^{a,b}_{Data}(I^a_y I^a_D + \gamma(I^a_{yx} I^a_{xD} + I^a_{yy} I^a_{yD} + I^a_{yy} I^a_{zD})), \tag{3.159}$$

and

$$L = \alpha\ (\Psi\ ')^{a,b}_{Smooth}\left((\overline{w^a} - w^a) + (\overline{dw^{a,b+1}} - dw^{a,b+1})\right) - l, \tag{3.160}$$

where

$$l = (\Psi')^{a,b}_{Data}(I^a_z I^a_D + \gamma(I^a_{zx} I^a_{xD} + I^a_{zy} I^a_{yD} + I^a_{zz} I^a_{zD})). \tag{3.161}$$

Now our three equations become:

$$\begin{aligned} & A(du^{a,b+1}) + B(dv^{a,b+1}) + C(dw^{a,b+1}) \\ = \; & \alpha\,(\Psi')^{a,b}_{Smooth}\left(\overline{u^a} - u^a + \overline{du^{a,b+1}} - du^{a,b+1}\right) - j, \qquad (3.162) \\ & D(du^{a,b+1}) + E(dv^{a,b+1}) + F(dw^{a,b+1}) \\ = \; & \alpha\,(\Psi')^{a,b}_{Smooth}\left(\overline{v^a} - v^a + \overline{dv^{a,b+1}} - dv^{a,b+1}\right) - k, \qquad (3.163) \\ & G(du^{a,b+1}) + H(dv^{a,b+1}) + I(dw^{a,b+1}) \\ = \; & \alpha\,(\Psi')^{a,b}_{Smooth}\left(\overline{w^a} - w^a + \overline{dw^{a,b+1}} - dw^{a,b+1}\right) - l. \qquad (3.164) \end{aligned}$$

Grouping similar term together we obtain:

$$\begin{aligned} & \left(A + \alpha\,(\Psi')^{a,b}_{Smooth}\right)(du^{a,b+1}) + B(dv^{a,b+1}) + C(dw^{a,b+1}) \\ & \qquad = \alpha\,(\Psi')^{a,b}_{Smooth}\left(\overline{u^a} + \overline{du^{a,b+1}} - u^k\right) - j, \qquad (3.165) \\ & D(du^{a,b+1}) + \left(E + \alpha\,(\Psi')^{a,b}_{Smooth}\right)(dv^{a,b+1}) + F(dw^{a,b+1}) \\ & \qquad = \alpha\,(\Psi')^{a,b}_{Smooth}\left(\overline{v^a} + \overline{dv^{a,b+1}} - v^a\right) - k, \qquad (3.166) \\ & G(du^{a,b+1}) + H(dv^{a,b+1}) + \left(I + \alpha\,(\Psi')^{a,b}_{Smooth}\right)(dw^{a,b+1}) \\ & \qquad = \alpha\,(\Psi')^{a,b}_{Smooth}\left(\overline{w^a} + \overline{dw^{a,b+1}} - w^a\right) - l, \qquad (3.167) \end{aligned}$$

which is expressed in matrix form as:

$$\begin{bmatrix} \left(A+\alpha(\Psi')^{a,b}_{Smooth}\right) & B & C \\ D & \left(E+\alpha(\Psi')^{a,b}_{Smooth}\right) & F \\ G & H & \left(I+\alpha(\Psi')^{a,b}_{Smooth}\right) \end{bmatrix} \begin{bmatrix} du^{a,b+1} \\ dv^{a,b+1} \\ dw^{a,b+1} \end{bmatrix}$$

$$= \begin{bmatrix} \alpha(\Psi')^{a,b}_{Smooth}\left(\overline{u^a}+\overline{du^{a,b+1}}-u^a\right)-j \\ \alpha(\Psi')^{a,b}_{Smooth}\left(\overline{v^a}+\overline{dv^{a,b+1}}-v^a\right)-k \\ \alpha(\Psi')^{a,b}_{Smooth}\left(\overline{w^a}+\overline{dw^{a,b+1}}-w^a\right)-l \end{bmatrix}. \tag{3.168}$$

The determinant of this array is:

$$det = \left(A+\alpha(\Psi')^{a,b}_{Smooth}\right)\begin{vmatrix} \left(E+\alpha(\Psi')^{a,b}_{Smooth}\right) & F \\ H & \left(I+\alpha(\Psi')^{a,b}_{Smooth}\right) \end{vmatrix}$$

$$-D\begin{vmatrix} B & C \\ H & \left(I+\alpha(\Psi')^{a,b}_{Smooth}\right) \end{vmatrix} + G\begin{vmatrix} B & C \\ \left(E+\alpha(\Psi')^{a,b}_{Smooth}\right) & F \end{vmatrix}$$

$$= \left(A+\alpha(\Psi')^{a,b}_{Smooth}\right)\left(\left(E+\alpha(\Psi')^{a,b}_{Smooth}\right)\left(I+\alpha(\Psi')^{a,b}_{Smooth}\right)-HF\right)$$

$$-D(B\left(I+\alpha(\Psi')^{a,b}_{Smooth}\right)-HC)+G(BF-C\left(E+\alpha(\Psi')^{a,b}_{Smooth}\right))$$

$$= (A+\alpha(\Psi')^{a,b}_{Smooth})(EI+E\alpha(\Psi')^{a,b}_{Smooth}+I\alpha(\Psi')^{a,b}_{Smooth}+\alpha^2((\Psi')^{a,b}_{Smooth})^2-HF)$$

$$-D(BI+B\alpha(\Psi')^{a,b}_{Smooth}-HC)+G(BF-CE-C\alpha(\Psi')^{a,b}_{Smooth}). \tag{3.169}$$

We can solve for $du^{a,b+1}$, $dv^{a,b+1}$ and $dw^{a,b+1}$ using Crammer's rule as:

$$\begin{bmatrix} du^{a,b+1} \\ dv^{a,b+1} \\ dw^{a,b+1} \end{bmatrix} = \begin{bmatrix} \left(A+\alpha(\Psi')^{a,b}_{Smooth}\right) & B & C \\ D & \left(E+\alpha(\Psi')^{a,b}_{Smooth}\right) & F \\ G & H & \left(I+\alpha(\Psi')^{a,b}_{Smooth}\right) \end{bmatrix}^{-1} \begin{bmatrix} \alpha(\Psi')^{a,b}_{Smooth}\left(\overline{u^a}+\overline{du^{a,b+1}}-u^a\right)-j \\ \alpha(\Psi')^{a,b}_{Smooth}\left(\overline{v^a}+\overline{dv^{a,b+1}}-v^a\right)-k \\ \alpha(\Psi')^{a,b}_{Smooth}\left(\overline{w^a}+\overline{dw^{a,b+1}}-w^a\right)-l \end{bmatrix}, \tag{3.170}$$

$$du^{a,b+1} = \frac{1}{det}\begin{bmatrix} \left(\alpha(\Psi')^{a,b}_{Smooth}\left(\overline{u^a}+\overline{du^{a,b+1}}-u^a\right)-j\right) & B & C \\ \left(\alpha(\Psi')^{a,b}_{Smooth}\left(\overline{v^a}+\overline{dv^{a,b+1}}-v^a\right)-k\right) & E & F \\ \left(\alpha(\Psi')^{a,b}_{Smooth}\left(\overline{w^a}+\overline{dw^{a,b+1}}-w^a\right)-l\right) & H & I \end{bmatrix}, \tag{3.171}$$

$$dv^{a,b+1} = \frac{1}{det}\begin{bmatrix} A & \left(\alpha(\Psi')^{a,b}_{Smooth}\left(\overline{u^a}+\overline{du^{a,b+1}}-u^a\right)-j\right) & C \\ D & \left(\alpha(\Psi')^{a,b}_{Smooth}\left(\overline{v^a}+\overline{dv^{a,b+1}}-v^a\right)-k\right) & F \\ G & \left(\alpha(\Psi')^{a,b}_{Smooth}\left(\overline{w^a}+\overline{dw^{a,b+1}}-w^a\right)-l\right) & I \end{bmatrix}, \tag{3.172}$$

$$dw^{a,b+1} = \frac{1}{det}\begin{bmatrix} A & B & \left(\alpha(\Psi')^{a,b}_{Smooth}\left(\overline{u^a}+\overline{du^{a,b+1}}-u^a\right)-j\right) \\ D & E & \left(\alpha(\Psi')^{a,b}_{Smooth}\left(\overline{v^a}+\overline{dv^{a,b+1}}-v^a\right)-k\right) \\ G & H & \left(\alpha(\Psi')^{a,b}_{Smooth}\left(\overline{w^a}+\overline{dw^{a,b+1}}-w^a\right)-l\right) \end{bmatrix}. \tag{3.173}$$

Using the cofactor expansion of 3×3 determinant, we have:

$$
\begin{aligned}
& du^{a,b+1} \\
= & \frac{1}{det}\left(\alpha(\Psi')^{a,b}_{Smooth}\left(\overline{u^a}+\overline{du^{a,b+1}}-u^a\right)-j\right)\begin{vmatrix}\left(E+\alpha(\Psi')^{a,b}_{Smooth}\right) & F \\ H & \left(I+\alpha(\Psi')^{a,b}_{Smooth}\right)\end{vmatrix} \\
- & \frac{1}{det}\left(\alpha(\Psi')^{a,b}_{Smooth}\left(\overline{v^a}+\overline{dv^{a,b+1}}-v^a\right)-k\right)\begin{vmatrix} B & C \\ H & \left(I+\alpha(\Psi')^{a,b}_{Smooth}\right)\end{vmatrix} \\
+ & \frac{1}{det}\left(\alpha(\Psi')^{a,b}_{Smooth}\left(\overline{w^a}+\overline{dw^{a,b+1}}-w^a\right)-l\right)\begin{vmatrix} B & C \\ \left(E+\alpha(\Psi')^{a,b}_{Smooth}\right) & F\end{vmatrix} \\
= & \frac{1}{det}\left(\alpha(\Psi')^{a,b}_{Smooth}\left(\overline{u^a}+\overline{du^{a,b+1}}-u^a\right)-j\right)\left(\left(E+\alpha(\Psi')^{a,b}_{Smooth}\right)\left(I+\alpha(\Psi')^{a,b}_{Smooth}\right)-HF\right) \\
- & \frac{1}{det}\left(\alpha(\Psi')^{a,b}_{Smooth}\left(\overline{v^a}+\overline{dv^{a,b+1}}-v^a\right)-k\right)\left(B\left(I+\alpha(\Psi')^{a,b}_{Smooth}\right)-CH\right) \\
+ & \frac{1}{det}\left(\alpha(\Psi')^{a,b}_{Smooth}\left(\overline{w^a}+\overline{dw^{a,b+1}}-w^a\right)-l\right)\left(BF-C\left(E+\alpha(\Psi')^{a,b}_{Smooth}\right)\right),
\end{aligned}
\tag{3.174}
$$

$$
\begin{aligned}
& dv^{a,b+1} \\
= & -\frac{1}{det}\left(\alpha(\Psi')^{a,b}_{Smooth}\left(\overline{u^a}+\overline{du^{a,b+1}}-u^a\right)-j\right)\begin{vmatrix} D & F \\ G & \left(I+\alpha(\Psi')^{a,b}_{Smooth}\right)\end{vmatrix} \\
+ & \frac{1}{det}\left(\alpha(\Psi')^{a,b}_{Smooth}\left(\overline{v^a}+\overline{dv^{a,b+1}}-v^a\right)-k\right)\begin{vmatrix}\left(A+\alpha(\Psi')^{a,b}_{Smooth}\right) & C \\ G & \left(I+\alpha(\Psi')^{a,b}_{Smooth}\right)\end{vmatrix} \\
- & \frac{1}{det}\left(\alpha(\Psi')^{a,b}_{Smooth}\left(\overline{w^a}+\overline{dw^{a,b+1}}-w^a\right)-l\right)\begin{vmatrix}\left(A+\alpha(\Psi')^{a,b}_{Smooth}\right) & C \\ D & F\end{vmatrix}. \\
= & \frac{1}{det}\left(\alpha(\Psi')^{a,b}_{Smooth}\left(\overline{u^a}+\overline{du^{a,b+1}}-u^a\right)-j\right)\left(D\left(I+\alpha(\Psi')^{a,b}_{Smooth}\right)-FG\right) \\
- & \frac{1}{det}\left(\alpha(\Psi')^{a,b}_{Smooth}\left(\overline{v^a}+\overline{dv^{a,b+1}}-v^a\right)-k\right)\left(\left(A+\alpha(\Psi')^{a,b}_{Smooth}\right)\left(I+\alpha(\Psi')^{a,b}_{Smooth}\right)-CG\right) \\
+ & \frac{1}{det}\left(\alpha(\Psi')^{a,b}_{Smooth}\left(\overline{w^a}+\overline{dw^{a,b+1}}-w^a\right)-l\right)\left(F\left(A+\alpha(\Psi')^{a,b}_{Smooth}\right)-CD\right)
\end{aligned}
\tag{3.175}
$$

and

$$
\begin{aligned}
& dw^{a,b+1} \\
= & \frac{1}{det}\left(\alpha(\Psi')^{a,b}_{Smooth}\left(\overline{u^a}+\overline{du^{a,b+1}}-u^a\right)-j\right)\begin{vmatrix} D & \left(E+\alpha(\Psi')^{a,b}_{Smooth}\right) \\ G & H \end{vmatrix} \\
- & \frac{1}{det}\left(\alpha(\Psi')^{a,b}_{Smooth}\left(\overline{v^a}+\overline{dv^{a,b+1}}-v^a\right)-k\right)\begin{vmatrix} \left(A+\alpha(\Psi')^{a,b}_{Smooth}\right) & B \\ G & H \end{vmatrix} \\
+ & \frac{1}{det}\left(\alpha(\Psi')^{a,b}_{Smooth}\left(\overline{w^a}+\overline{dw^{a,b+1}}-w^a\right)-l\right)\begin{vmatrix} \left(A+\alpha(\Psi')^{a,b}_{Smooth}\right) & B \\ D & \left(E+\alpha(\Psi')^{a,b}_{Smooth}\right) \end{vmatrix} \\
= & \frac{1}{det}\left(\alpha(\Psi')^{a,b}_{Smooth}\left(\overline{u^a}+\overline{du^{a,b+1}}-u^a\right)-j\right)\left(DH-G\left(E+\alpha(\Psi')^{a,b}_{Smooth}\right)\right) \\
- & \frac{1}{det}\left(\alpha(\Psi')^{a,b}_{Smooth}\left(\overline{v^a}+\overline{dv^{a,b+1}}-v^a\right)-k\right)\left(\left(A+\alpha(\Psi')^{a,b}_{Smooth}\right)H-BG\right) \\
+ & \frac{1}{det}\left(\alpha(\Psi')^{a,b}_{Smooth}\left(\overline{w^a}+\overline{dw^{a,b+1}}-w^a\right)-l\right)\left(\left(A+\alpha(\Psi')^{a,b}_{Smooth}\right)\left(E+\alpha(\Psi')^{a,b}_{Smooth}\right)-BD\right).
\end{aligned}
\tag{3.176}
$$

Initially, we start with $du^{a,b+1}=0$, $dv^{a,b+1}=0$ and $dw^{a,b+1}=0$. Then we solve for the next set of $(du^{a,b+1}, dv^{a,b+1}, dw^{a,b+1})$ by using these three equations, while computing the $\overline{du^{a,b+1}}$, $\overline{dv^{a,b+1}}$, $\overline{dw^{a,b+1}}$, $\overline{u^a}$, $\overline{v^a}$, $\overline{w^a}$, by taking the average of a $7\times7\times7$ neighborhood around each pixel.

3.2.7 Algorithm

We give the pseudo code for our 3D algorithm below.

Initialization: $u=0$; $v=0$; $w=0$.

Smooth original images with σ.

Build image at the top of the pyramid(the coarsest level) using η as the reduction factor.

for k =number of pyramid levels to 1 **do**

Compute derivatives and differences using two images.

$du1=0$; $du2=0$; $dv1=0$; $dv2=0$; $dw1=0$; $dw2=0$.

for $a = 1$ to *first inner fixed point iterations* **do**
  calculate $(\Psi 1)'_{Data}$ using $du1$, $dv1$, $dw1$.
  calculate $(\Psi 1)'_{Smooth}$ using $du1$, $dv1$, $dw1$.
  for $b = 1$ to *second inner fixed point iterations* **do**
    solve for $du2$, $dv2$, $dw2$
    if norm(($du1$-$du2$) + ($dv1$-$dv2$) + ($dw1$-$dw2$))<TOL **then**
      break.
    else
      $du1$=$du2$; $dv1$=$dv2$; $dw1$=$dw2$;
    end if
  end for
end for
$u = u + du1$; $v = v + dv1$; $w = w + dw1$.
if $k! = 0$ **then**
  build images at the next pyramid level(finer level)
  project down u, v, w to the next level
  perform warping at the next level using u, v, w
end if
end for
output velocity.
perform error calculation.

The norm(x) is the Euclidean length of vector x and TOL is the threshold value for convergence, we use 10^{-3}.

3.2.8 Spatial Differentiation

We use 4 point central differences to compute spatial derivatices I_x, I_y and I_z. We use the kernel $(0.0866, -0.6666, 0.0, 0.6666, -0.0866)$. Second order derivatives are

computed using the same kernel on the first order derivatives. Thus I_{xz} is computed by applying the kernel in the z dimension of the matrix holding the I_x data.n In MATLAB, data is reflected at the matrix boundaries to yield a same dimensioned derivative matrix. Of course, derivatives near the boundaries will be erronous. It is assumade that second order derivatives are symmetric, i.e. I_{xz} and I_{zx} are the same. To compute spatio-temporal derivatives we apply this kernel on the original data at the 2 frames and then apply simple pixel differences in time on those results. Thus I_{xt} is computed by applying this kernel on $I(t)$ and $I(t+1)$ to get $I_x(t)$ and $I_x(t+1)$ and then computing I_{xt} as $I_x(t+1) - I_x(t)$.

3.2.9 Temporal Differentiation

Now we discuss the method we use to compute the temporal derivative of the images. We compare the computed result with the correct derivatives we generated in Chapter 4. Before calculating derivatives, all input images are smoothed with a Gaussian filter with standard deviation $\sigma = 1.3$. The derivatives defined in Equations (3.46) to (3.58) depend on $I(x,y,z,t)$ and $I(x+u,y+v,z+w,t+1)$. Let $I(\mathbf{x}) = I(x,y,z,t)$ be the left image and $I(\mathbf{x\_r}) = I(x,y,z,t+1)$ be the right image. At each level of the pyramid, we need to compute the displacement $\mathbf{w} = (du, dv, dw)$ from the left image to right image. At the top of the pyramid (the coarsest level) we set $\mathbf{w^0} = 0$ so that

$$\begin{aligned} I_D^0 &= I(\mathbf{x} + \mathbf{w^0}) - \mathbf{I(x)} \\ &= I(x+0, y+0, z+0, t+1) - I(x,y,z,t) \\ &= I(\mathbf{x\_r}) - I(\mathbf{x}). \end{aligned} \tag{3.177}$$

This is the difference between the left image and the right image. In the image processing nomenclature this is called a 2 point difference and is known to be a poor approximation to a temporal derivative. When we go down to the next finer level in the pyramid, $\mathbf{w}$ will be initialized as the solution from the previous coarser level

(after being re-scaled and re-sampled appropriately). We first reverse warp the right image toward the left image using **w**. As **w** becomes more accurate, I_D will approach 0 and the right image will be more and more similar to the left one.

3.2.10 3D Inverse Warping

Given a flow field for an image $I(t)$, $\vec{v}(t) = (u(t), v(t))$, and the next image $I(t+1)$ in the sequence we can compute the first image using inverse warping. That is, for floating point image locations $(i + u(t), j + v(t))$ one uses cubic interpolation on the grayvalues at the 4 neighbouring integer image locations to compute the grayvalue at (i, j). This ensures each pixel of the interpolated image gets a grayvalue (if $(i + u(t), j + v(t))$ is outside the image boundaries the interpoplated value is set to 0). Inverse warping can be done with **interp3** in MATLAB.

3.2.11 Projecting Velocities Between Adjacent Levels

One other thing we should mention is that the size (including the width, the height and the depth) of volumes at different pyramid levels are not the same. Therefore, we need some re-scaling functional to project the velocities down from a coarser level to a finer level. We use trilinear interpolation to do this job. In MATLAB, this can also be done using function **interp3**.

Trilinear interpolation is a method for spatial interpolation on a 3-dimensional regular grid. It works over a volume of discrete datasets. Let $\mathbf{v^{i+1}}$ be the velocity volume at a coarser level $i+1$ and $\mathbf{v^i}$ be the velocity volume at its adjacent finer level i, since the rescaling factor is known, for each point (x_n, y_n, z_n) in $\mathbf{v^i}$, we can compute the coordinates of its corresponding point (x, y, z) in $\mathbf{v^{i+1}}$. The corresponding point we get is usually a intermediate one, which means x,y and z are quite likely to be non-integer. Therefore, we use trilinear interpolation to predict the value at this

intermediate point with eight values of the integer coordinate positions surrounding the point.

Let x_s, y_s and z_s be differences between x, y, z and the smaller coordinates related, we have:

$$x_s = x - \lfloor x \rfloor, \tag{3.178}$$

$$y_s = y - \lfloor y \rfloor, \tag{3.179}$$

$$z_s = z - \lfloor z \rfloor. \tag{3.180}$$

We first interpolate along z direction as:

$$i_1 = \mathbf{v^{i+1}}(\lfloor x \rfloor, \lfloor y \rfloor, \lfloor z \rfloor)(1 - z_s) + \mathbf{v^{i+1}}(\lfloor x \rfloor, \lfloor y \rfloor, \lceil z \rceil) z_s, \tag{3.181}$$

$$i_2 = \mathbf{v^{i+1}}(\lfloor x \rfloor, \lceil y \rceil, \lfloor z \rfloor)(1 - z_s) + \mathbf{v^{i+1}}(\lfloor x \rfloor, \lceil y \rceil, \lceil z \rceil) z_s, \tag{3.182}$$

$$j_1 = \mathbf{v^{i+1}}(\lceil x \rceil, \lfloor y \rfloor, \lfloor z \rfloor)(1 - z_s) + \mathbf{v^{i+1}}(\lceil x \rceil, \lfloor y \rfloor, \lceil z \rceil) z_s, \tag{3.183}$$

$$j_2 = \mathbf{v^{i+1}}(\lceil x \rceil, \lceil y \rceil, \lfloor z \rfloor)(1 - z_s) + \mathbf{v^{i+1}}(\lceil x \rceil, \lceil y \rceil, \lceil z \rceil) z_s. \tag{3.184}$$

Then interpolate along y direction as:

$$k_1 = i_1(1 - y_s) + i_2 y_s, \tag{3.185}$$

$$k_2 = j_1(1 - y_s) + j_2 y_s. \tag{3.186}$$

Lastly, we interpolate these values along the x direction as:

$$\mathbf{v^i}(x_n, y_n, z_n) = k_1(1 - x_s) + k_2 x_s. \tag{3.187}$$

Here, $\lfloor x \rfloor$ refers to the floor function result of x, which gives the next smallest integer to x while $\lceil x \rceil$ refers to the ceiling function result of x, which gives the next largest integer to x.

3.3 3D Uras, Girosi, Verri and Torre algorithm

The optical flow algorithm presented by Uras, Girosi, Verri and Torre [28] is a 2^{nd}-order technique based on the optical flow (motion) constraint equation and the 3 equations resulting by differentiating this equation with respect to x, y and z:

$$\begin{aligned} I_x u + I_y v + I_z w + I_t &= 0, \\ I_{xx} u + I_{xy} v + I_{xz} w + I_{tx} &= 0, \\ I_{yx} u + I_{yy} v + I_{yz} w + I_{ty} &= 0, \\ I_{zx} u + I_{zy} v + I_{zz} w + I_{tz} &= 0. \end{aligned} \tag{3.188}$$

or we can rewrite the equations in matrix form:

$$\begin{bmatrix} I_x & I_y & I_z \\ I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} -I_t \\ -I_{tx} \\ -I_{ty} \\ -I_{tz} \end{bmatrix}. \tag{3.189}$$

If the matrix is invertible (the aperture problem can be overcome in 3D) the solution of this matrix is:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} I_x & I_y & I_z \\ I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}^{-1} \begin{bmatrix} -I_t \\ -I_{tx} \\ -I_{ty} \\ -I_{tz} \end{bmatrix}. \tag{3.190}$$

We use these equations to verify the correctness of the derivatives we generated in Chapter 4. That is, if we have correct derivatives, then this 3D optical flow should be perfect.

Chapter 4

Experimental Technique

This chapter describes the method used to generate of our synthetic dataset, the method to compute image derivatives, the angular error matrix used for quantitative analysis flow field and gives an introduction of the gated MRI cardiac datasets.

4.1 Generation of 3D Sinusoid Volume Datasets

The main advantage of synthetic data is that the motion fields can be calculated precisely. Therefore, we can perform quantitative evaluation on the algorithm's output. In total, we generate three sinusoid volume datasets: **sinL**, **sinR** and **sin**. Each volume contains 31 slices of 256 × 256 data (unsigned shorts in the range [0-4095], i.e. 12 bits). The **sin** data is a combination of **sinL** and **sinR** which have different velocities in its left part and right part.

The formula for generating **sinL** and **sinR** is:

$$sin(\mathbf{k_1 x} + \omega_1 t) + sin(\mathbf{k_2 x} + \omega_2 t) + sin(\mathbf{k_3 x} + \omega_3 t), \tag{4.1}$$

where $\mathbf{k_i} = (k_{ix}, k_{iy}, k_{iz})$ are the spatial frequencies, $\mathbf{x} = (x, y, z)$ are 3D spatial

Dimension	**sinL**	**sinR**
x	3	-3
y	2	-2
z	1	-1

Table 4.1: Speeds for the **sinL** *and* **sinR** *data.*

coordinates, ω_i is the temporal frequency and t is the temporal coordinate. Table 4.1 shows the motion speeds of both the **sinL** data and the **sinR** data. Figure 4.1 shows three middle slices in the x, y and z dimensions of the 9^{th} volume of **sinL**, Figure 4.2 shows the correct flow field of the 9^{th} volume of **sinL** with downsampling rates 8 in the z dimension and 32 in the x and y dimension with a scale factor 10. Figure 4.3 shows three middle slices in the x, y and z dimensions of the 9^{th} volume of **sinR**, Figure 4.4 shows the correct flow field of the 9^{th} volume of **sinR** with downsampling rates 8 in the z dimension and 32 in x and y dimension with a scale factor 10. To handle the different axes system (between postscript and X11) we transform the original velocities u, v, w to u_1, v_1, w_1 using:

$$u_1 = v, \tag{4.2}$$

$$v_1 = -u, \tag{4.3}$$

$$w_1 = -w. \tag{4.4}$$

This transformation is for visual purposes only and no effect on the analysis. All flow fields for the sinusoidal datasets in this thesis are shifted in this way before displaying.

The algorithm to make the **sin** dataset (from the **sinL** and **sinR** datasets) is:

Initialization: $offset_x = 120$; $offset_y = 120$; $offset_z = 10$.
Initialization: $add_x = 0$; $add_y = 0$; $add_z = 0$.
for $v = 1$ to number of volumes **do**
  for $k = 1$ to 31 **do**

for $i = 1$ to 256 **do**
for $j = 1$ to 256 **do**
if $i > (offset_x + add_x)$ and $j > (offset_y + add_y)$ and $k > (offset_z + add_z)$
then
sin$[v]$ = **sinR**$[v]$;
else
sin$[v]$ = **sinL**$[V]$;
end if
end for
end for
end for
$add_x = add_x + u; add_y = add_y + v; add_z = add_z + w.$
end forwhere u,v and w are velocities for **sinR**.

The **sin** data is actually a combination of **sinL** and **sinR**, the correct flow field and derivatives of **sin** are generated by combining the corresponding data from **sinL** and **sinR**. This will result in some errors in the computed flow at the border where two sinusoidal datasets meet because it is difficult to compute the derivatives accurately near the border. Figure 4.5 shows three middle slices in the x, y and z dimensions of the 9^{th} volume of **sin**. Figure 4.6 shows the correct flow field of the 9^{th} volume of **sin** with downsampling rates 8 in the z dimension and 32 in x and y dimensions with a scale factor 10.

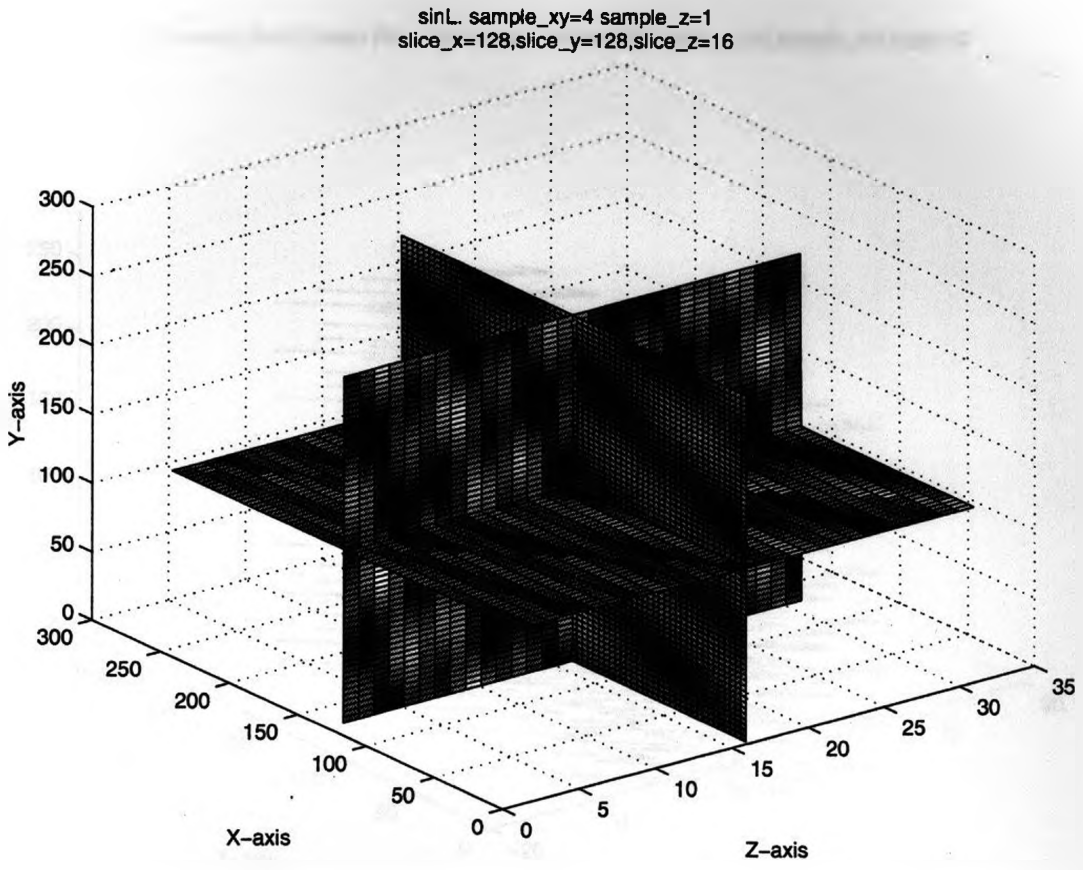


Figure 4.1: Middle slices (the 128^{th} *slice in* x *and* y *dimensions and the* 15^{th} *slice in* z *dimension) of the 3D* **sinL** *data volume 9 with a sampling rate of* 4 *in the* x *and* y *dimensions and* 1 *in the* z *dimension.*

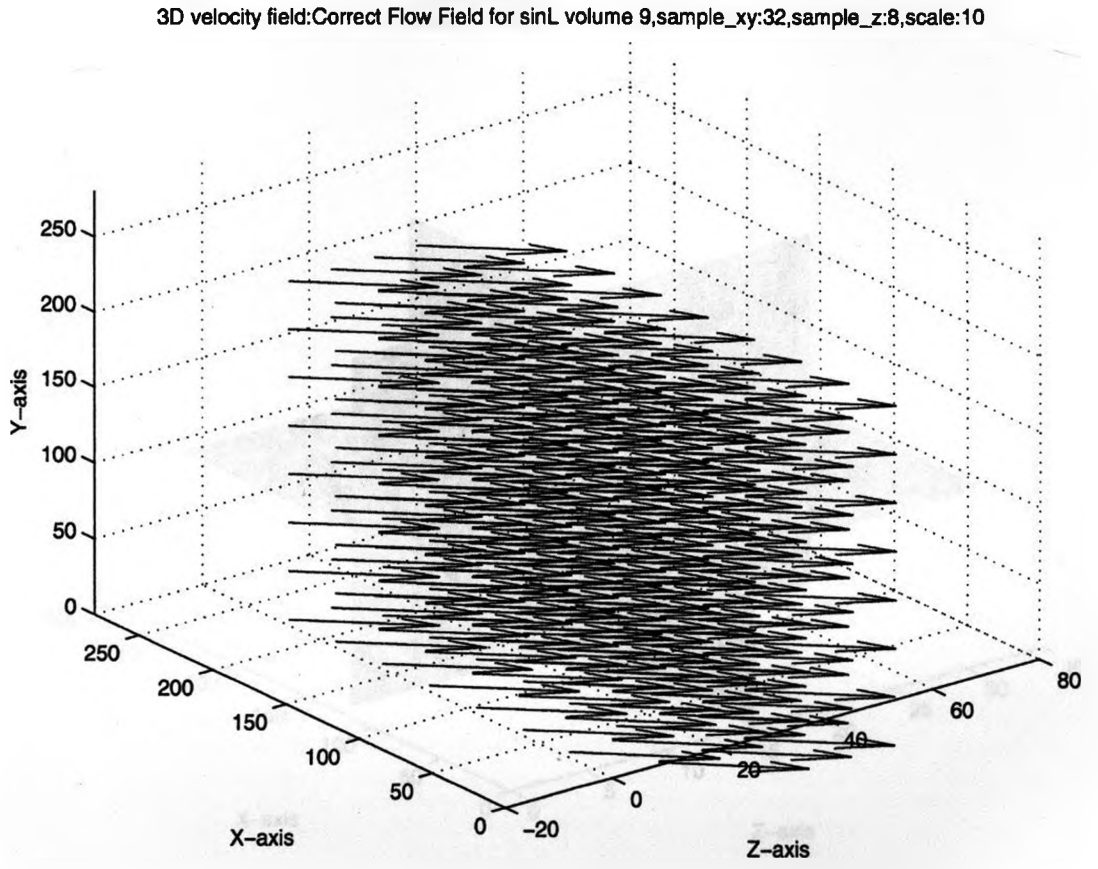


Figure 4.2: The correct flow field of the 9^{th} *volume of* **sinL** *with a downsampling rate* 8 *in the* z *dimension and* 32 *in the* x *and* y *dimensions with a scale factor of*10.

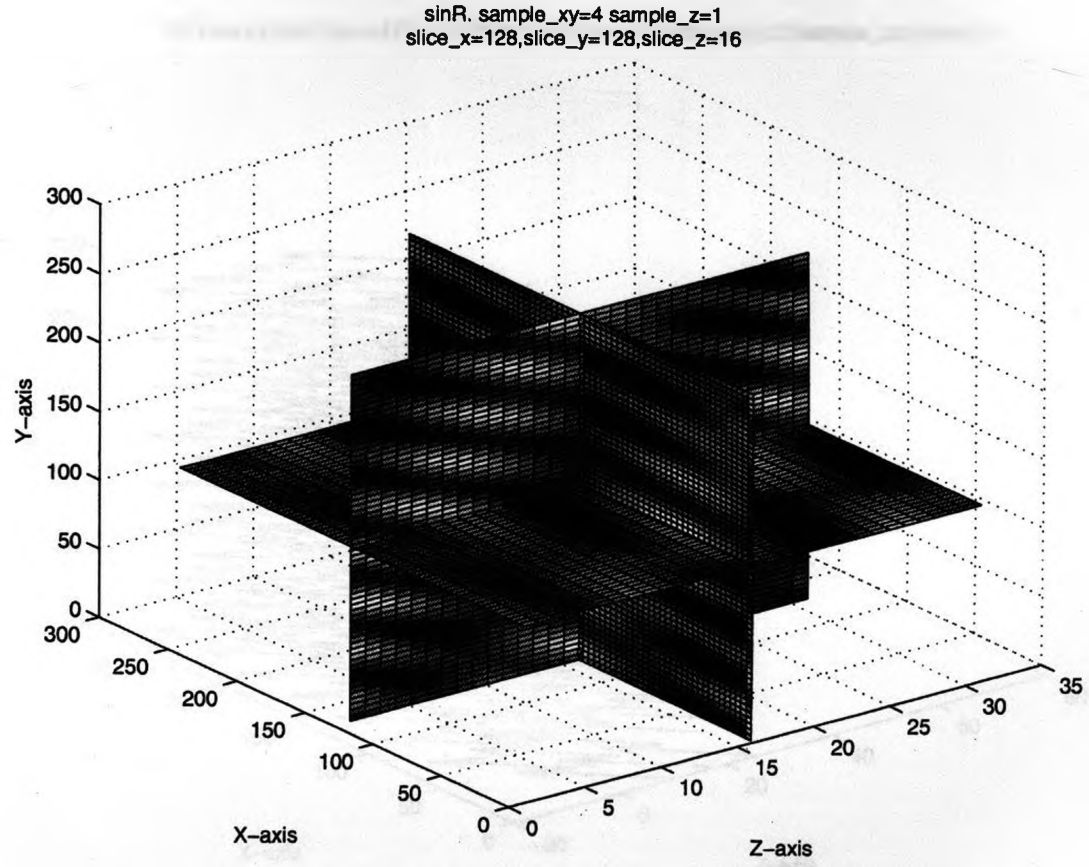


Figure 4.3: Middle slices (the 128^{th} *slice in the* x *and* y *dimensions and the* 15^{th} *slice in* z *dimension) of the 3D* **sinR** *data volume* 9 *with a sampling rate* 4 *in the* x *and* y *dimensions and* 1 *in the* z *dimension.*

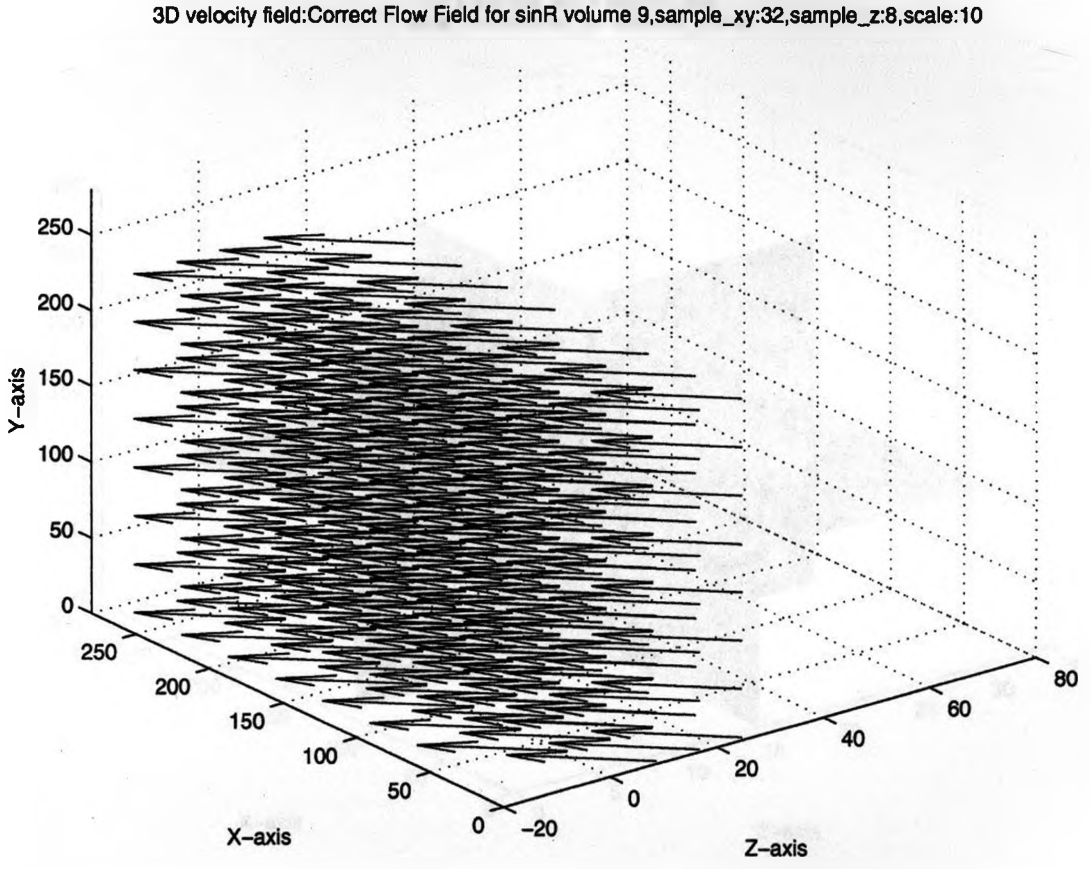


Figure 4.4: The correct flow field of the 9^{th} *volume of* **sinR** *with a sampling rate of* 8 *in the* z *dimension and* 32 *in the* x *and* y *dimensions with a scale factor of* 10.

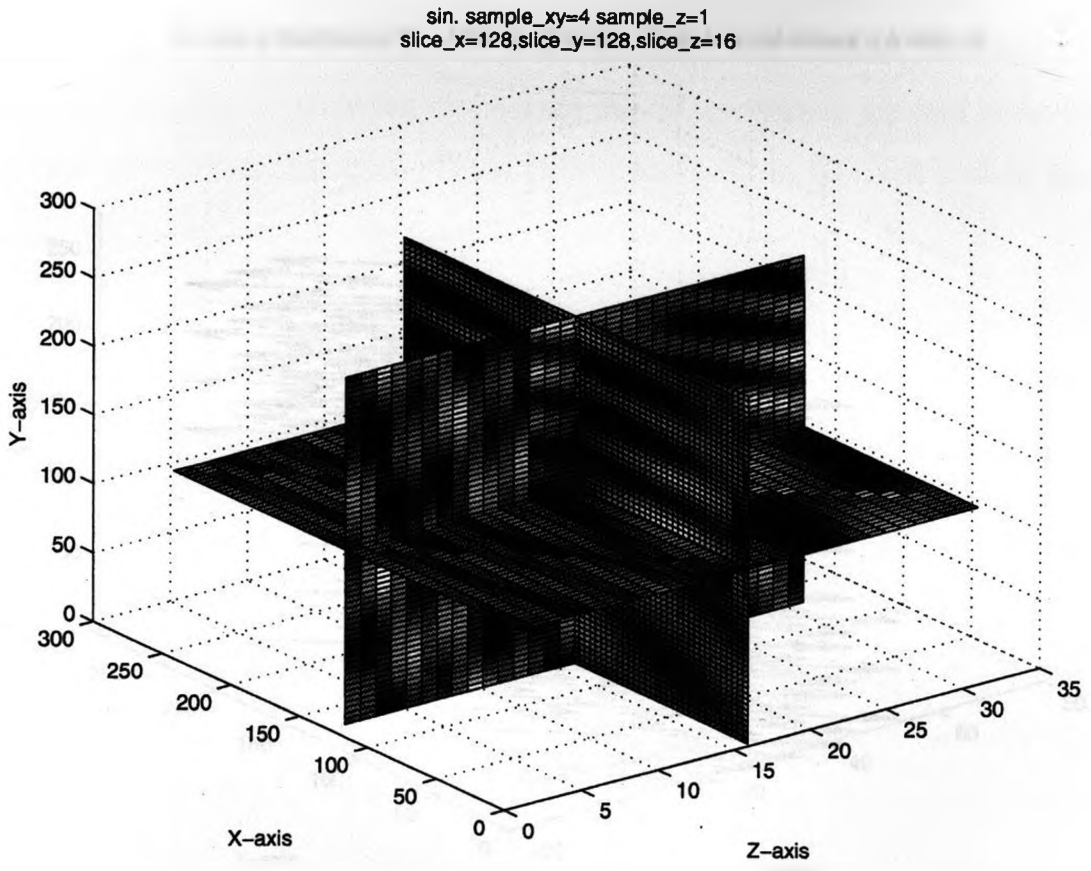


Figure 4.5: Middle slices (the 128^{th} slice in the x and y dimensions and the 15^{th} slice in the z dimension) of 3D **sin** *data volume 9 with a sampling rate of 4 in the x and y dimensions and 1 in the z dimension.*

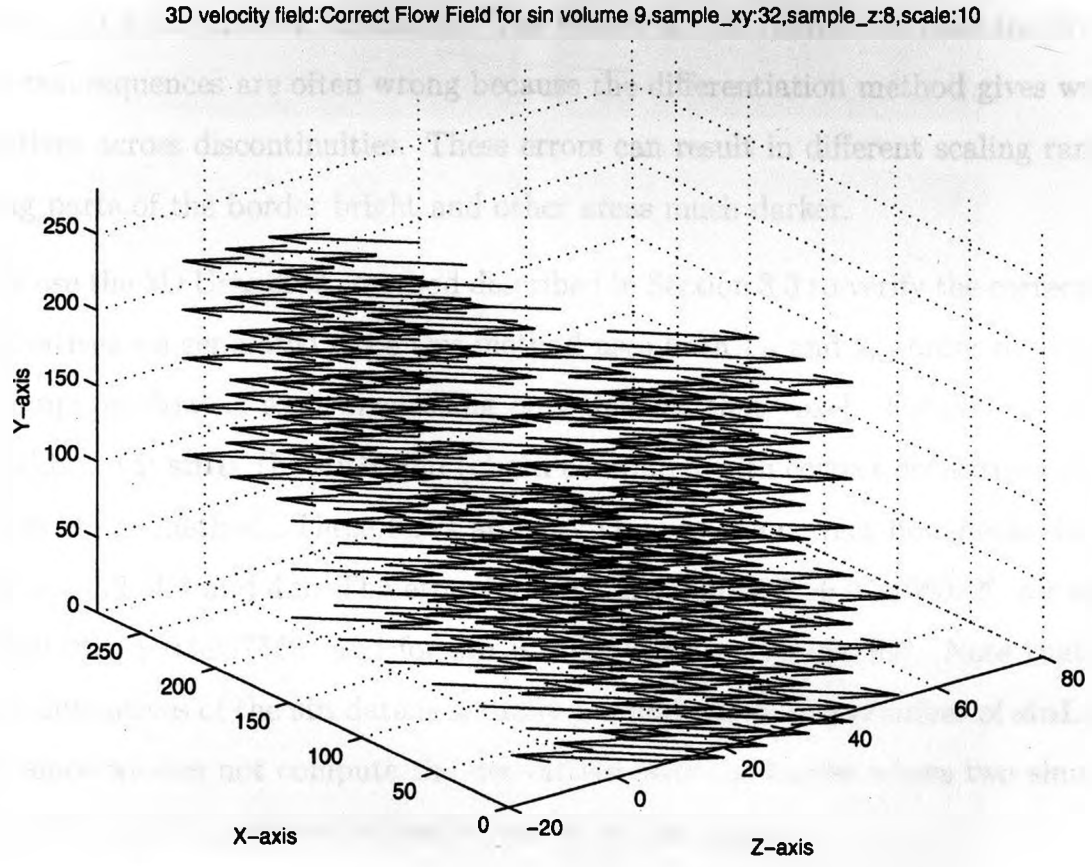
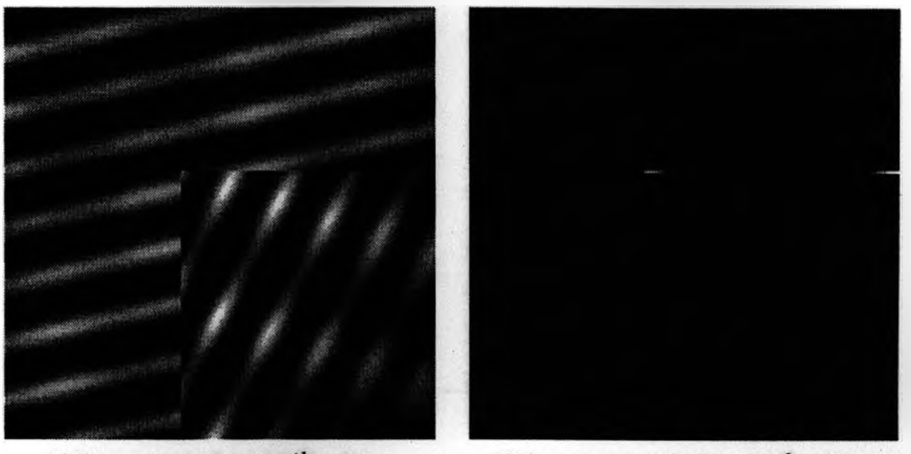


Figure 4.6: The correct flow field of the 9^{th} *volume of* **sin** *with a sampling rate of* 8 *in the* z *dimension and* 32 *in the* x *and* y *dimensions and a scale factor of* 10.

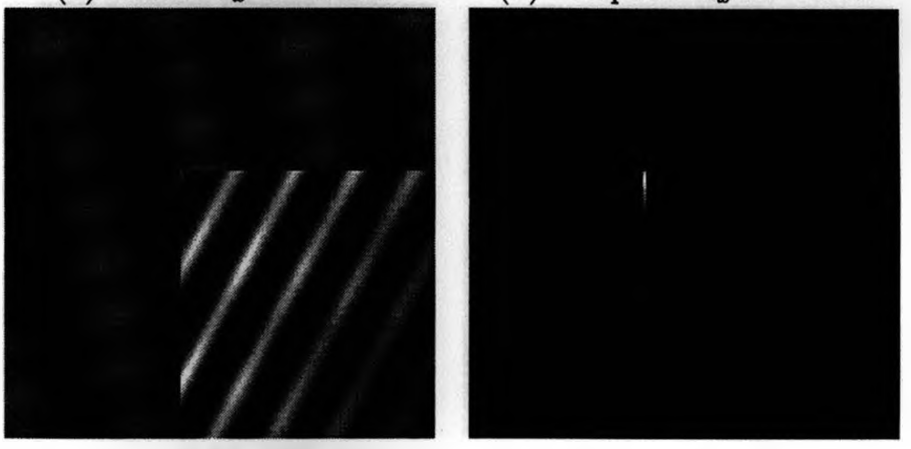
4.2 Correct and Computed Spatial Derivatives

Figures 4.7, 4.8 and 4.9 show the 15^{th} slice of the correct and computed derivatives of the 9^{th} volume for the **sin** dataset. The computed derivatives are generated by the differentiation method we described in Section 3.2.8. We can tell from these images that the correct derivatives and computed derivatives have almost the same patterns but with different intensities. The reason is that derivatives near the border of the two sequences are often wrong because the differentiation method gives wrong derivatives across discontinuities. These errors can result in different scaling ranges, making parts of the border bright and other areas much darker.

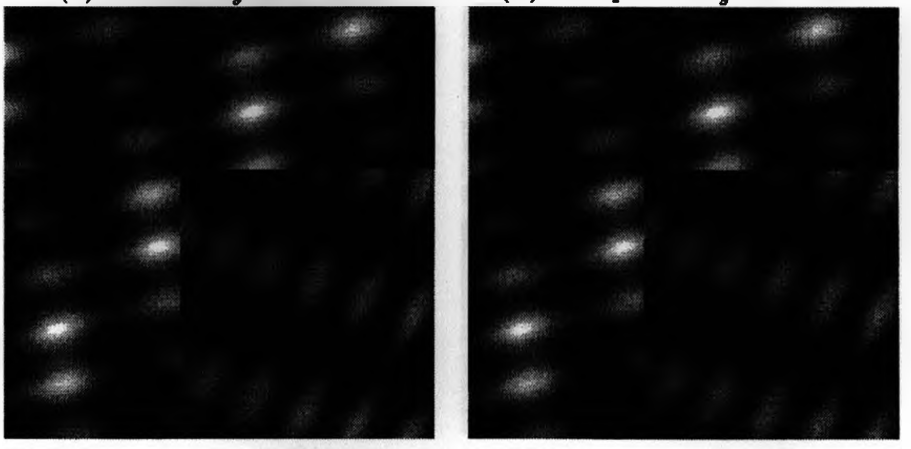
We use the 3D Uras et al. method described in Section 3.3 to verify the correctness of derivatives we generated since this method uses both 1_{st} and 2_{nd} order derivatives and computes flow at each pixel using only data at that voxel. Figure 4.10 shows flow fields of (a) **sinL**, (b) **sinR** and (c) **sin** computed with correct derivatives by the 3D Uras et al. method. These flows are very close to the correct flow fields showed in Figures 4.2, 4.4 and 4.6. The error for **sinL** is $0.0001597° \pm 0.1166072°$, for **sinR** is $0.0001187° \pm 0.0937559°$ and for **sin** is $2.0564527° \pm 17.4464889°$. Note that the correct derivatives of the **sin** data is actually a combination of derivatives of **sinL** and **sinR**, since we can not compute the derivatives near the border where two sinusoid data meet, we get erroneous derivative values in this region.



(a) correct I_x 15^{th} slice (b) computed I_x 15^{th} slice

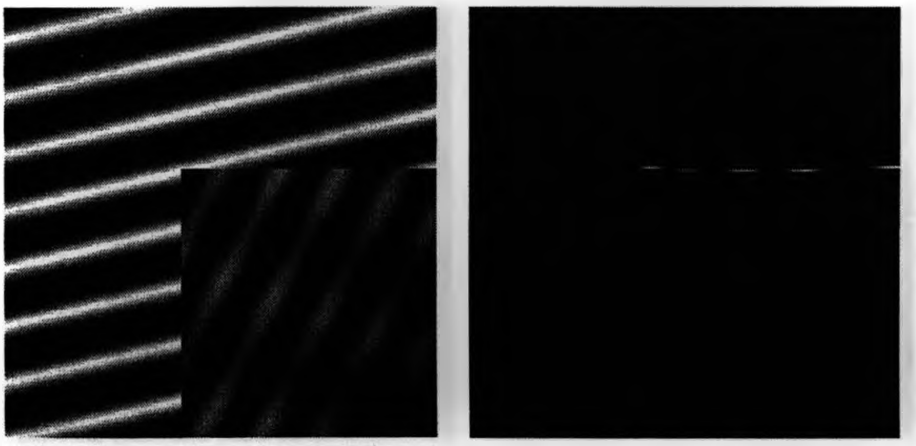


(c) correct I_y 15^{th} slice (d) computed I_y 15^{th} slice

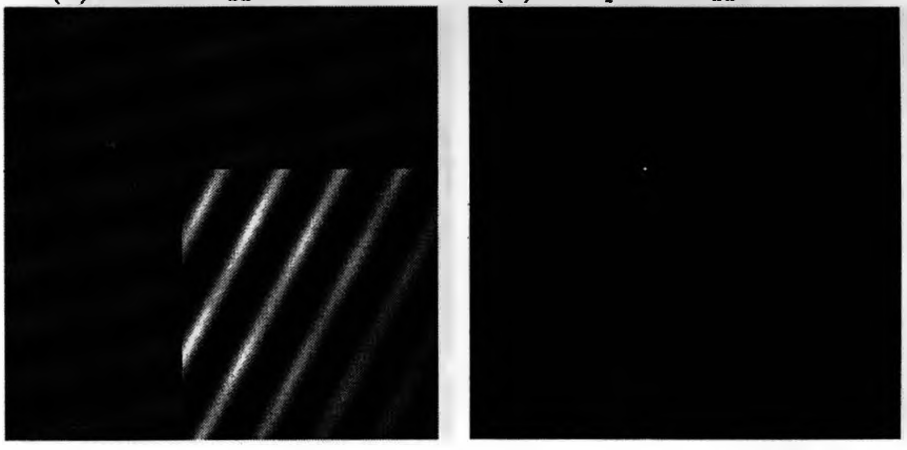


(e) correct I_z 15^{th} slice (f) computed I_z 15^{th} slice

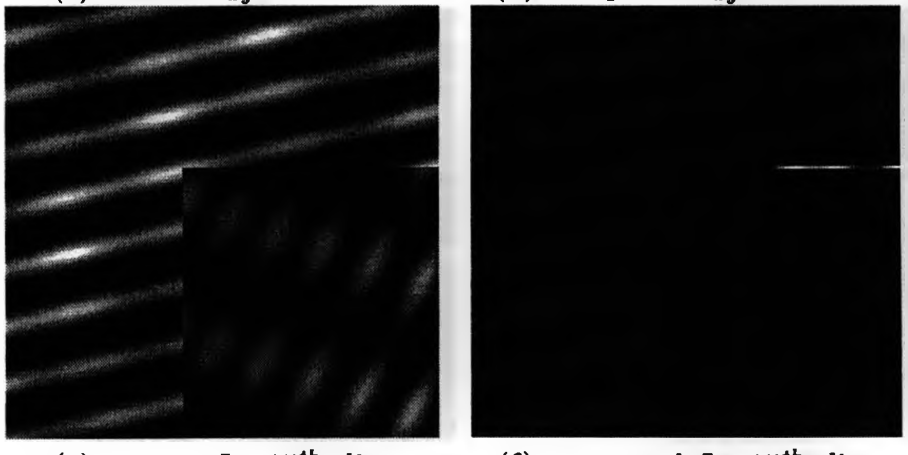
Figure 4.7: Spatial derivatives: (a) the correct and (b) the computed I_x, (c) the correct and (d) the computed I_y and (e) the correct and (f) the computed I_z for the 15^{th} slice of the 9^{th} volume for the 3D **sin** *data.*



(a) correct I_{xx} 15^{th} slice (b) computed I_{xx} 15^{th} slice

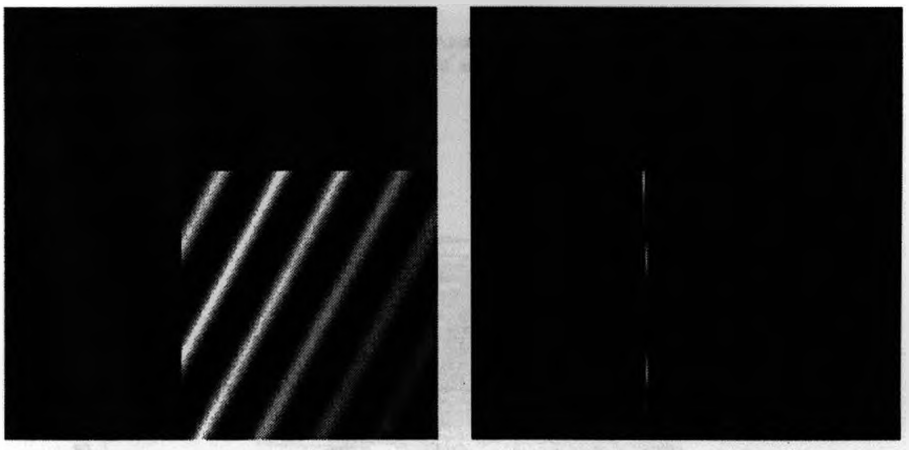


(c) correct I_{xy} 15^{th} slice (d) computed I_{xy} 15^{th} slice



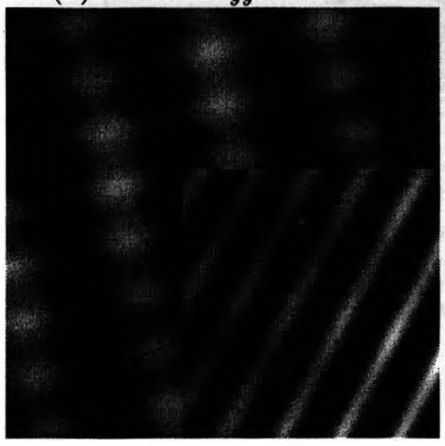
(e) correct I_{xz} 15^{th} slice (f) computed I_{xz} 15^{th} slice

Figure 4.8: Spatial derivatives: (a) the correct and (b) the computed I_{xx}, (c) the correct and (d) the computed I_{xy} and (e) the correct and (f) the computed I_{xz} for the 15^{th} slice of the 9^{th} volume for the 3D **sin** *data.*

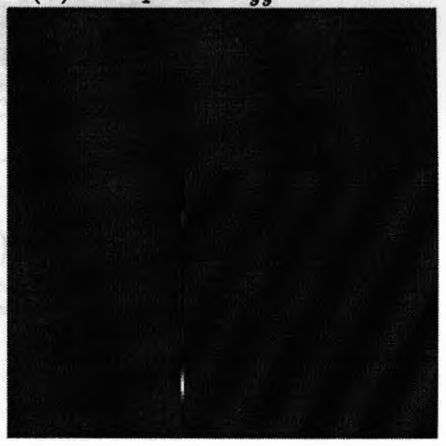


(a) correct I_{yy} 15^{th} slice

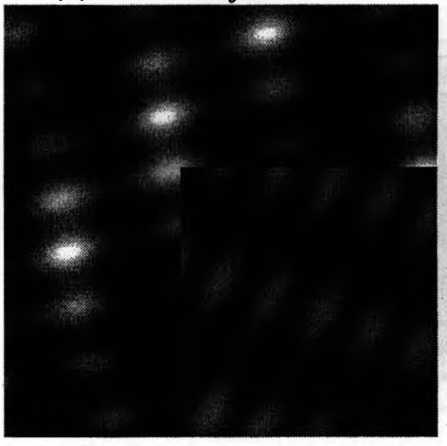
(b) computed I_{yy} 15^{th} slice



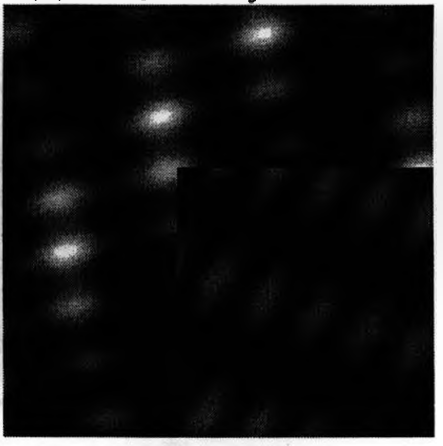
(c) correct I_{yz} 15^{th} slice



(d) computed I_{yz} 15^{th} slice



(e) correct I_{zz} 15^{th} slice



(f) computed I_{zz} 15^{th} slice

Figure 4.9: Spatial derivatives: (a) the correct and (b) the computed I_{yy}, (c) the correct and (d) the computed I_{yz} and (e) the correct and (f) the computed I_{zz} for the 15^{th} slice of the 9^{th} volume for the 3D **sin** *data.*

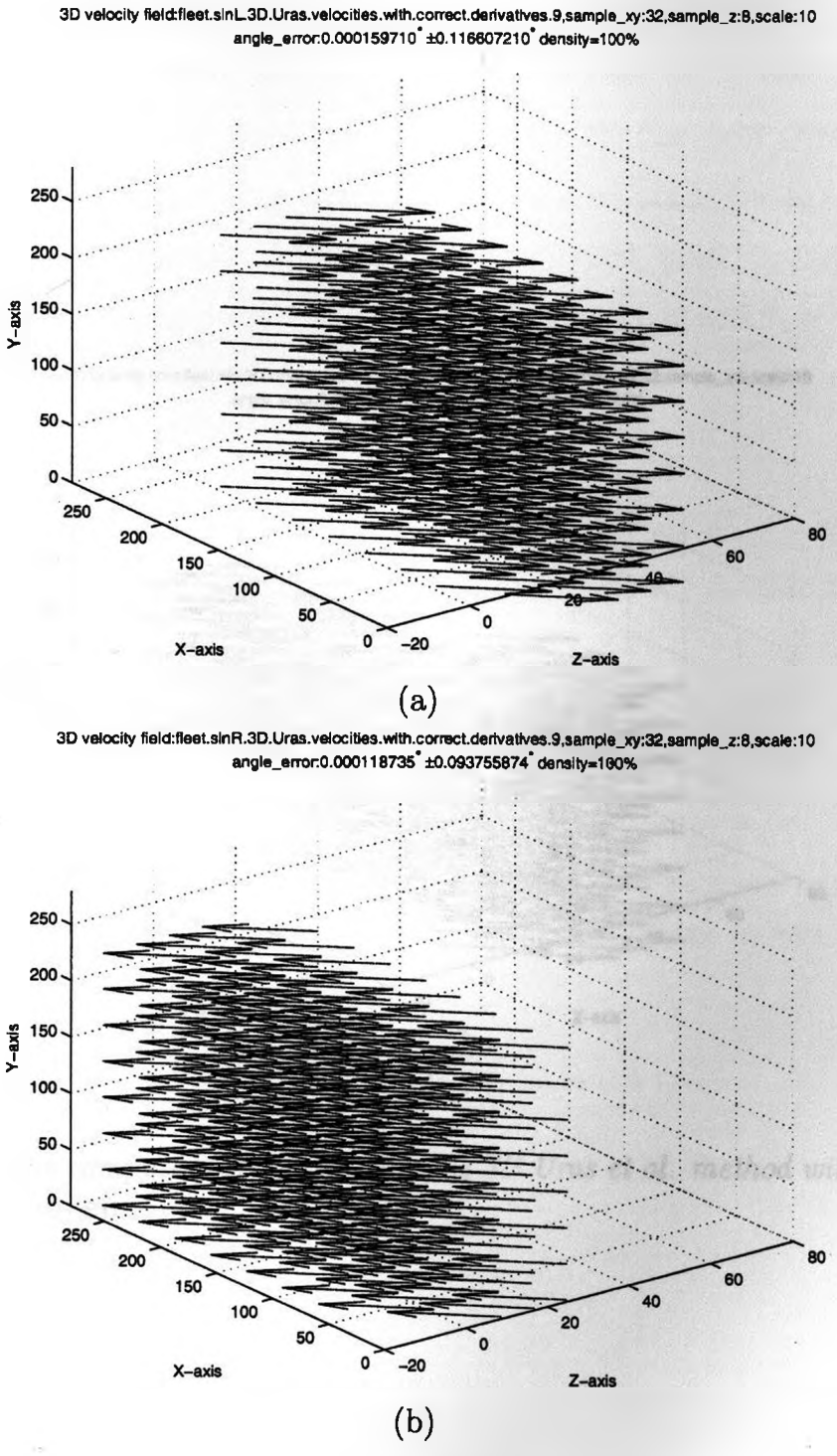
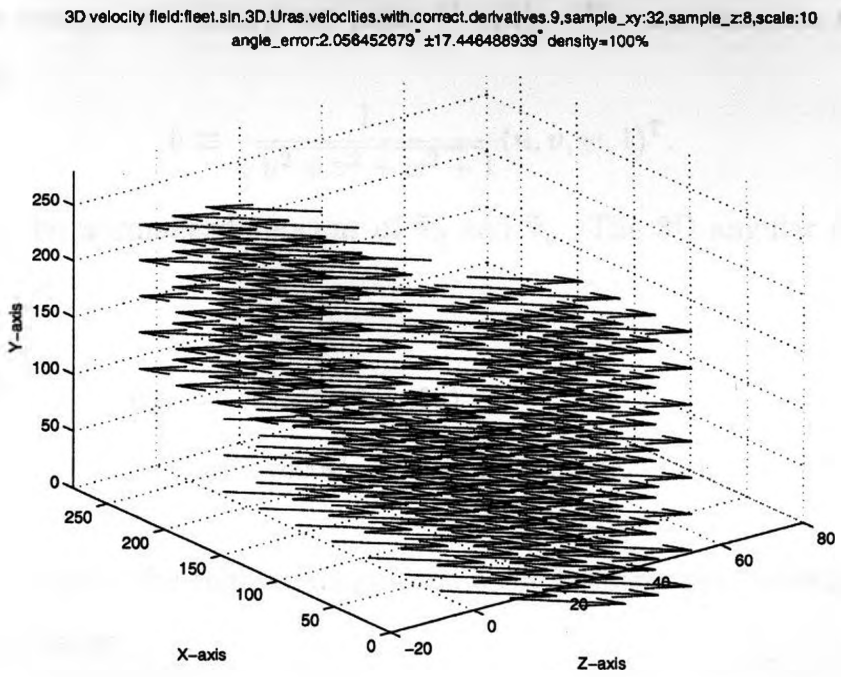


Figure 4.10: The computed flow field using the 3D Uras et al. method with the correct derivatives for (a) **sinL** *and (b)* **sinR**.



(c)

Figure 4.10: The computed flow field using the 3D Uras et al. method with the correct derivatives for (c) **sin**.

4.3 3D Error Measurement

We use the average Fleet 4D angular error measurement [6] to measure the error between the correct 3D flow field and the computed 3D flow field. Velocity can be written as displacement per time unit as in $\tilde{\mathbf{v}} = (u, v, w)$ pixels/frame or as a space-time direction vector $(u, v, w, 1)$ in units of (pixel, pixel, pixel, frame). Let $\tilde{\mathbf{v}}_\mathbf{c} = (u_c, v_c, w_c, 1)$ represent the correct velocity at the pixel (i, j, k) and $\tilde{\mathbf{v}}_\mathbf{e} = (u_e, v_e, w_e, 1)$ represent the computed velocity at point (i, j, k). We can compute a normalized velocity using:

$$\hat{v} \equiv \frac{1}{\sqrt{u^2 + v^2 + w^2 + 1}}(u, v, w, 1)^T. \tag{4.5}$$

Let $\hat{\mathbf{v}}_\mathbf{c}$ and $\hat{\mathbf{v}}_\mathbf{e}$ be normalized versions of $\tilde{\mathbf{v}}_\mathbf{c}$ and $\tilde{\mathbf{v}}_\mathbf{e}$. The 3D angular error between the $\tilde{\mathbf{v}}_\mathbf{c}$ and $\tilde{\mathbf{v}}_\mathbf{e}$ is:

$$\begin{aligned} \psi_E &= \arccos(\hat{\mathbf{v}}_\mathbf{c} \cdot \hat{\mathbf{v}}_\mathbf{e}) \\ &= \arccos(\hat{u}_c \cdot \hat{u}_e + \hat{v}_c \cdot \hat{v}_e + \hat{w}_c \cdot \hat{w}_e). \end{aligned} \tag{4.6}$$

We can also compute the relative magnitude error between the correct flow and the computed flow using:

$$\text{relative\_magnitude}_E = \frac{\|\vec{v}_c - \vec{v}_e\|}{\|\vec{v}_c\|}, \tag{4.7}$$

and the direction error as:

$$\text{direction}_E = \arccos(\frac{\vec{v}_c}{\|\vec{v}_c\|} \cdot \frac{\vec{v}_e}{\|\vec{v}_e\|}). \tag{4.8}$$

4.4 Gated MRI Cardiac Datasets

Nowadays, it is possible to acquire good gated MRI (Magnetic Resonance Imagery) data of a human beating heart. The 3D motions of the heart can provide useful

information for physician to detect heart disease. However, the measurement of 3D velocities of a beating heart is still challenging [5].

The gated MRI data we use was generated by the Robarts Research Institute at the University of Western Ontario. Each set of this data contains 20 volumes of 3D data for one synchronized heart beat, in our experiment, we test two dataset, **10phase** and **5phase**, both with volume size $256 \times 256 \times 75$ and voxel intensities in range $[0-4095]$. The heart motion is much more complex than our synthetic sinusoid datasets. A beating human heart is not rigid and deformable, and its motion is discontinuous in space and time: different chambers in the heart are contracting/expanding at different times and the heart as a whole undergoes a twisting motion as it beats. The word "gated" refers to the way the data is collected: 1 or a few slices of each volume set are acquired at the same instance in a cardiac cycle. A patient lies in an MRI machine and holds his breath for approximately 42 second intervals to acquire each set of slices. This data acquisition method relies on the patient not moving or breathing during the the acquisition (this minimizes heart motion caused by a moving diaphragm) [5]. Figure 4.11 shows the (a) 20^{th}, (b) 35^{th}, (c) 40^{th} and (d) 55^{th} slices of the 9^{th} volume of 10phase, and the (e) 20^{th}, (f) 35^{th}, (g) 40^{th} and (h) 55^{th} slices of the 9^{th} volume of 5phase. It is know that this are some breathing problems with the 5phase data (the diaphragm is moving) while the 10phase data has no such problems.

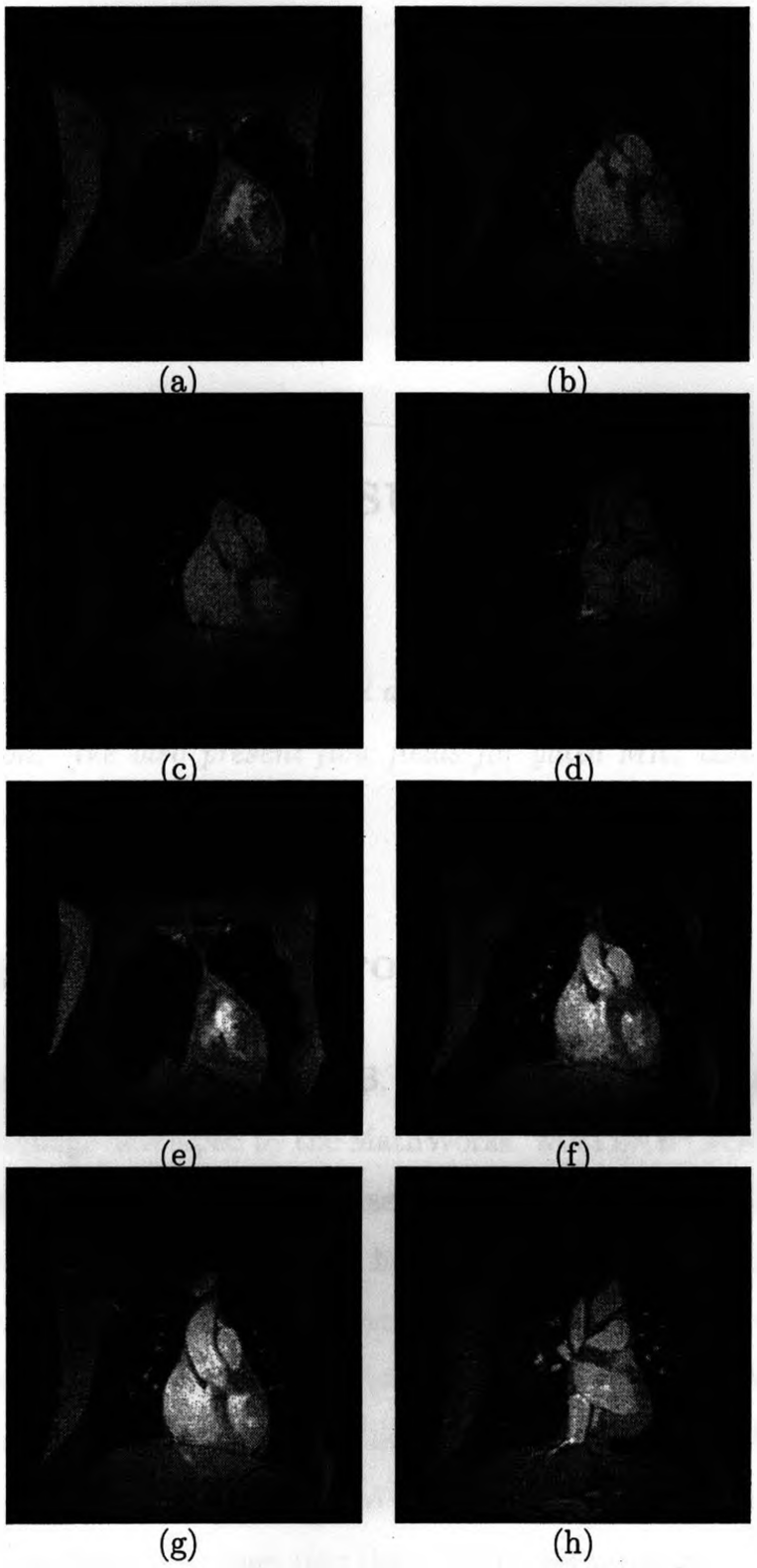


Figure 4.11: MRI cardiac data: (a)-(d) the 20^{th}, the 35^{th}, the 40^{th} and the 55^{th} slices of the 9^{th} volume of the 10phase MRI dataset, (e)-(h) the 20^{th}, the 35^{th}, the 40^{th} and the 55^{th} slices of the 9^{th} volume of the 5phase MRI dataset.

Chapter 5

Experimental Result

We test our algorithm on three sinusoidal datasets: **sinL**, **sinR** *and* **sin** *allow quantitative evaluation. We also present flow fields for gated MRI cardiac datasets for qualitative evaluation.*

5.1 Programming Environment

We implement our algorithm in MATLAB, a numerical computing environment and programming language developed by the MathWorks. MATLAB (**Mat**rix **Lab**oratory) is a software package that provides the user with high performance numerical computation and visualization capabilities. It has more than 50 sets of built-in functions, available via toolboxes. These toolboxes are collections of functions written for specialized applications, for example, image processing and graphics. We use the IPT (Image Processing Toolbox) for the work described here. MATLAB has an interface for programming languages C, JAVA and Fortran. This allows users to integrate functions written in these languages into their MATLAB program. Another thing we should mention is that matrices (vectors are special cases of matrices and a scalar is an array of size 1×1) are at the heart of MATLAB, since all data in MATLAB are

stored in matrices [12]. MATLAB is optimized for array operations, i.e. allows vectorized code to be written. According to results in Gonzalez, Woods and Eddins [24], for a simple application that computes sin results at each point of a certain matrix, the un-vectorized function takes more than 34 times as long as the vectorized code to compute the same result. Vectorized MATLAB code (although interpretated) is as fast as compile C code. Therefore, in our programming, we vectorized our code as much as possible to get the highest efficiency. Sand and Teller [26] provided MATLAB code for the 2D Brox et al. method [7] on the MathWorks file sharing website. We used part of their code and but most of our code is new.

5.2 3D Sinusoid Data Result

We use three sinusoid data sets: **sinL**, **sinR** and **sin** for our quantitative error analysis.

5.2.1 Error Comparison

Tables 5.1, 5.2 and 5.3 show the comparison between 3D Horn and Schunck and 3D Brox et al's algorithm (more results for the Brox et al.'s algorithm with different parameters will be reported in following sections.) for **sinL**, **sinR** and **sin** respectively. AAE. is the abbreviation for Average Angular Error and STD. is the abbreviation for Standard Deviation. We implemented both the original 3D Horn and Schunck and a hierarchical 3D Horn and Schunck. The latter uses a coarse-to-fine warping strategy that is the same as the Brox et al. method and performs 3D Horn and Schunck algorithm at each level of the pyramid with a warping step at each level. Another thing we should mention here is that the derivatives are computed using the method specified in Section 3.2.8 and Section 3.2.9, which means I_t is not a temporal derivative but a difference. Poor differentiation is the main reason for the poor results

Technique	AAE.	STD.
3D Horn and Schunck ($\alpha = 100.0$)	22.89°	9.08°
3D Horn and Schunck ($\alpha = 100.0$,10-levels pyramid)	10.09°	6.52°
3D Brox et al.($\alpha = 100.0$,$\gamma = 100.0$,10-levels pyramid)	2.06°	2.56°
3D Brox et al.($\alpha = 100.0$,$\gamma = 0.0$,10-levels pyramid)	0.65°	1.10°

Table 5.1: The 3D angular errors and standard deviations for flow fields with 100% *density computed by 3D Horn and Schunck* ($\alpha = 100.0$, 100 *iterations), hierarchical 3D Horn and Schunck* ($\alpha = 100.0$, 10 *levels of pyramid,* 100 *iterations), and 3D Brox et al's algorithm* ($\alpha = 100.0$,$\gamma = 100.0$ *and* $\gamma = 0.0$, *10-levels pyramid, outer iteration*= 1, *inner iteration*= 100 *), all with* $3 \times 3 \times 3$ *median filter, for the* 9^{th} *volume of* **sinL**.

Technique	AAE.	STD.
3D Horn and Schunck ($\alpha = 100.0$)	23.26°	12.56°
3D Horn and Schunck ($\alpha = 100.0$,10-levels pyramid)	9.90°	6.87°
3D Brox et al.($\alpha = 100.0$,$\gamma = 100.0$,10-levels pyramid)	1.99°	3.55°
3D Brox et al.($\alpha = 100.0$,$\gamma = 0.0$,10-levels pyramid)	0.75°	1.56°

Table 5.2: The 3D angular errors and standard deviations for flow fields with 100% *density computed by 3D Horn and Schunck* ($\alpha = 100.0$, 100 *iterations), hierarchical 3D Horn and Schunck* ($\alpha = 100.0$, 10 *levels of pyramid,* 100 *iterations), and 3D Brox et al's algorithm* ($\alpha = 100.0$,$\gamma = 100.0$ *and* $\gamma = 0.0$, *10-levels pyramid, outer iteration*= 1, *inner iteration*= 100 *), all with* $3 \times 3 \times 3$ *median filter, for the* 9^{th} *volume of* **sinR**.

for 3D Horn and Schunck. Table 5.4 shows the Horn and Schunck results computed with correct derivatives of these three data set. We can see these results are much more accurate than those computed using the Brox et al.'s differentiation. For all three datasets, hierarchical 3D Horn and Schunck gives much better results than the original 3D Horn and Schunck. 3D Brox et al.'s results ($\alpha = 100$, $\gamma = 100$ and 10 levels of pyramid) surpasses them both. When the gradient parameter γ is turned off even better results can be obtained (see below).

5.2.2 The Motion Constraint Versus The Gradient Constraint

As showed in Section 5.2.1, for all three datasets, we get better results when the gradient constraint is turned off. To determine the reason for this, we performed experiments with the correct derivatives we generated. Tables 5.5, 5.6 and 5.7 show

Technique	AAE.	STD.
3D Horn and Schunck ($\alpha = 100.0$)	32.32°	24.49°
3D Horn and Schunck ($\alpha = 100.0$,10-levels pyramid)	22.67°	29.00°
3D Brox et al.($\alpha = 100.0$,$\gamma = 100.0$,10-levels pyramid)	16.20°	29.96°
3D Brox et al.($\alpha = 100.0$,$\gamma = 0.0$,10-levels pyramid)	13.11°	28.71°

Table 5.3: The 3D angular errors and standard deviations for flow fields with 100% *density computed by 3D Horn and Schunck ($\alpha = 100.0$, 100 iterations), hierarchical 3D Horn and Schunck ($\alpha = 100.0$, 10 levels of pyramid, 100 iterations), and 3D Brox et al's algorithm ($\alpha = 100.0$,$\gamma = 100.0$ and $\gamma = 0.0$, 10-levels pyramid, outer iteration= 1, inner iteration= 100), all with $3 \times 3 \times 3$ median filter, for the 9^{th} volume of* **sin**.

Data	AAE.	STD.
sinL	0.24°	0.25°
sinR	0.78°	0.41°
sin	12.95°	24.49°

Table 5.4: The 3D angular errors and standard deviations for flow field with 100% *density of 3D Horn and Schunck ($\alpha = 10.0$, iteration= 300) computed using correct derivatives.*

results for Brox et al algorithm with one level pyramid (because we only have correct derivatives for the bottom level) for **sinL**, **sinR** and **sin** respectively. The term "temporal derivative" means that we use both the correct spatial and temporal derivatives while "temporal difference" means we use the correct spatial derivatives but with the difference between the left and right images as Brox et al. did. We also tried to add Gaussian noise to the correct derivatives. The STD column gives the standard deviation of the Gaussian noise added (when the intensity range of the image is changed to $[0, 1]$). In all these cases, the algorithm returns better result when the gradient constraint is turned on. Thus we believe our program is implemented correctly. The results in Tables 5.5, 5.6 and 5.7 indicates that poor temporal differentiation is the main cause of errors. It is quite likely the reason we get worse results when the gradient constraint is turned on is because of the use of I_{xt}, I_{yt} and I_{zt}, errors accumulate when there is a large number of pyramid levels, and differences as derivatives is not a good approximation.

I_t, I_{xt},I_{yt},I_{zt}	Gaussian STD.	α	γ	AAE.	STD.
temporal derivative	0.0	100.0	100.0	0.0002°	0.1138°
temporal derivative	0.0	100.0	0.0	0.6604°	0.3478°
temporal derivative	0.1	100.0	100.0	5.6249°	1.7654°
temporal derivative	0.1	100.0	0.0	10.3639°	2.4744°
temporal difference	0.0	100.0	100.0	12.6533°	4.5595°
temporal difference	0.0	100.0	0.0	13.8256°	4.9218°

Table 5.5: Results for the 9^{th} volume of **sinL** *computed with correct derivatives.*

I_t, I_{xt},I_{yt},I_{zt}	Gaussian STD	α	γ	AAE.	STD.
temporal derivative	0.0	100.0	100.0	0.0000°	0.0000°
temporal derivative	0.0	100.0	0.0	1.2923°	0.3373°
temporal derivative	0.1	100.0	100.0	11.9526°	2.1515°
temporal derivative	0.1	100.0	0.0	18.8333°	1.9264°
temporal difference	0.0	100.0	100.0	13.5081°	4.2302°
temporal difference	0.0	100.0	0.0	15.9578°	4.9214°

Table 5.6: Results for the 9^{th} volume of **sinR** *computed with correct derivatives.*

I_t, I_{xt},I_{yt},I_{zt}	Gaussian STD	α	γ	AAE.	STD.
temporal derivative	0.0	100.0	100.0	6.7241°	20.5102°
temporal derivative	0.0	100.0	0.0	17.2953°	22.3916°
temporal derivative	0.1	100.0	100.0	27.6497°	21.7041°
temporal derivative	0.1	100.0	0.0	31.3198°	21.2707°
temporal difference	0.0	100.0	100.0	26.4230°	25.6826°
temporal difference	0.0	100.0	0.0	28.7888°	23.6694°

Table 5.7: Results for the 9^{th} volume of **sin** *computed with correct derivatives.*

5.2.3 Inner and Outer Iterations and Convergence

Another experiment we performed with the generated derivatives was to see the effect of the inner and outer iteration on the result. As we discussed in Section 3.2.4, the outer iterations update Ψ'_{data} and Ψ'_{smooth}, which are defined by Equations (3.119) and (3.120). The inner iterations update the increments in velocities du, dv and dw. The robust estimation $\Psi(s^2)$ is defined as $\sqrt{s^2+\epsilon}(\epsilon = 0.001)$. Then the derivative $\Psi'(s^2)$ is given as:

$$\Psi'(s^2) = \frac{1}{2\sqrt{s^2+\epsilon}}. \tag{5.1}$$

We define parts of the motion and gradient constaint terms, $eqID$, $eqIx$, $eqIy$ and $eqIz$, as:

$$eqID = (I_D + I_x du + I_y dv + I_z dw)^2, \tag{5.2}$$

$$eqIx = (I_{xD} + I_{xx} du + I_{xy} dv + I_{xz} dw)^2, \tag{5.3}$$

$$eqIy = (I_{yD} + I_{yx} du + I_{yy} dv + I_{yz} dw)^2, \tag{5.4}$$

$$eqIz = (I_{zD} + I_{zx} du + I_{zy} dv + I_{zz} dw)^2. \tag{5.5}$$

Then, Equation (3.119) can be rewriten as:

$$\Psi'_{Data} = \Psi'(eqID + \gamma(eqIx + eqIy + eqIz)). \tag{5.6}$$

If we use temporal derivative as I_D, then Equation (5.2) became the motion constraint equation. As velocities get better, Equations (5.2), (5.3), (5.4) and (5.5) should become close to zero. Tables 5.8, 5.9 and 5.10 show the average change in Ψ'_{data} and Ψ'_{smooth}, the average values of $eqID$, $eqIx$, $eqIy$ and $eqIz$, the average angle error and standard deviation for 10 inner iterations for the 1^{st}, the 5^{th} and the 10^{th} outer iteration of **sinL** using correct derivatives. Tables 5.11, 5.12 and 5.13 show same items for **sinR** and Tables 5.14, 5.15 and 5.16 show those results for **sin**. We can

see as we perform more iterations, $eqID$, $eqIx$, $eqIy$ and $eqIz$ are getting smaller and smaller, that means the constraints are better satisfied. Brox er al. [7] used 10 inner and 10 outer iterations only for all their results. However, they never report any investigation into the effect of this number of iterations.

The result for **sinL** converges faster than **sinR** and **sin**, in the 10^{th} outer iteration, values of these equations are almost zeros, and the average angle error goes down to 0.01°, which is very close to the correct velocities. For all three datasets, our program works as expected, the average angle error decrease steadily for both outer and inner iterations.

Tables 5.17, 5.18 and 5.19 show the average change in Ψ'_{data} and Ψ'_{smooth}, the average of $eqID$, $eqIx$, $eqIy$ and $eqIz$, the average angle error and standard for 10 inner iterations for the 1^{st}, the 5^{th} and the 10^{th} iterations of **sinL** using derivatives computed by Brox's method (simple differences). Tables 5.20, 5.21 and 5.22 show the same items for **sinR** and Tables 5.23, 5.24 and 5.25 show those iterms for **sin**. In all cases, $eqID$, $eqIx$, $eqIy$ and $eqIz$ still decrease as more iteration are performed, although it is harder for them to approach 0 when these terms reach certain values. We can say that our algorithm still seeks to find better solutions that satisfy these terms, but this time, the changes in average angle error are not non-negative: the error typically decreases for first number of outer iterations and then stays the same or even increases as showed in Tables 5.19 and 5.25. Since the values of $eqID$, $eqIx$, $eqIy$ and $eqIz$ depend on both the velocities and the derivatives, it is possible that poorer derivatives cause the problem (and simple differences are not good derivatives). If the derivatives are not correct, then even correct velocities won't satisfy these equations.

The difference in Ψ'_{data} typically increases most of the time when we use correct temporal derivatives. From Equation 5.1, we can see that Ψ'_{data} gets its maximum when $s = 0$, which means $eqID$, $eqIx$, $eqIy$ and $eqIz$ are all perfectly satisfied. At the beginning of one implementation, $eqID$, $eqIx$, $eqIy$ and $eqIz$ are all large numbers (as the initially the flow is set to 0) so Ψ'_{data} is very small. Thus large changes in

Inner	Ψ'_{data} diff.	$\Psi'_{sm.}$ diff.	ave_eqID	ave_eqIx	ave_eqIy	ave_eqIz	AEE.±STD.
1	0.0005757	15.8113	165388.024	4880.088	290.829	3744.364	$70.52° \pm 1.36°$
2	0.0000186	1.5657	154326.567	4541.909	273.573	3375.727	$66.34° \pm 2.15°$
3	0.0000192	1.5775	143712.121	4247.741	258.062	3067.970	$62.52° \pm 2.59°$
4	0.0000198	1.0530	133757.418	3985.681	243.885	2805.491	$59.06° \pm 2.82°$
5	0.0000203	0.7108	124535.555	3748.520	230.806	2578.182	$55.95° \pm 2.92°$
6	0.0000207	0.5015	116051.067	3531.575	218.680	2379.038	$53.16° \pm 2.98°$
7	0.0000211	0.3676	108273.283	3331.648	207.407	2203.005	$50.66° \pm 3.02°$
8	0.0000215	0.2785	101155.125	3146.444	196.915	2046.296	$48.42° \pm 3.07°$
9	0.0000219	0.2168	94643.381	2974.236	187.147	1905.989	$46.42° \pm 3.14°$
10	0.0000223	0.1715	88684.248	2813.667	178.054	1779.768	$44.62° \pm 3.22°$

Table 5.8: Average differences in Ψ'_{data} and Ψ'_{smooth} and the values for eqID, eqIx, eqIy, eqIz, average angle error and standard deviations for 10 inner iterations of the 1^{st} outer iteration of the 9^{th} volume of **sinL** *computed with correct derivatives ($\alpha = 100$, $\gamma = 100$, number of outer iterations=10, number of inner iterations=10, 1 pyramid level with no median filtering).*

Ψ'_{data} may return small difference. As we get better velocities, the values of those equations drop dramaticlly, this will lead to increases in Ψ'_{data}, so its quite likely the average difference for Ψ'_{data} increase. When using our computed derivatives, things are different. It seems to be much more difficult for $eqID$, $eqIx$, $eqIy$ and $eqIz$ to get smaller, if these values stay large then we will have small Ψ'_{data} values, which, in turn, yield small differences in Ψ'_{data} values.

In this thesis, we used 1 outer iteration and 100 inner iterations most of the time, because this gives much better results than using different sets of numbers of inner and outer iterations in our implementation. Again, we emphasize that Brox et al. [7] do not investigate this behaviour. One possible reason we don't get good results when using more outer iterations may be the poor quality of the temporal derivatives we use. These derivatives are used to update Ψ'_{data} and Ψ'_{smooth} in each outer iteration which, in turn, may introduce errors in further iterations.

Inner	Ψ'_{data} diff.	$\Psi'_{sm.}$ diff.	ave_eqID	ave_eqIx	ave_eqIy	ave_eqIz	AEE.±STD.
1	0.0082739	7.7968	1098.440	11.958	14.419	19.12164	8.97° ± 2.14°
2	0.0016693	0.2932	842.522	8.329	11.642	13.39991	8.30° ± 2.17°
3	0.0016386	0.2449	644.129	6.164	9.452	9.98087	7.73° ± 2.19°
4	0.0016760	0.3083	498.180	4.688	7.709	7.69513	7.22° ± 2.21°
5	0.0017140	0.3414	390.839	3.621	6.315	6.06505	6.77° ± 2.22°
6	0.0017587	0.3530	310.990	2.827	5.197	4.85543	6.37° ± 2.23°
7	0.0018112	0.3538	250.749	2.228	4.296	3.93430	6.02° ± 2.24°
8	0.0018687	0.3482	204.671	1.769	3.569	3.22025	5.69° ± 2.24°
9	0.0019387	0.3386	168.977	1.415	2.981	2.65944	5.40° ± 2.24°
10	0.0020133	0.3264	141.007	1.141	2.503	2.21451	5.14° ± 2.24°

Table 5.9: Average differences in Ψ'_{data} and Ψ'_{smooth} and values of eqID, eqIx, eqIy, eqIz, average angle error and standard deviation for 10 inner iterations of the 5^{th} outer iteration of the 9^{th} volume of **sinL** *computed with correct derivatives (α = 100, γ = 100, number of outer iterations=10, number of inner iteration=10, 1 pyramid level with no median filtering).*

Inner	Ψ'_{data} diff.	$\Psi'_{sm.}$ diff.	ave_eqID	ave_eqIx	ave_eqIy	ave_eqIz	AEE.±STD.
1	8.5584141	15.7504	0.035	0.000	0.001	0.000	0.10° ± 0.20°
2	1.2472684	0.0164	0.016	0.000	0.001	0.000	0.07° ± 0.17°
3	0.6580387	0.0147	0.008	0.000	0.000	0.000	0.06° ± 0.15°
4	0.4979960	0.0111	0.004	0.000	0.000	0.000	0.04° ± 0.14°
5	0.4392564	0.0078	0.002	0.000	0.000	0.000	0.03° ± 0.13°
6	0.4179021	0.0052	0.001	0.000	0.000	0.000	0.02° ± 0.12°
7	0.4137857	0.0034	0.001	0.000	0.000	0.000	0.02° ± 0.12°
8	0.4172870	0.0022	0.000	0.000	0.000	0.000	0.01° ± 0.12°
9	0.4219028	0.0014	0.000	0.000	0.000	0.000	0.01° ± 0.12°
10	0.4219727	0.0009	0.000	0.000	0.000	0.000	0.01° ± 0.12°

Table 5.10: Average differences in Ψ'_{data} and Ψ'_{smooth} and values of eqID, eqIx, eqIy, eqIz, average angle error and standard deviation for 10 inner iteration of the 10^{th} outer iteration of the 9^{th} volume of **sinL** *computed with correct derivatives (α = 100, γ = 100, number of outer iteration= 10, number of inner iteration= 10, 1 pyramid level with no median filtering).*

Inner	Ψ'_{data} diff.	$\Psi'_{sm.}$ diff.	ave_eqID	ave_eqIx	ave_eqIy	ave_eqIz	AEE.±STD.
1	0.0006263	15.8113	174020.271	1473.309	5202.047	1442.264	70.72° ± 1.41°
2	0.0000209	1.5508	162608.505	1345.116	4758.219	1333.896	66.73° ± 2.41°
3	0.0000204	1.6096	151591.336	1235.412	4376.090	1239.250	63.11° ± 3.09°
4	0.0000207	1.0678	141162.375	1139.544	4040.418	1154.832	59.86° ± 3.53°
5	0.0000212	0.7067	131404.948	1054.529	3741.404	1078.512	56.97° ± 3.77°
6	0.0000216	0.4857	122339.637	978.328	3472.350	1008.889	54.41° ± 3.86°
7	0.0000221	0.3457	113952.119	909.487	3228.473	944.987	52.15° ± 3.84°
8	0.0000225	0.2538	106209.658	846.929	3006.221	886.084	50.16° ± 3.74°
9	0.0000230	0.1934	99070.802	789.826	2802.851	831.620	48.41° ± 3.58°
10	0.0000236	0.1521	92490.992	737.519	2616.178	781.143	46.86° ± 3.38°

Table 5.11: Average differences in Ψ'_{data} and Ψ'_{smooth} and values of eqID, eqIx, eqIy, eqIz, average angle error and standard deviation for 10 inner iterations of the 1^{st} outer iteration of the 9^{th} volume of **sinR** *computed with correct derivatives ($\alpha = 100$, $\gamma = 100$, number of outer iteration= 10, number of inner iteration= 10, 1 pyramid level with no median filtering).*

Inner	Ψ'_{data} diff.	$\Psi'_{sm.}$ diff.	ave_eqID	ave_eqIx	ave_eqIy	ave_eqIz	AEE.±STD.
1	0.0087410	8.0905	1541.063	7.256	16.437	14.975	23.64° ± 2.91°
2	0.0006652	0.3158	1498.022	6.549	14.175	13.976	23.11° ± 2.92°
3	0.0003681	0.1222	1435.803	6.094	12.770	13.232	22.60° ± 2.93°
4	0.0003003	0.0839	1371.598	5.748	11.742	12.613	22.10° ± 2.94°
5	0.0002687	0.0765	1309.652	5.458	10.923	12.068	21.61° ± 2.94°
6	0.0002525	0.0744	1250.991	5.204	10.240	11.574	21.13° ± 2.94°
7	0.0002441	0.0743	1195.713	4.973	9.650	11.116	20.66° ± 2.93°
8	0.0002400	0.0751	1143.638	4.759	9.131	10.686	20.21° ± 2.91°
9	0.0002386	0.0761	1094.515	4.560	8.666	10.278	19.77° ± 2.90°
10	0.0002390	0.0774	1048.092	4.372	8.244	9.889	19.33° ± 2.88°

Table 5.12: Average differences of Ψ'_{data} and Ψ'_{smooth} and values of eqID, eqIx, eqIy, eqIz, average angle error and standard deviation for 10 inner iterations of the 5^{th} outer iteration of the 9^{th} volume of **sinR** *computed with correct derivatives ($\alpha = 100$, $\gamma = 100$, number of outer iteration= 10, number of inner iteration= 10, 1 pyramid level with no median filtering).*

Inner	Ψ'_{data} diff.	$\Psi'_{sm.}$ diff.	ave_eqID	ave_eqIx	ave_eqIy	ave_eqIz	AEE.±STD.
1	0.0437556	12.0810	90.592	0.375	0.582	0.855	5.16° ± 1.64°
2	0.0040966	0.0805	83.322	0.339	0.516	0.769	4.92° ± 1.61°
3	0.0033379	0.0970	75.844	0.309	0.468	0.699	4.70° ± 1.58°
4	0.0032807	0.1165	68.964	0.282	0.427	0.638	4.48° ± 1.54°
5	0.0033012	0.1262	62.755	0.258	0.390	0.584	4.28° ± 1.50°
6	0.0033703	0.1303	57.170	0.236	0.357	0.535	4.09° ± 1.46°
7	0.0034637	0.1317	52.140	0.216	0.327	0.491	3.91° ± 1.42°
8	0.0035742	0.1313	47.601	0.198	0.300	0.451	3.73° ± 1.38°
9	0.0036981	0.1298	43.497	0.181	0.275	0.414	3.57° ± 1.34°
10	0.0038334	0.1274	39.771	0.166	0.252	0.381	3.41° ± 1.30°

Table 5.13: Average differences of Ψ'_{data} and Ψ'_{smooth} and values of eqID, eqIx, eqIy, eqIz, average angle error and standard deviation for 10 inner iterations of the 10^{th} outer iteration of the 9^{th} volume of **sinR** *computed with correct derivatives (α = 100, γ = 100, number of outer iteration= 10, number of inner iteration= 10, 1 pyramid level with no median filtering).*

Inner	Ψ'_{data} diff.	$\Psi'_{sm.}$ diff.	ave_eqID	ave_eqIx	ave_eqIy	ave_eqIz	AEE.±STD.
1	0.0005877	15.8113	170839.255	3722.804	2099.715	2897.699	70.69° ± 1.73°
2	0.0000193	2.0166	159501.420	3455.617	1924.327	2623.482	66.85° ± 3.05°
3	0.0000188	1.6025	148917.033	3230.226	1776.601	2400.909	63.41° ± 4.17°
4	0.0000190	1.0764	139030.474	3031.516	1647.400	2211.502	60.34° ± 5.16°
5	0.0000193	0.7289	129889.763	2853.311	1532.762	2047.496	57.63° ± 6.04°
6	0.0000196	0.5160	121482.966	2691.516	1429.927	1903.577	55.23° ± 6.83°
7	0.0000199	0.3808	113774.957	2543.410	1336.984	1776.070	53.12° ± 7.56°
8	0.0000202	0.2914	106717.469	2407.037	1252.514	1662.261	51.25° ± 8.24°
9	0.0000205	0.2304	100257.859	2280.929	1175.425	1560.079	49.61° ± 8.86°
10	0.0000208	0.1866	94343.514	2163.938	1104.849	1467.893	48.15° ± 9.44°

Table 5.14: Average differences in Ψ'_{data} and Ψ'_{smooth} and values of eqID, eqIx, eqIy, eqIz, average angle error and standard deviation for 10 inner iterations of the 1^{st} outer iteration of the 9^{th} volume of **sin** *computed with correct derivatives (α = 100, γ = 100, number of outer iteration= 10, number of inner iteration= 10, 1 pyramid level with no median filtering).*

Inner	Ψ'_{data} diff.	$\Psi'_{sm.}$ diff.	ave_eqID	ave_eqIx	ave_eqIy	ave_eqIz	AEE.±STD.
1	0.0131558	5.8824	2724.246	20.078	28.104	34.088	19.30° ± 20.05°
2	0.0347743	0.2602	2234.735	14.483	22.206	25.176	18.60° ± 20.12°
3	0.0059093	0.2068	2041.729	12.709	20.032	22.482	18.00° ± 20.19°
4	0.0051870	0.2228	1905.180	11.597	18.541	20.735	17.49° ± 20.25°
5	0.0045679	0.2327	1804.209	10.821	17.427	19.489	17.03° ± 20.30°
6	0.0043418	0.2334	1726.878	10.246	16.556	18.546	16.62° ± 20.34°
7	0.0042219	0.2287	1666.133	9.807	15.859	17.809	16.25° ± 20.37°
8	0.0041733	0.2211	1617.340	9.465	15.294	17.218	15.91° ± 20.40°
9	0.0041689	0.2119	1577.376	9.195	14.830	16.736	15.61° ± 20.43°
10	0.0041954	0.2017	1544.075	8.977	14.445	16.335	15.32° ± 20.45°

Table 5.15: Average differences in Ψ'_{data} and Ψ'_{smooth} and values of eqID, eqIx, eqIy, eqIz, average angle error and standard deviation for 10 inner iteration of the 5^{th} outer iteration of the 9^{th} volume of **sin** *computed with correct derivatives ($\alpha = 100$, $\gamma = 100$, number of outer iteration= 10, number of inner iteration= 10, 1 pyramid level with no median filtering).*

Inner	Ψ'_{data} diff.	$\Psi'_{sm.}$ diff.	ave_eqID	ave_eqIx	ave_eqIy	ave_eqIz	AEE.±STD.
1	5.2872770	11.2821	684.577	5.147	7.791	9.594	6.90° ± 18.75°
2	0.6563276	0.0441	667.738	5.040	7.599	9.421	6.80° ± 18.71°
3	0.3178679	0.0422	658.751	5.011	7.546	9.373	6.71° ± 18.67°
4	0.2265391	0.0396	651.884	4.991	7.507	9.332	6.64° ± 18.64°
5	0.1905575	0.0362	646.484	4.975	7.475	9.300	6.58° ± 18.62°
6	0.1744226	0.0330	642.100	4.963	7.449	9.274	6.52° ± 18.60°
7	0.1672531	0.0301	638.465	4.952	7.427	9.252	6.47° ± 18.59°
8	0.1642882	0.0275	635.401	4.943	7.408	9.233	6.43° ± 18.57°
9	0.1626395	0.0252	632.786	4.935	7.392	9.216	6.39° ± 18.56°
0	0.1600340	0.0232	630.531	4.928	7.377	9.201	6.36° ± 18.55°

Table 5.16: Average differences in Ψ'_{data} and Ψ'_{smooth} and values of eqID, eqIx, eqIy, eqIz, average angle error and standard deviation for 10 inner iteration of the 10^{th} outer iteration of the 9^{th} volume of **sin** *computed with correct derivatives ($\alpha = 100$, $\gamma = 100$, number of outer iteration= 10, number of inner iteration= 10, 1 pyramid level with no median filtering).*

Inner	Ψ'_{data} diff.	$\Psi'_{sm.}$ diff.	ave_eqID	ave_eqIx	ave_eqIy	ave_eqIz	AEE.±STD.
1	0.0006244	15.8113	147292.081	4269.110	256.143	3048.934	71.16° ± 1.56°
2	0.0000178	2.2258	139472.223	4018.280	243.491	2765.318	67.54° ± 2.45°
3	0.0000169	1.6046	131891.727	3806.540	232.376	2547.300	64.20° ± 2.98°
4	0.0000168	1.0286	124726.129	3620.117	222.311	2366.048	61.13° ± 3.30°
5	0.0000166	0.6756	118060.324	3452.592	213.074	2211.716	58.31° ± 3.48°
6	0.0000165	0.4738	111910.149	3299.907	204.525	2077.912	55.74° ± 3.59°
7	0.0000164	0.3513	106259.089	3159.446	196.575	1960.467	53.39° ± 3.65°
8	0.0000163	0.2727	101075.082	3029.396	189.159	1856.413	51.24° ± 3.70°
9	0.0000161	0.2190	96320.190	2908.439	182.230	1763.553	49.28° ± 3.74°
10	0.0000160	0.1800	91955.575	2795.567	175.749	1680.195	47.49° ± 3.77°

Table 5.17: Average differences in Ψ'_{data} and Ψ'_{smooth} and values of eqID, eqIx, eqIy, eqIz, average angle error and standard deviation for 10 inner iterations of the 1^{st} outer iteration of the 9^{th} volume of **sinL** *computed with Brox derivatives (α = 100, γ = 100, number of outer iteration= 10, number of inner iteration= 10, 1 pyramid level with no median filtering).*

Inner	Ψ'_{data} diff.	$\Psi'_{sm.}$ diff.	ave_eqID	ave_eqIx	ave_eqIy	ave_eqIz	AEE.±STD.
1	0.0021455	2.833	19413.530	486.131	59.527	178.935	17.46° ± 7.70°
2	0.0001285	0.2273	19433.546	459.870	58.512	158.668	17.46° ± 7.84°
3	0.0000563	0.1099	19152.747	448.188	57.868	149.400	17.46° ± 7.94°
4	0.0000396	0.0642	18823.464	441.682	57.403	143.217	17.46° ± 8.02°
5	0.0000312	0.0455	18514.738	437.589	57.040	138.626	17.48° ± 8.08°
6	0.0000261	0.0359	18242.864	434.750	56.746	134.992	17.50° ± 8.13°
7	0.0000225	0.0300	18008.108	432.641	56.502	132.013	17.53° ± 8.17°
8	0.0000196	0.0258	17806.208	430.997	56.297	129.519	17.57° ± 8.21°
9	0.0000173	0.0226	17632.203	429.671	56.124	127.403	17.60° ± 8.25°
10	0.0000154	0.0200	17481.584	428.578	55.978	125.591	17.64° ± 8.28°

Table 5.18: Average differences in Ψ'_{data} and Ψ'_{smooth} and values of eqID, eqIx, eqIy, eqIz, average angle error and standard deviation for 10 inner iterations of the 5^{th} outer iteration of the 9^{th} volume of **sinL** *computed with Brox derivatives (α = 100, γ = 100, number of outer iteration= 10, number of inner iteration= 10, 1 pyramid level with no median filtering).*

Inner	Ψ'_{data} diff.	$\Psi'_{sm.}$ diff.	ave_eqID	ave_eqIx	ave_eqIy	ave_eqIz	AEE.±STD.
1	0.0031046	1.7147	15032.074	371.603	55.587	78.968	21.49° ± 9.10°
2	0.0000485	0.0168	15050.697	369.376	55.517	77.853	21.55° ± 9.11°
3	0.0000106	0.0080	15022.048	368.590	55.506	77.456	21.60° ± 9.12°
4	0.0000070	0.0050	14988.586	368.241	55.504	77.207	21.64° ± 9.12°
5	0.0000052	0.0036	14958.512	368.056	55.499	77.021	21.67° ± 9.12°
6	0.0000042	0.0028	14932.918	367.941	55.490	76.872	21.70° ± 9.12°
7	0.0000035	0.0023	14911.338	367.858	55.476	76.746	21.72° ± 9.12°
8	0.0000030	0.0019	14893.067	367.794	55.458	76.638	21.74° ± 9.12°
9	0.0000026	0.0017	14877.477	367.741	55.438	76.545	21.76° ± 9.12°
10	0.0000023	0.0015	14864.063	367.696	55.416	76.463	21.77° ± 9.12°

Table 5.19: Average differences in Ψ'_{data} and Ψ'_{smooth} amd values of eqID, eqIx, eqIy, eqIz, average angle error and standard deviation for 10 inner iterations of the 10^{th} outer iteration of the 9^{th} volume of **sinL** *computed with Brox derivatives ($\alpha = 100$, $\gamma = 100$, number of outer iteration= 10, number of inner iteration= 10, 1 pyramid level with no median filtering).*

Inner	Ψ'_{data} diff.	$\Psi'_{sm.}$ diff.	ave_eqID	ave_eqIx	ave_eqIy	ave_eqIz	AEE.±STD.
1	0.0006843	15.8113	158925.576	1330.642	4700.317	1176.034	71.13° ± 1.51°
2	0.0000242	2.0313	149866.908	1225.133	4332.489	1088.983	67.51° ± 2.54°
3	0.0000207	1.5790	141030.326	1135.687	4019.917	1019.625	64.19° ± 3.27°
4	0.0000201	1.0373	132610.519	1057.754	3746.652	958.945	61.19° ± 3.76°
5	0.0000199	0.6863	124698.745	988.744	3503.933	904.641	58.48° ± 4.06°
6	0.0000199	0.4772	117325.017	926.910	3285.860	855.306	56.05° ± 4.22°
7	0.0000200	0.3465	110485.872	871.034	3088.320	810.088	53.88° ± 4.27°
8	0.0000201	0.2616	104159.753	820.223	2908.300	768.399	51.94° ± 4.24°
9	0.0000202	0.2044	98316.181	773.797	2743.510	729.808	50.20° ± 4.15°
10	0.0000203	0.1643	92921.036	731.221	2592.141	693.979	48.65° ± 4.01°

Table 5.20: Average differences in Ψ'_{data} and Ψ'_{smooth} and values of eqID, eqIx, eqIy, eqIz, average angle error and standard deviation for 10 inner iteration of the 1^{st} outer iteration of the 9^{th} volume of **sinR** *computed with Brox derivatives ($\alpha = 100$, $\gamma = 100$, number of outer iteration= 10, number of inner iteration= 10, 1 pyramid level with no median filtering).*

Inner	Ψ'_{data} diff.	$\Psi'_{sm.}$ diff.	ave_eqID	ave_eqIx	ave_eqIy	ave_eqIz	AEE.±STD.
1	0.0029900	3.8625	15752.018	83.246	266.967	111.326	28.65° ± 7.87°
2	0.0002654	0.2532	15866.260	79.768	253.039	107.947	28.49° ± 8.08°
3	0.0000890	0.1369	15769.865	78.111	246.599	106.852	28.32° ± 8.27°
4	0.0000571	0.0854	15625.330	77.159	242.754	106.201	28.15° ± 8.43°
5	0.0000421	0.0597	15480.882	76.543	240.165	105.785	27.98° ± 8.58°
6	0.0000335	0.0452	15350.549	76.100	238.256	105.483	27.81° ± 8.72°
7	0.0000281	0.0360	15236.861	75.758	236.761	105.254	27.65° ± 8.84°
8	0.0000244	0.0297	15138.610	75.479	235.541	105.073	27.49° ± 8.96°
9	0.0000217	0.0250	15053.641	75.242	234.516	104.923	27.33° ± 9.06°
10	0.0000192	0.0215	14979.794	75.036	233.639	104.795	27.18° ± 9.16°

Table 5.21: Average differences of Ψ'_{data} and Ψ'_{smooth} and values of eqID, eqIx, eqIy, eqIz, average angle error and standard deviation for 10 inner iterations of the 5^{th} outer iteration of the 9^{th} volume of **sinR** *computed with Brox derivatives (α = 100, γ = 100, number of outer iteration= 10, number of inner iteration= 10, 1 pyramid level with no median filtering).*

Inner	Ψ'_{data} diff.	$\Psi'_{sm.}$ diff.	ave_eqID	ave_eqIx	ave_eqIy	ave_eqIz	AEE.±STD.
1	0.0050282	2.3226	13369.295	64.826	189.553	95.292	23.41° ± 12.04°
2	0.0002897	0.0263	13381.035	64.450	187.967	94.724	23.37° ± 12.07°
3	0.0000431	0.0140	13354.548	64.299	187.276	94.615	23.33° ± 12.09°
4	0.0000290	0.0093	13323.898	64.217	186.879	94.556	23.29° ± 12.11°
5	0.0000232	0.0070	13295.874	64.161	186.606	94.515	23.24° ± 12.12°
6	0.0000199	0.0056	13271.489	64.116	186.396	94.481	23.19° ± 12.14°
7	0.0000176	0.0047	13250.435	64.077	186.223	94.451	23.15° ± 12.15°
8	0.0000162	0.0041	13232.170	64.042	186.074	94.423	23.10° ± 12.16°
9	0.0000152	0.0037	13216.192	64.010	185.943	94.397	23.06° ± 12.17°
10	0.0000141	0.0033	13202.094	63.979	185.826	94.373	23.02° ± 12.17°

Table 5.22: Average differences in Ψ'_{data} and Ψ'_{smooth} and values of eqID, eqIx, eqIy, eqIz, average angle error and standard deviation for 10 inner iterations of the 10^{th} outer iteration of the 9^{th} volume of **sinR** *computed with Brox derivatives (α = 100, γ = 100, number of outer iteration= 10, number of inner iteration= 10, 1 pyramid level with no median filtering).*

Inner	Ψ'_{data} diff.	$\Psi'_{sm.}$ diff.	ave_eqID	ave_eqIx	ave_eqIy	ave_eqIz	AEE.±STD.
1	0.0006147	15.8113	165581.084	4995.388	2959.526	3359.341	71.58° ± 2.78°
2	0.0000220	3.0029	156974.453	4492.147	2506.178	2874.610	68.38° ± 4.39°
3	0.0000176	1.5886	148067.529	4290.321	2324.687	2675.563	65.44° ± 5.76°
4	0.0000169	1.0100	140136.877	4117.965	2179.022	2517.092	62.77° ± 6.92°
5	0.0000164	0.6699	132919.615	3973.430	2060.203	2388.246	60.35° ± 7.92°
6	0.0000161	0.4728	126351.566	3846.196	1958.239	2279.019	58.16° ± 8.80°
7	0.0000159	0.3527	120364.526	3732.287	1868.857	2184.658	56.18° ± 9.59°
8	0.0000157	0.2752	114901.082	3628.943	1789.298	2101.960	54.40° ± 10.30°
9	0.0000156	0.2224	109908.605	3534.378	1717.767	2028.736	52.78° ± 10.94°
10	0.0000155	0.1844	105339.423	3447.311	1652.983	1963.382	51.33° ± 11.52°

Table 5.23: Average differences in Ψ'_{data} and Ψ'_{smooth} and values of eqID, eqIx, eqIy, eqIz, average angle error and standard deviation for 10 inner iterations of the 1^{st} outer iteration of the 9^{th} volume of **sin** *computed with Brox derivatives ($\alpha = 100$, $\gamma = 100$, number of outer iteration= 10, number of inner iteration= 10, 1 pyramid level with no median filtering).*

Inner	Ψ'_{data} diff.	$\Psi'_{sm.}$ diff.	ave_eqID	ave_eqIx	ave_eqIy	ave_eqIz	AEE.±STD.
1	0.0024073	2.5376	31519.601	909.836	280.887	337.260	28.99° ± 22.11°
2	0.0002555	0.1798	31445.193	825.819	259.154	299.651	28.97° ± 22.17°
3	0.0000629	0.0899	31064.129	814.224	255.886	292.046	28.97° ± 22.22°
4	0.0000417	0.0538	30742.754	808.199	254.104	287.425	28.98° ± 22.26°
5	0.0000311	0.0383	30475.279	804.830	253.022	284.270	28.99° ± 22.30°
6	0.0000251	0.0300	30254.750	802.607	252.271	281.892	29.01° ± 22.33°
7	0.0000209	0.0248	30072.799	801.016	251.714	280.012	29.03° ± 22.35°
8	0.0000178	0.0210	29921.600	799.807	251.284	278.479	29.06° ± 22.37°
9	0.0000155	0.0181	29794.818	798.854	250.941	277.206	29.08° ± 22.38°
10	0.0000136	0.0158	29687.521	798.079	250.662	276.134	29.11° ± 22.39°

Table 5.24: Average differences in Ψ'_{data} and Ψ'_{smooth} and values of eqID, eqIx, eqIy, eqIz, average angle error and standard deviation for 10 inner iterations of the 5^{th} outer iteration of the 9^{th} volume of **sin** *computed with Brox derivatives ($\alpha = 100$, $\gamma = 100$, number of outer iteration= 10, number of inner iteration= 10, 1 pyramid level with no median filtering).*

Inner	Ψ'_{data} diff.	$\Psi'_{sm.}$ diff.	ave_eqID	ave_eqIx	ave_eqIy	ave_eqIz	AEE.±STD.
1	0.0037653	1.6409	27424.766	648.578	220.982	224.724	30.84° ± 22.01°
2	0.0000856	0.0126	27425.272	637.404	219.588	223.019	30.86° ± 22.01°
3	0.0000106	0.0062	27388.975	636.753	219.372	222.730	30.88° ± 22.00°
4	0.0000070	0.0040	27357.891	636.436	219.257	222.555	30.89° ± 22.00°
5	0.0000053	0.0029	27332.043	636.290	219.184	222.434	30.90° ± 22.00°
6	0.0000043	0.0023	27310.677	636.198	219.126	222.339	30.91° ± 22.00°
7	0.0000037	0.0019	27292.862	636.132	219.077	222.262	30.91° ± 22.00°
8	0.0000032	0.0017	27277.817	636.081	219.032	222.197	30.91° ± 22.00°
9	0.0000028	0.0014	27264.951	636.039	218.991	222.140	30.92° ± 22.00°
10	0.0000026	0.0013	27253.823	636.002	218.952	222.091	30.92° ± 22.00°

Table 5.25: Average differences in Ψ'_{data} and Ψ'_{smooth} and values of eqID, eqIx, eqIy, eqIz, average angle error and standard deviation for 10 inner iterations of the 10^{th} outer iteration of the 9^{th} volume of **sin** *computed with Brox derivatives ($\alpha = 100$, $\gamma = 100$, number of outer iteration= 10, number of inner iteration= 10, 1 pyramid level with no median filtering).*

5.2.4 Hierarchical Flow Field

One way to understand the Brox et al. method is to see how fields change at different pyramid levels. Figure 5.1 show the evolution of flow fields for the 9^{th} volume of **sinL** from level 10 to level 1. The top level number of pyramid is 10 and the re-scale factor $\eta = 0.95$. At top of the pyramid, the size of the volume is the smallest and then both the volume size and the magnitude of the flow get larger and larger as it goes down the pyramid. Since both the correct velocities are known, we can compute the correct velocities at any level with a certain η value. This allows quantitative evaluation at each level of the pyramid so we can see how the accuracy of result evolve from level to level. Table 5.26 shows the angle error, standard deviation, the direction error and the magnitude error for the **sinL** flow fields in Figure 5.1. Table 5.27 shows results for the 9^{th} volume of **sinR** with the same parameters, with Figure 5.2 shows its corresponding flow fields. We can see for these two continuous data, the angle error is decreasing as a whole, but sometimes increase a little. The reason for this can be that with both the interpolation from level to level and warping with inaccuracies can be introduced. These errors may accumulate as we go down the pyramid and when the refinement is smaller than the errors introduced, the average angular error will

increase. This is of particular distinct when it is difficult to get accurate velocities in part of the image. Table 5.28 shows the angle error for the 9^{th} volume of **sin** data at different levels while Figure 5.3 shows its corresponding flow fields. We can see that the angular error goes down to 13.56° at level 7, but then increase to 16.20° as the final result. This is most likely because we can't compute accurate velocities in the border region where **sinL** and **sinR** meet. Warping at this part of the image will add error and these errors will accumulate as the pyramidal processing continues. When the refinement can't surpass the errors, the angular error increases.

We also exam the effect of warping. As we discussed in Section 3.2.8, the spatial difference between left and right images will be smaller and smaller as we descend the pyramid, if the velocities we compute become more and more accurate. Figure 5.4(a)-(c) shows the difference between the 15^{th} slice of the left image and the right images for the 9^{th} volume of **sinL** data at (a) the 10^{th}, (b) the 9^{th} and (c) the 1^{st} pyramid level. White means no difference while black means large difference. We can observe obvious refinement between the top two levels, but for later levels, refinements between adjacent levels are very slight and so we can hardly tell if there is any changes in the image. Therefore, we just show the final result at the last level. Figure 5.4(d)-(f) shows the difference between the 15^{th} slice of the left image and the right image for the 9^{th} volume of **sinR** data at (d) the 10^{th}, (e) the 9^{th} and (f) the 1^{st} pyramid level. This case is almost the same as for **sinL** data. Figure 5.4 (g)-(i) shows the difference between the 15^{th} slice of the left image and the right image for the 9^{th} volume of the **sin** data at (g) the 10^{th}, (h) the 9^{th}, and (i) the 1^{st} pyramid level. We can see in the border where sharp discontinuity occurs, there is always a large difference because we can't actually get the correct velocities for this part and therefore warping in this region will introduce errors.

Pyramid level	AAE. ± STD.	Magnitude_Error	Direction_Error	Density(%)
10	11.45° ± 2.57°	0.26	11.49°	100.00
9	6.86° ± 2.84°	0.14	7.31°	100.00
8	3.37° ± 3.11°	0.07	3.55°	100.00
7	2.58° ± 3.33°	0.05	2.70°	100.00
6	2.66° ± 3.57°	0.05	2.77°	100.00
5	2.43° ± 2.85°	0.05	2.52°	100.00
4	2.34° ± 2.74°	0.05	2.42°	100.00
3	2.30° ± 2.67°	0.05	2.37°	100.00
2	2.28° ± 2.74°	0.04	2.35°	100.00
1	2.06° ± 2.56°	0.04	2.12°	100.00

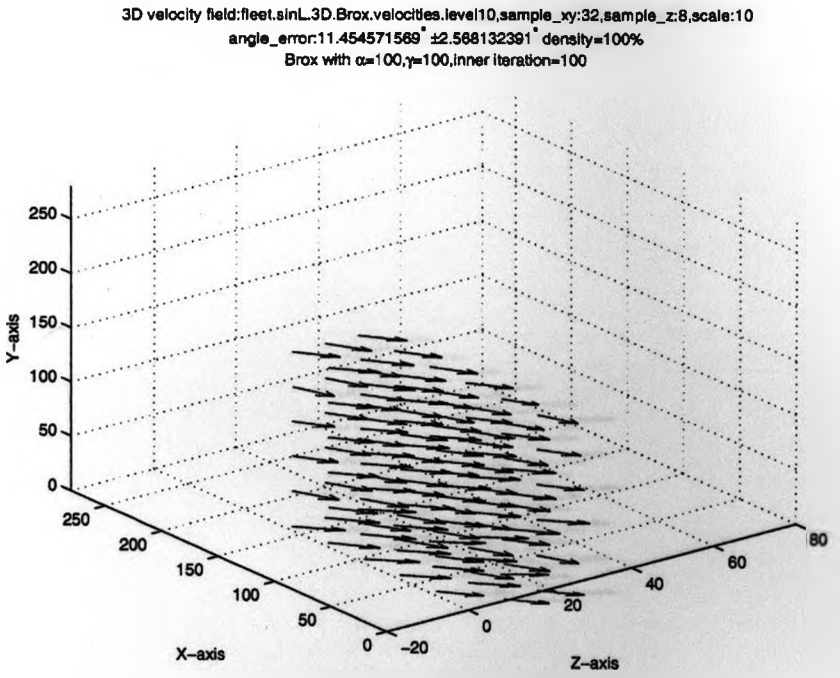
Table 5.26: The 3D angular errors, standard deviation, magnitude error, direction error and density for the flow field at all pyramid levels of the 9^{th} volume of the **sinL** *data (10 levels of pyramid, $\alpha = 100.0$, $\gamma = 100.0$, number of outer iteration= 1, number of inner iteration= 100, size of median filter= $3 \times 3 \times 3$).*

Pyramid level	AAE. ± STD.	Magnitude_Error	Direction_Error	Density(%)
10	23.06° ± 5.93°	0.45	24.65°	100.00
9	9.55° ± 4.81°	0.19	10.19°	100.00
8	3.87° ± 4.02°	0.08	4.04°	100.00
7	2.64° ± 4.02°	0.06	2.72°	100.00
6	2.61° ± 4.40°	0.06	2.69°	100.00
5	2.62° ± 4.14°	0.06	2.69°	100.00
4	2.67° ± 4.07°	0.06	2.74°	100.00
3	2.47° ± 4.01°	0.05	2.53°	100.00
2	2.15° ± 4.02°	0.05	2.19°	100.00
1	1.99° ± 3.55°	0.04	2.02°	100.00

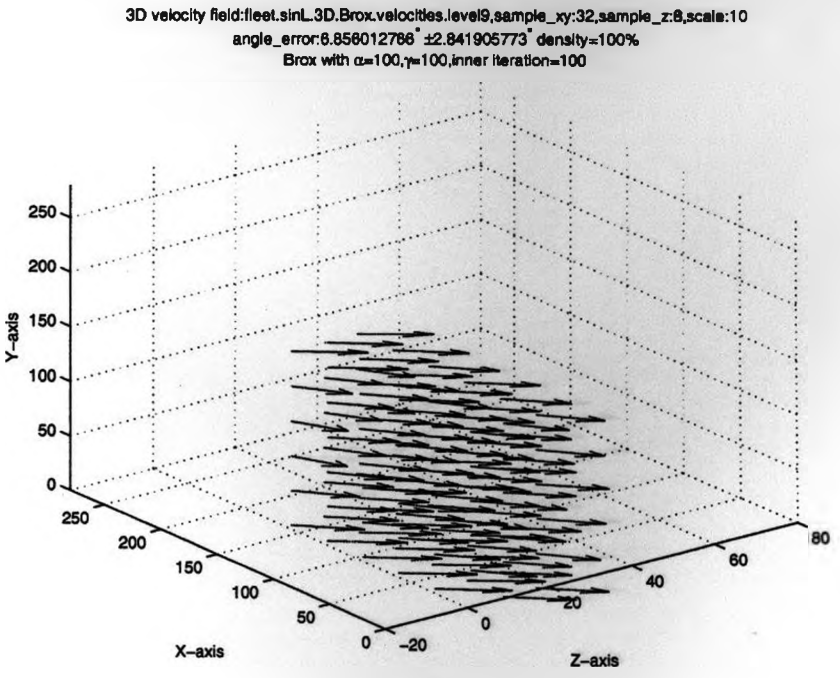
Table 5.27: The 3D angular errors, standard deviation, magnitude error, direction error and density for the flow field at all pyramid levels of the 9^{th} volume of the **sinR** *data (10 levels of pyramid, $\alpha = 100.0$, $\gamma = 100.0$, number of outer iteration= 1, number of inner iteration= 100, size of median filter= $3 \times 3 \times 3$).*

Pyramid level	AAE. ± STD.	Magnitude_Error	Direction_Error	Density(%)
10	25.12° ± 22.27°	0.47	27.57°	100.00
9	20.12° ± 24.19°	0.41	22.45°	100.00
8	15.45° ± 24.48°	0.29	17.16°	100.00
7	13.56° ± 25.26°	0.31	14.98°	100.00
6	13.85° ± 25.60°	0.30	14.83°	100.00
5	14.95° ± 26.86°	0.40	16.92°	100.00
4	16.05° ± 28.30°	1.69	17.91°	100.00
3	16.89° ± 29.79°	0.37	18.50°	100.00
2	16.55° ± 29.92°	0.30	17.77°	100.00
1	16.20° ± 29.96°	0.28	17.19°	100.00

Table 5.28: The 3D angular errors, standard deviation, magnitude error, direction error and density for the flow field at all pyramid levels of the 9^{th} *volume of the* **sin** *data (10 levels of pyramid,* $\alpha = 100.0$, $\gamma = 100.0$, *number of outer iteration=* 1, *number of inner iteration=* 100, *size of median filter=* $3 \times 3 \times 3$*).*

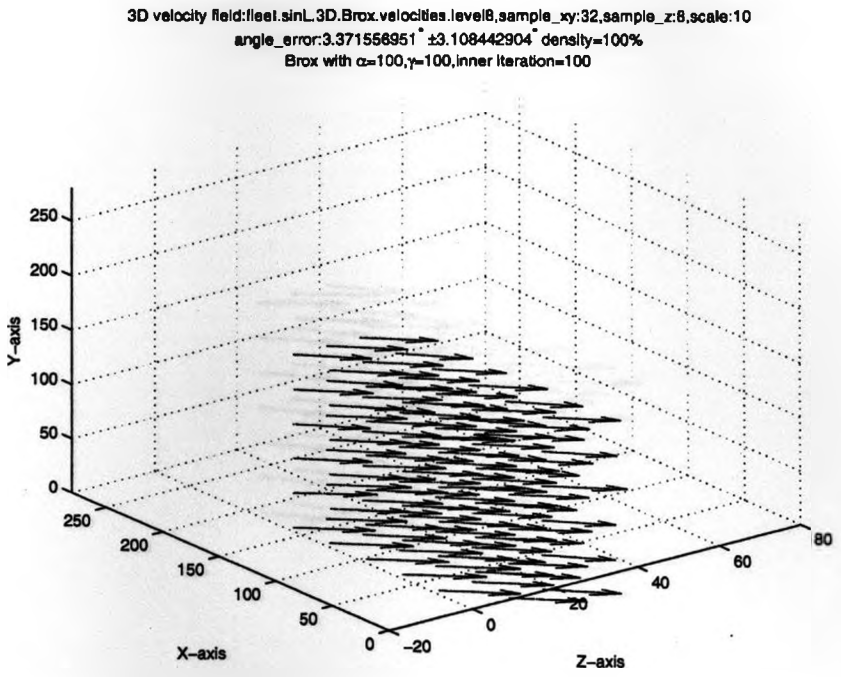


(a)

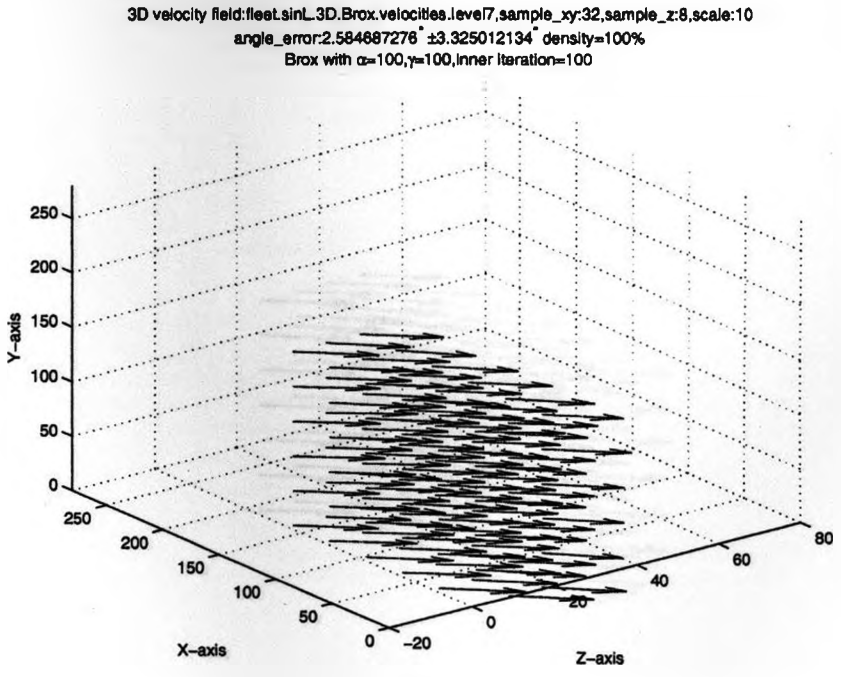


(b)

Figure 5.1: The computed flow field at (a) the 10^{th} *and (b) the* 9^{th} *level of the* 9^{th} *volume of* **sinL** *(*$\alpha = 100.0$*,* $\gamma = 100.0$*, pyramid level=* 10*, outer iteration=* 1*, inner iteration=* 100*, with* $3*3*3$ *median filter), downsampling rate* 32 *in x and y directions and* 8 *in z direction, scale factor=* 10*.*

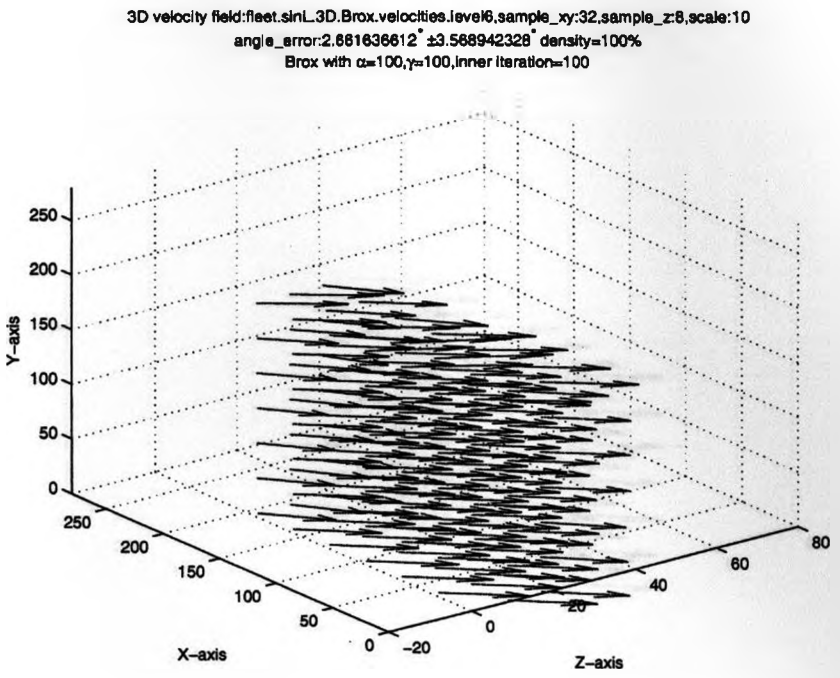


(c)

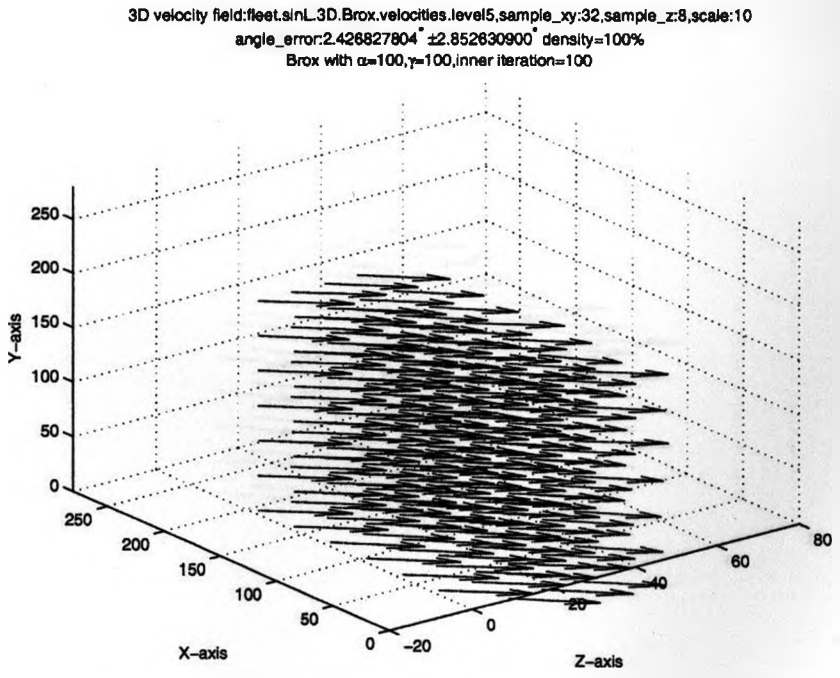


(d)

Figure 5.1: The computed flow field at (c)the 8^{th} and (d) the 7^{th} level of the 9^{th} volume of **sinL** *($\alpha = 100.0$, $\gamma = 100.0$, pyramid level= 10, outer iteration= 1, inner iteration= 100, with $3*3*3$ median filter), downsampling rate 32 in x and y directions and 8 in z direction, scale factor= 10.*

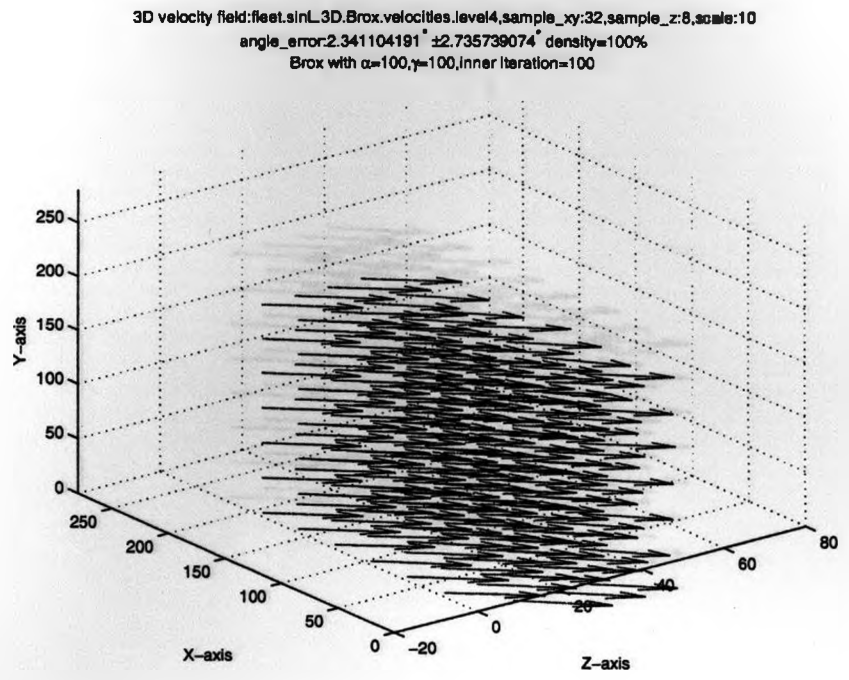


(e)

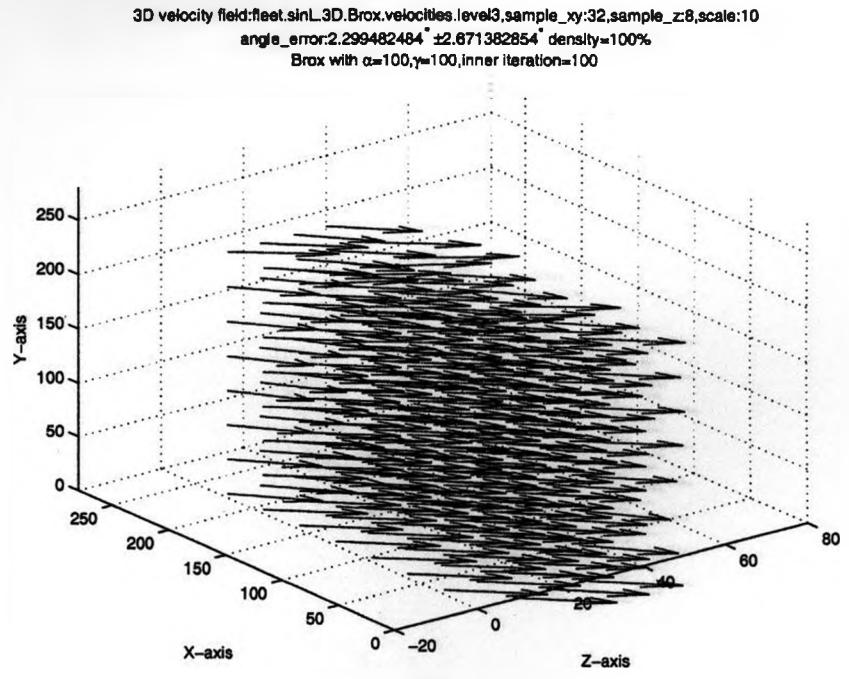


(f)

Figure 5.1: The computed flow field at (e)the 5^{th} and (f) the 5^{th} level of the 9^{th} volume of **sinL** *($\alpha = 100.0$, $\gamma = 100$, pyramid level= 10, outer iteration= 1, inner iteration= 100.0, with $3 * 3 * 3$ median filter), downsampling rate 32 in x and y directions and 8 in z direction, scale factor= 10.*

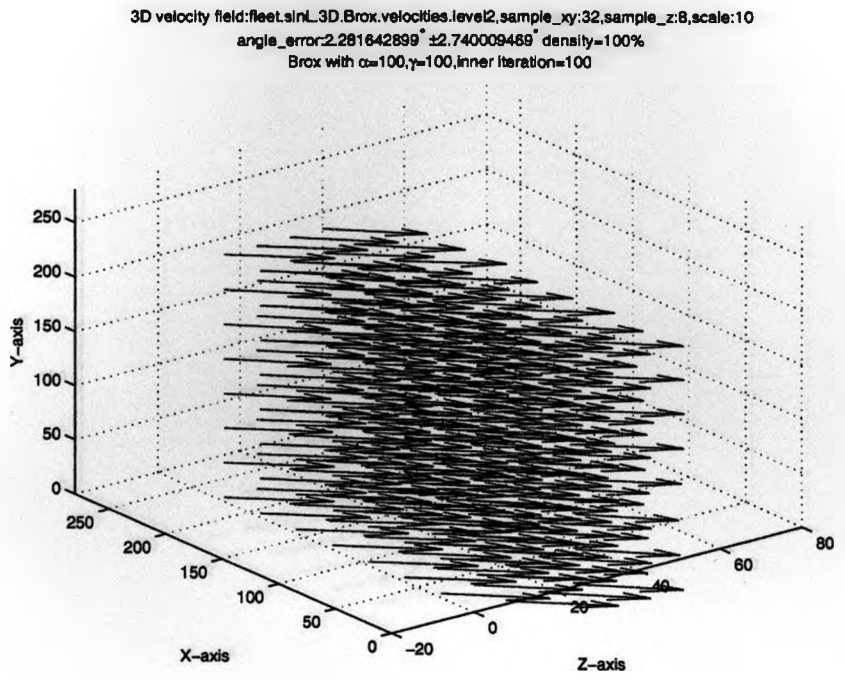


(g)

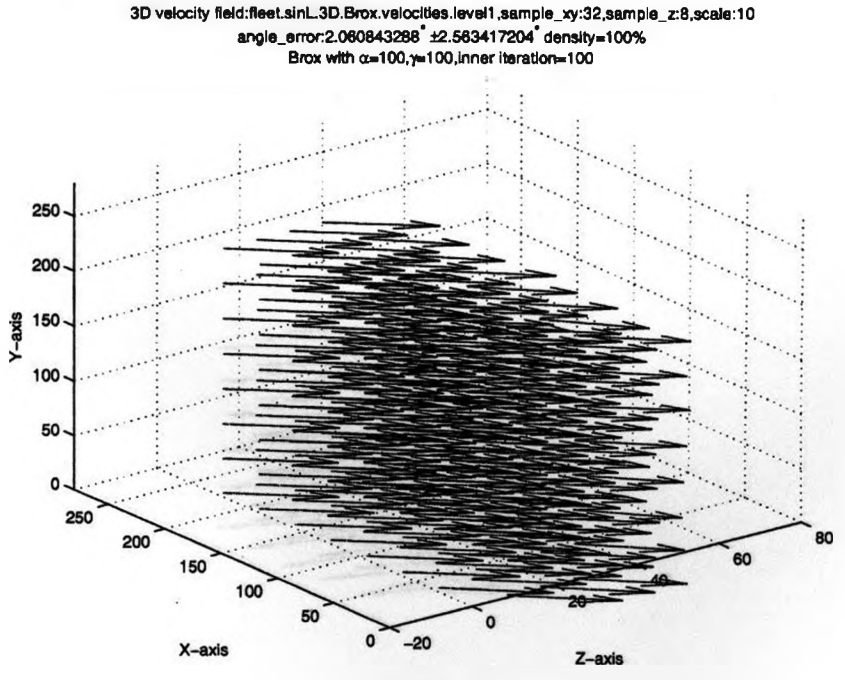


(h)

Figure 5.1: The computed flow field at (g)the 4^{th} *and (h) the* 3^{th} *level of the* 9^{th} *volume of* **sinL** *(*$\alpha = 100.0$*,* $\gamma = 100.0$*, pyramid level=* 10*, outer iteration=* 1*, inner iteration=* 100*, with* $3*3*3$ *median filter), downsampling rate* 32 *in* x *and* y *directions and* 8 *in* z *direction, scale factor=* 10*.*

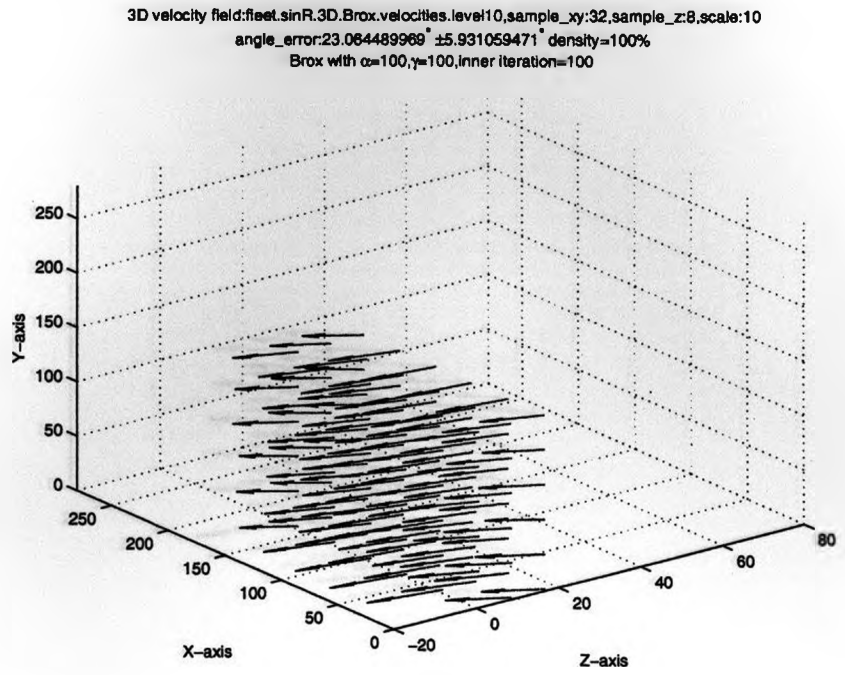


(i)

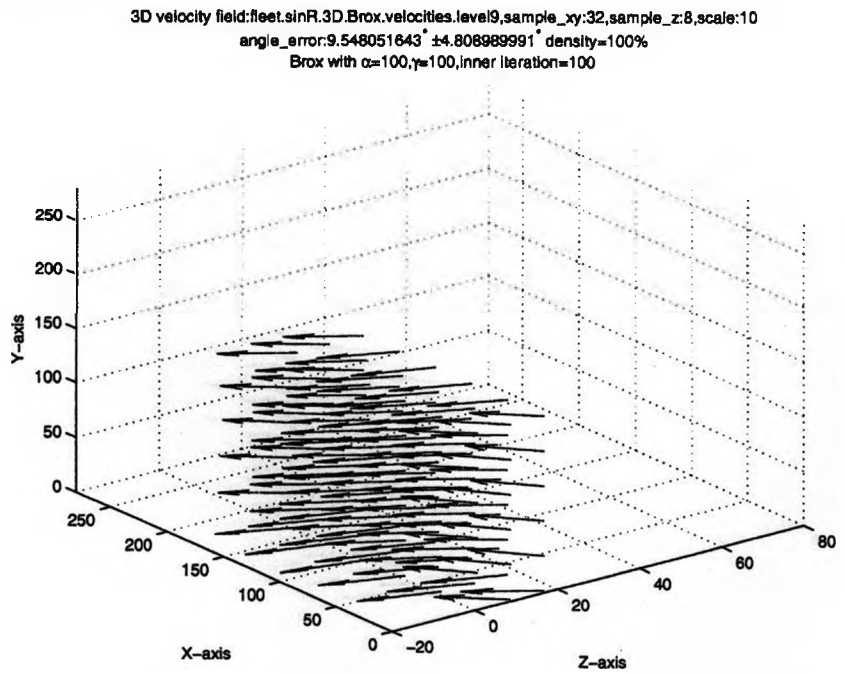


(j)

Figure 5.1: The computed flow field at (i)the 2^{th} *and (j) the* 1^{th} *level of the* 9^{th} *volume of* **sinL** *(*$\alpha = 100.0$, $\gamma = 100.0$, *pyramid level=* 10, *outer iteration=* 1, *inner iteration=* 100, *with* $3*3*3$ *median filter), downsampling rate* 32 *in x and y directions and* 8 *in z direction, scale factor=* 10.

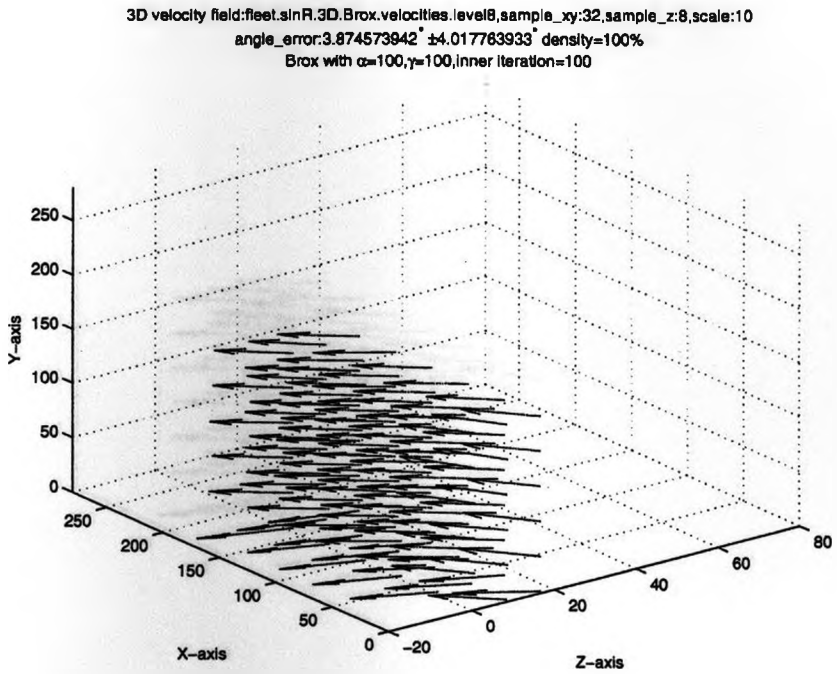


(a)

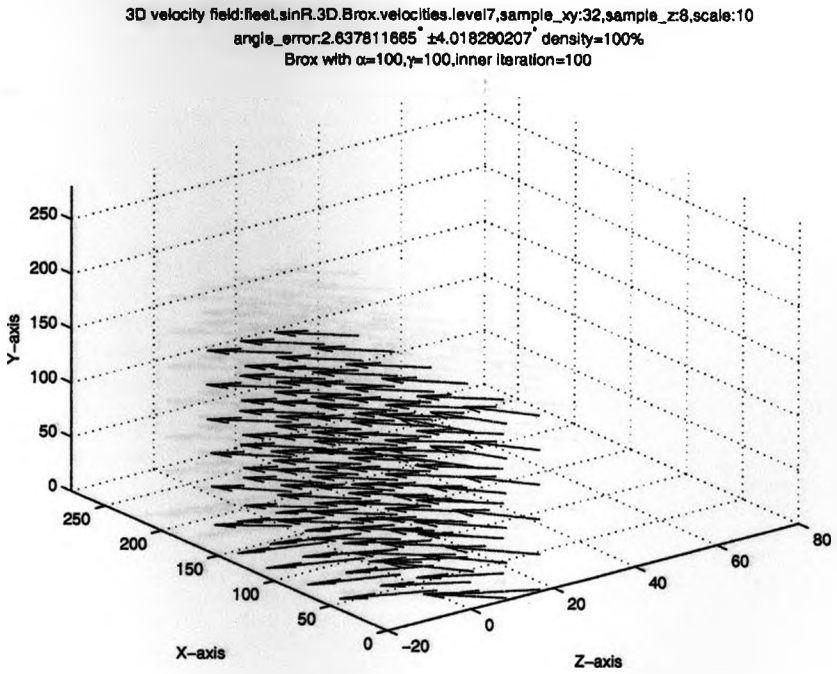


(b)

Figure 5.2: The computed flow field at (i)the 10^{th} *and (j) the* 9^{th} *level of the* 9^{th} *volume of* **sinR** *(*$\alpha = 100.0$, $\gamma = 100.0$, *pyramid level=* 10, *outer iteration=* 1, *inner iteration=* 100, *with* $3*3*3$ *median filter), downsampling rate* 32 *in* x *and* y *directions and* 8 *in* z *direction, scale factor=* 10.

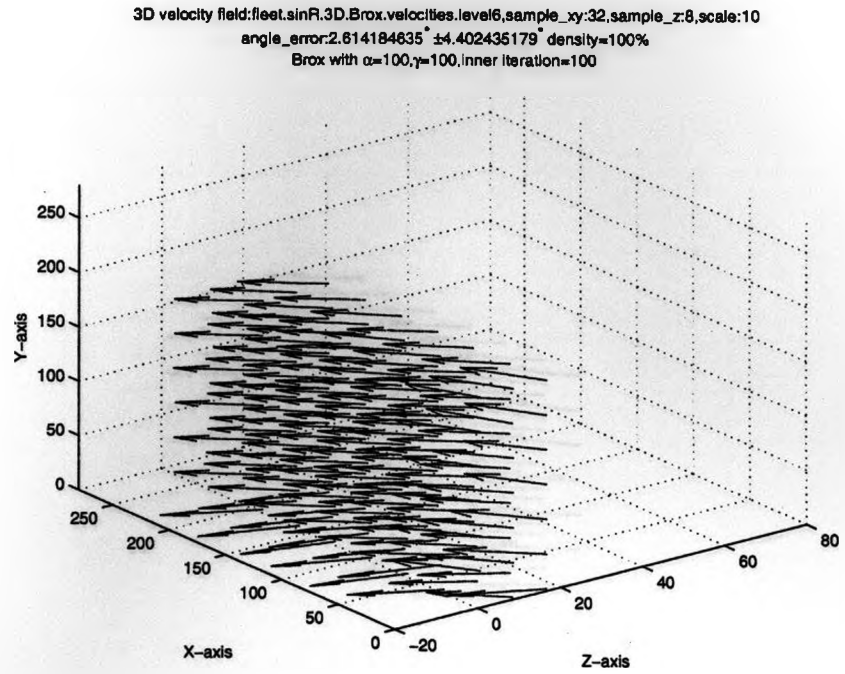


(c)

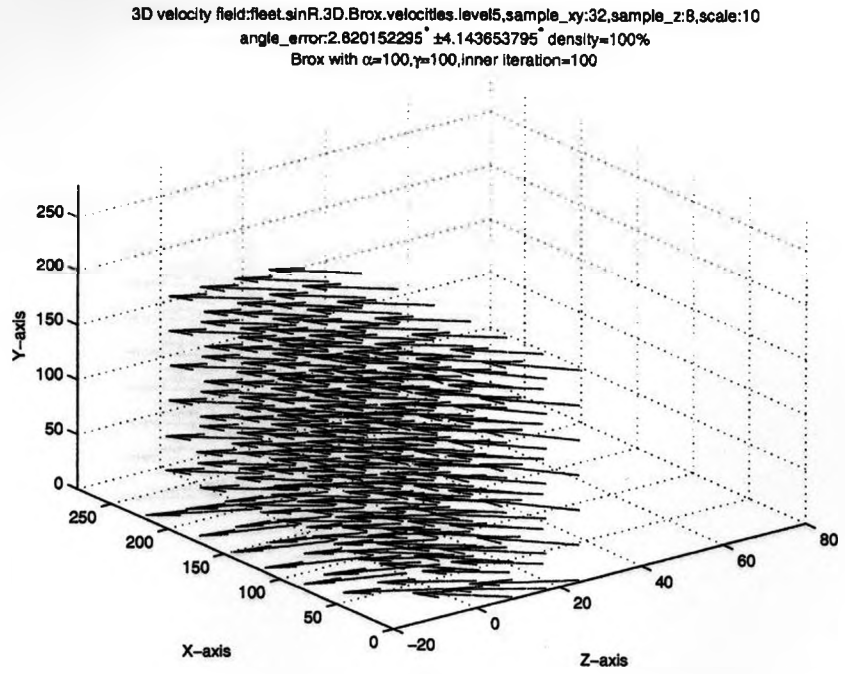


(d)

Figure 5.2: The computed flow field at (i)the 8^{th} and (j) the 7^{th} level of the 9^{th} volume of **sinR** *($\alpha = 100.0$, $\gamma = 100.0$, pyramid level= 10, outer iteration= 1, inner iteration= 100, with $3*3*3$ median filter), downsampling rate 32 in x and y directions and 8 in z direction, scale factor= 10.*

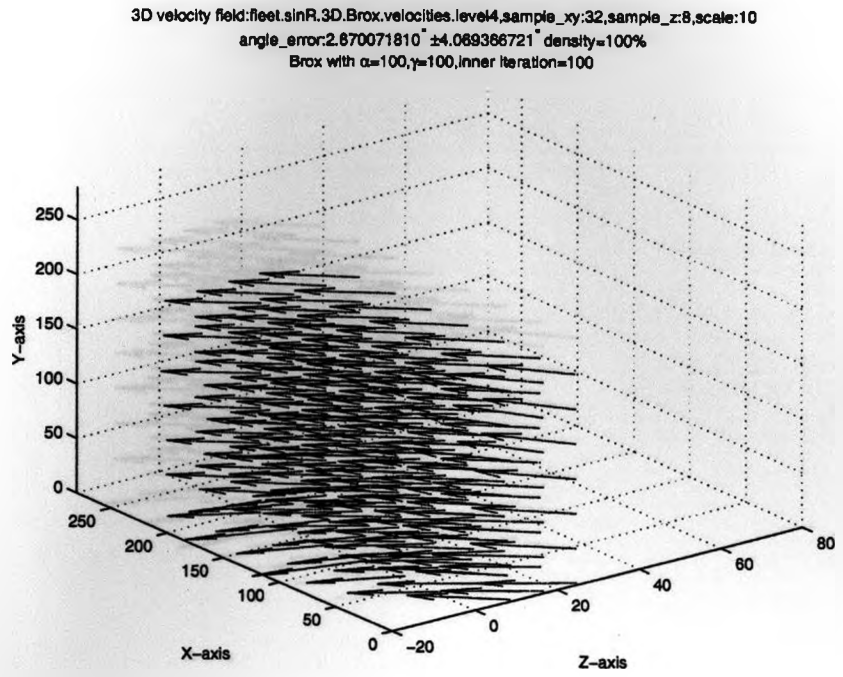


(e)

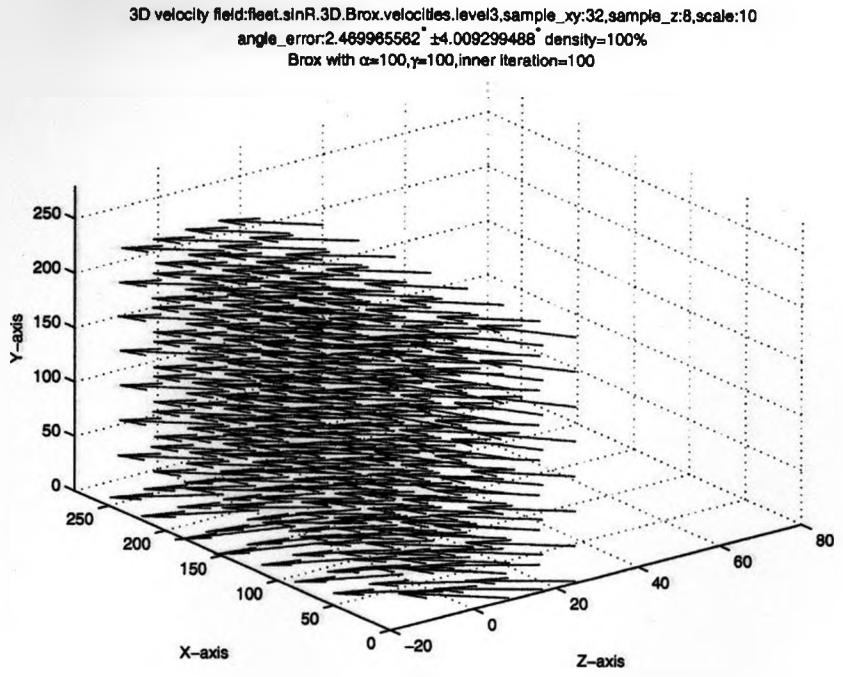


(f)

Figure 5.2: The computed flow field at (i)the 6^{th} and (j) the 5^{th} level of the 9^{th} volume of **sinR** *($\alpha = 100.0$, $\gamma = 100.0$, pyramid level= 10, outer iteration= 1, inner iteration= 100, with $3*3*3$ median filter), downsampling rate 32 in x and y directions and 8 in z direction, scale factor= 10.*



(g)



(h)

Figure 5.2: The computed flow field at (i)the 4^{th} and (j) the 3^{th} level of the 9^{th} volume of **sinR** *($\alpha = 100.0$, $\gamma = 100.0$, pyramid level= 10, outer iteration= 1, inner iteration= 100, with $3*3*3$ median filter), downsampling rate 32 in x and y directions and 8 in z direction, scale factor= 10.*

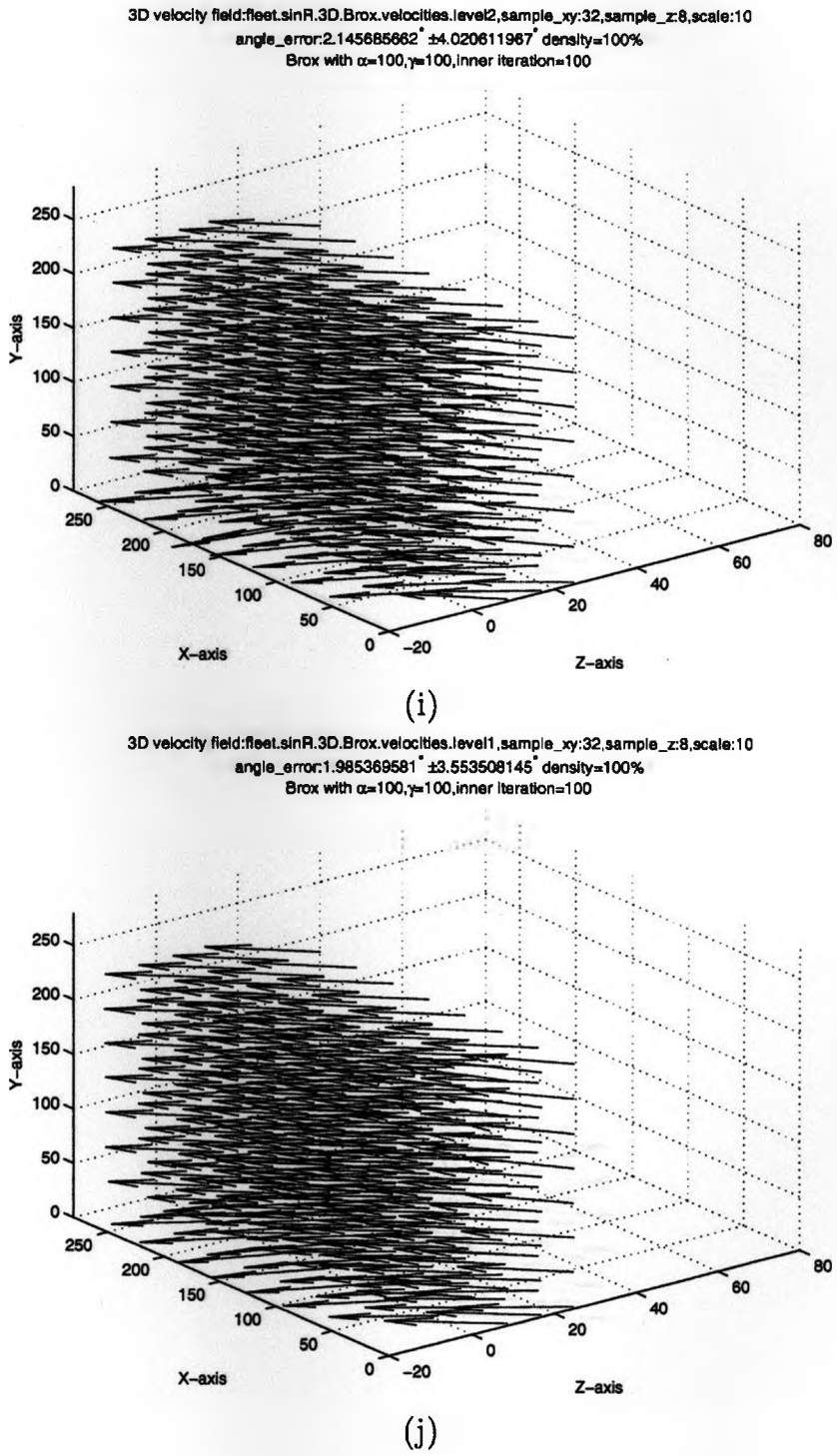


Figure 5.2: The computed flow field at (i)the 2^{th} *and (j) the* 1^{th} *level of the* 9^{th} *volume of* **sinR** *(*$\alpha = 100.0$*,* $\gamma = 100.0$*, pyramid level=* 10*, outer iteration=* 1*, inner iteration=* 100*, with* $3*3*3$ *median filter), downsampling rate* 32 *in* x *and* y *directions and* 8 *in* z *direction, scale factor=* 10*.*

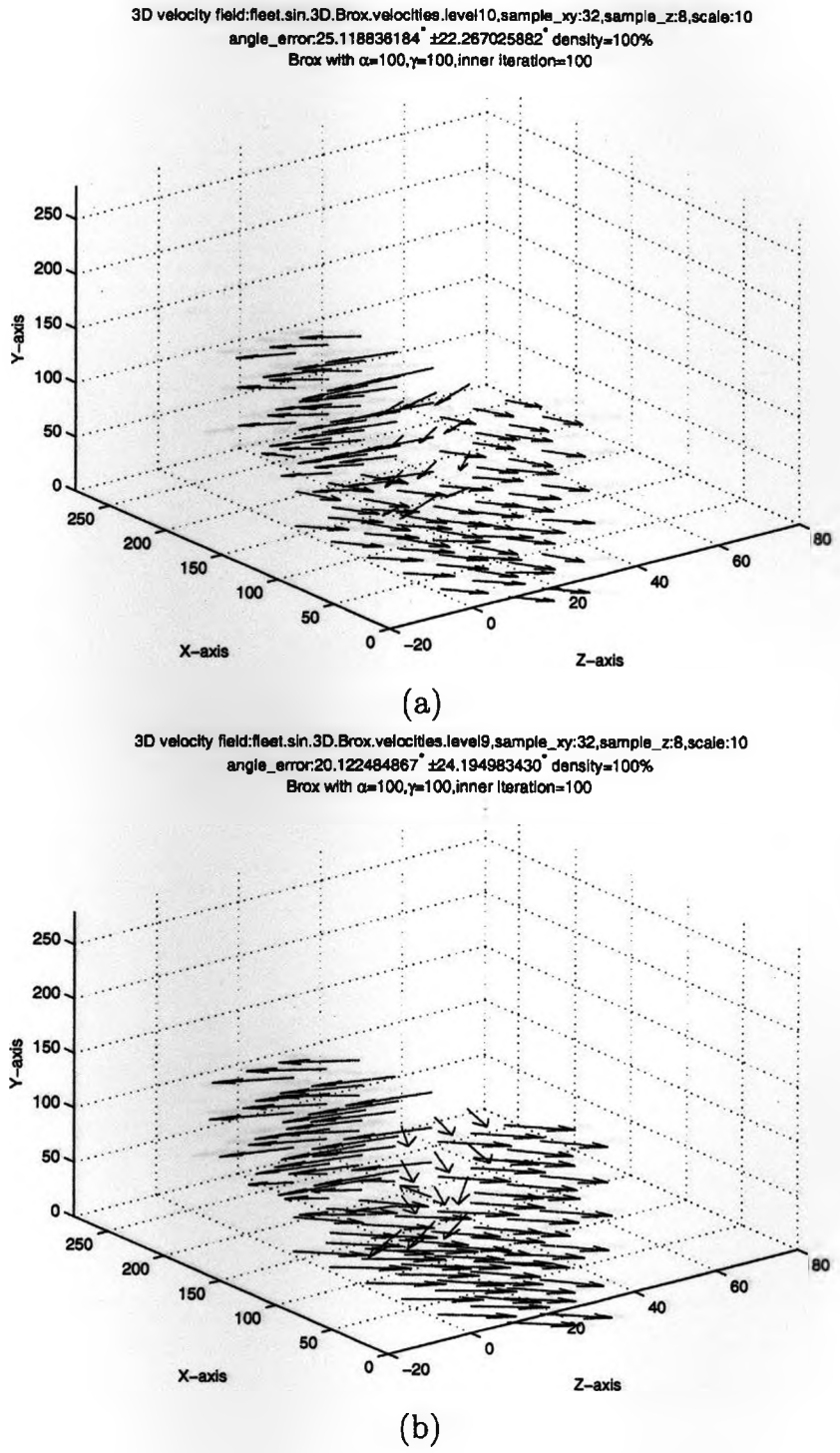
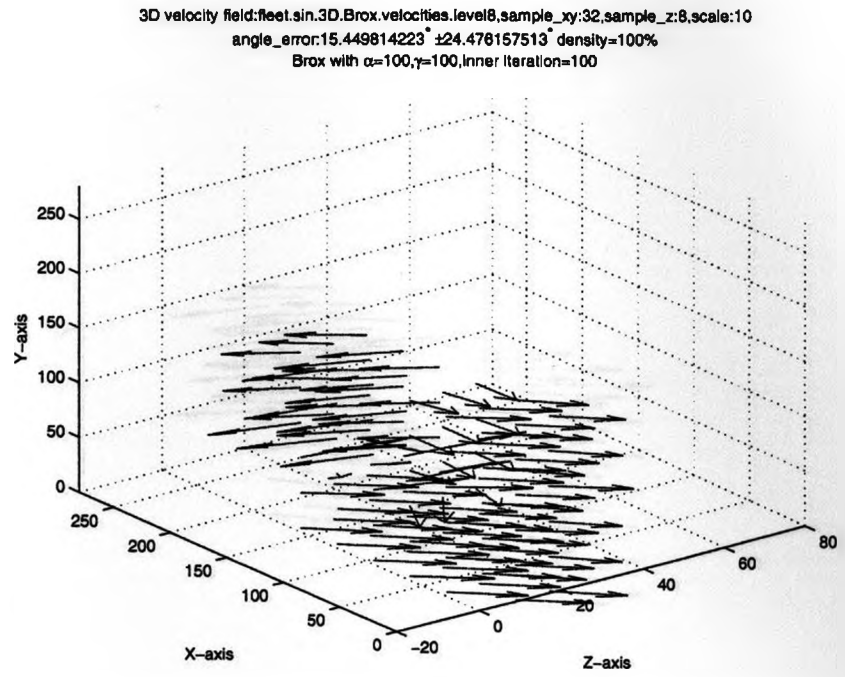
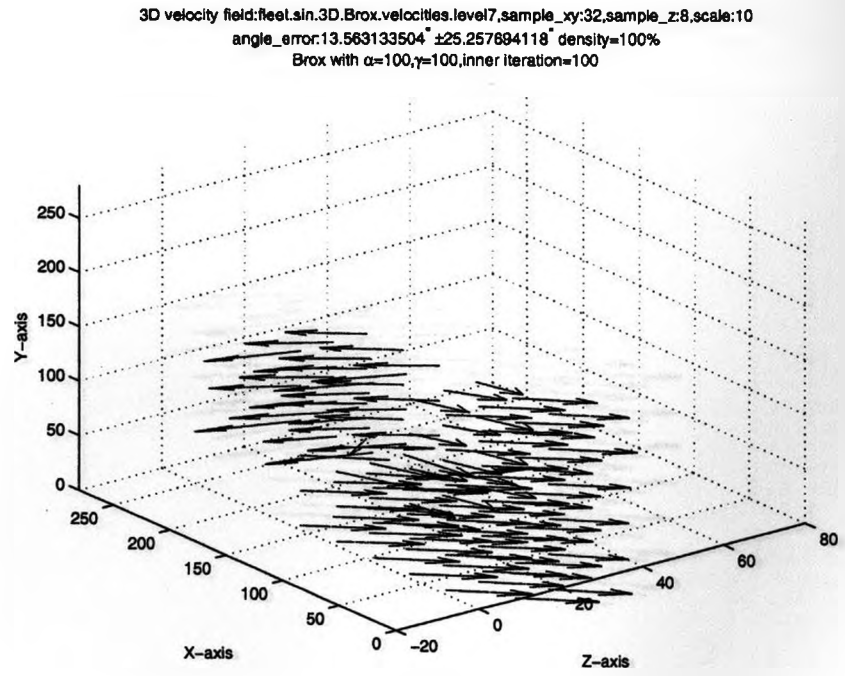


Figure 5.3: The computed flow field at (i)the 10^{th} *and (j) the* 9^{th} *level of the* 9^{th} *volume of* **sin** *(*$\alpha = 100.0$*,* $\gamma = 100.0$*, pyramid level=* 10*, outer iteration=* 1*, inner iteration=* 100*, with* $3*3*3$ *median filter), downsampling rate* 32 *in* x *and* y *directions and* 8 *in* z *direction, scale factor=* 10.

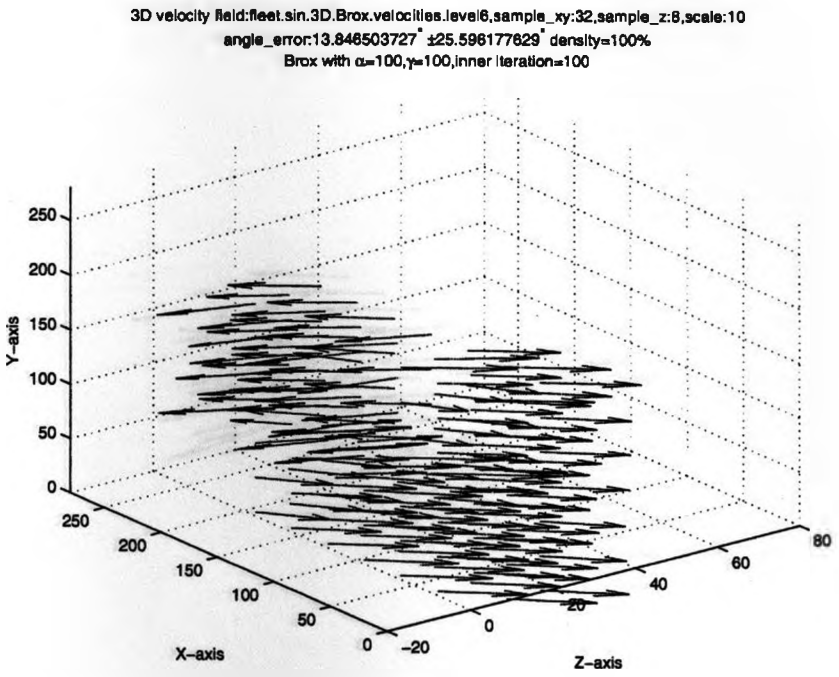


(c)

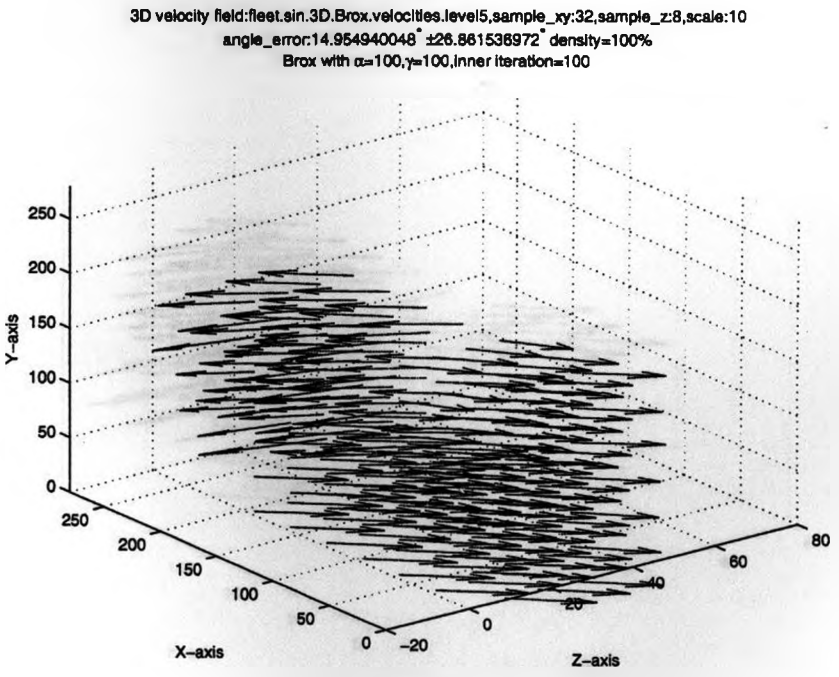


(d)

Figure 5.3: The computed flow field at (i)the 8^{th} and (j) the 7^{th} level of the 9^{th} volume of **sin** *(α = 100.0, γ = 100.0, pyramid level= 10, outer iteration= 1, inner iteration= 100, with 3∗3∗3 median filter), downsampling rate 32 in x and y directions and 8 in z direction, scale factor= 10.*

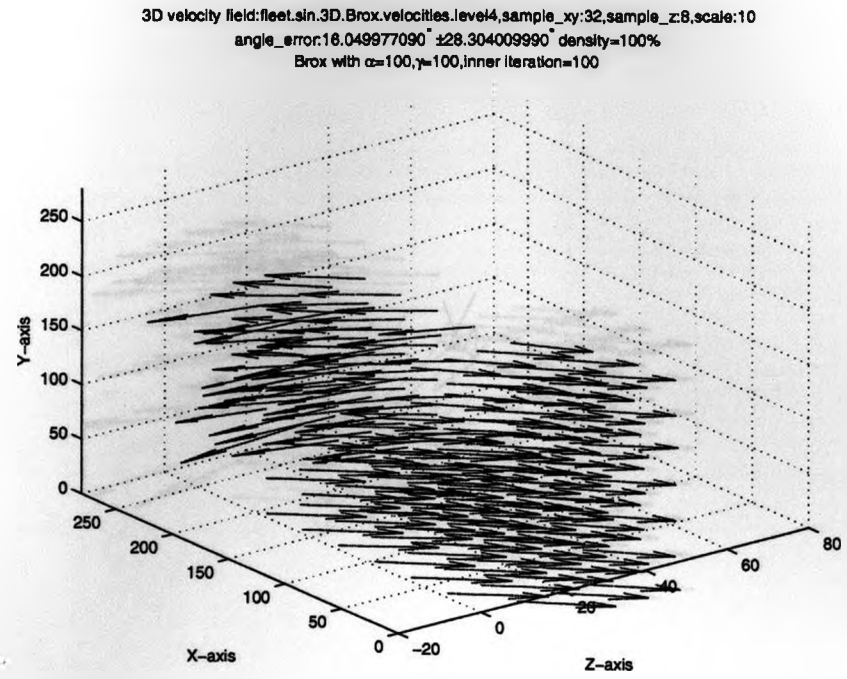


(e)

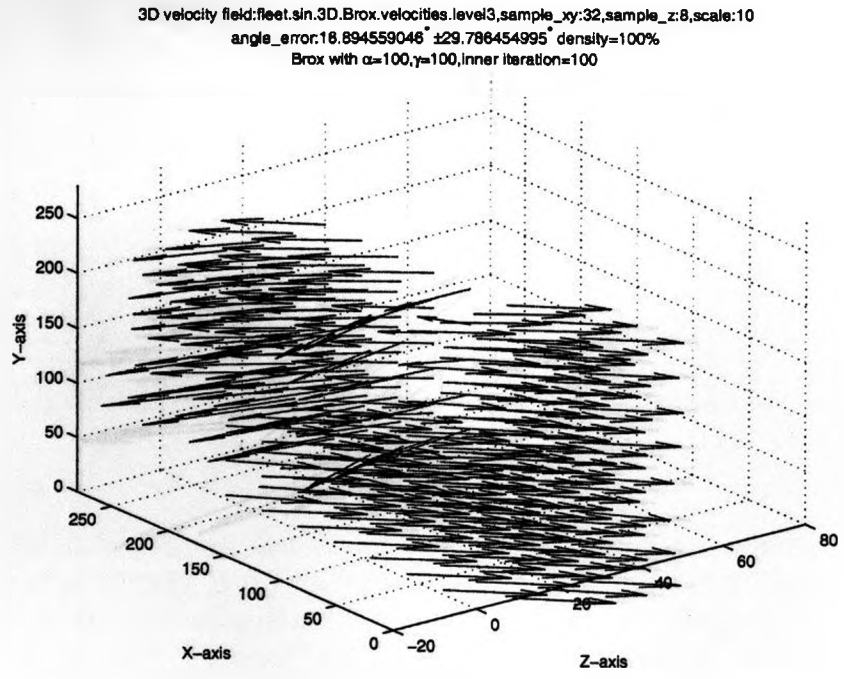


(f)

Figure 5.3: The computed flow field at (i)the 6^{th} *and (j) the* 5^{th} *level of the* 9^{th} *volume of* **sin** *(*$\alpha = 100.0$, $\gamma = 100.0$, *pyramid level=* 10, *outer iteration=* 1, *inner iteration=* 100, *with* $3*3*3$ *median filter), downsampling rate* 32 *in x and y directions and* 8 *in z direction, scale factor=* 10.

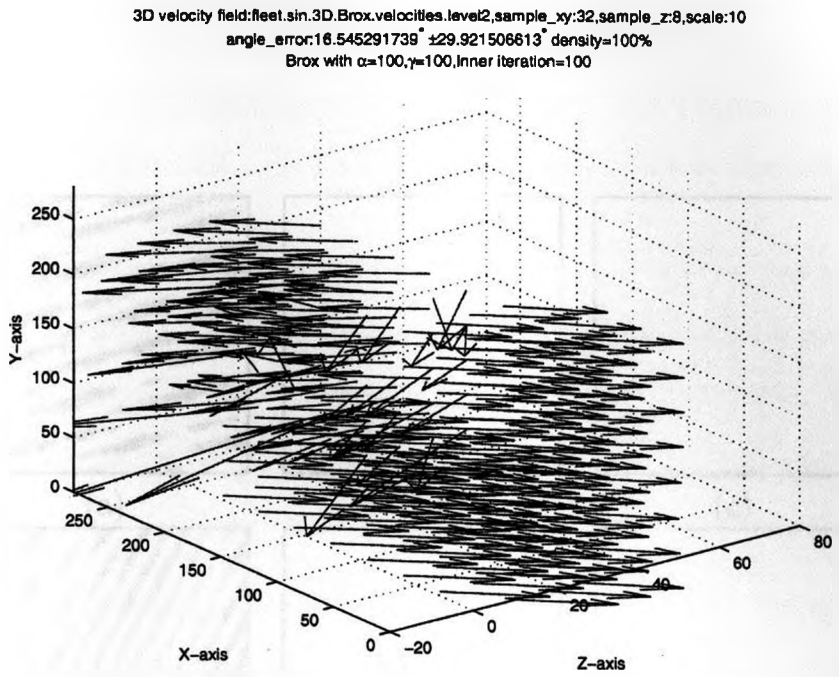


(g)

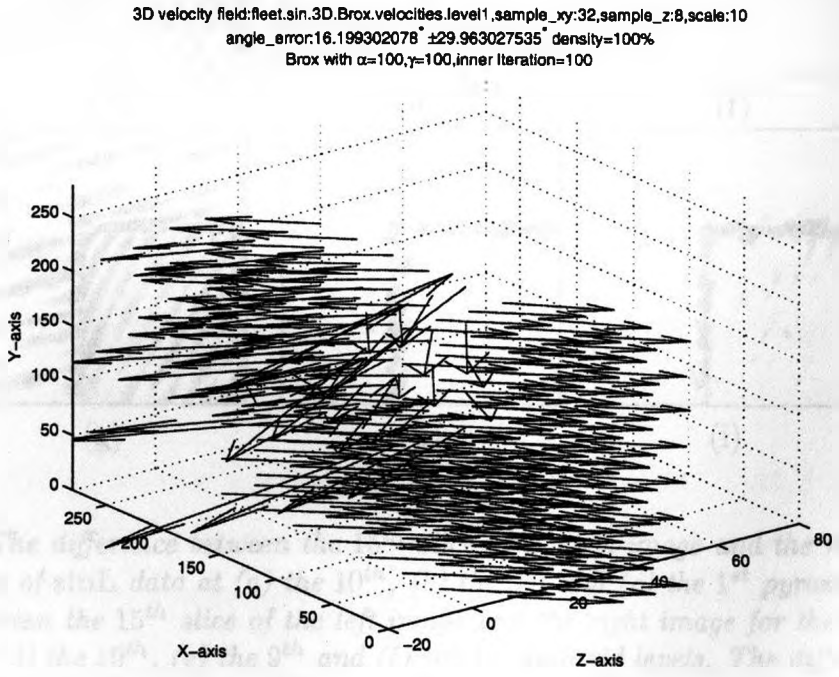


(h)

Figure 5.3: The computed flow field at (i)the 4^{th} and (j) the 3^{th} level of the 9^{th} volume of **sin** *(α = 100.0, γ = 100.0, pyramid level= 10, outer iteration= 1, inner iteration= 100, with $3*3*3$ median filter), downsampling rate 32 in x and y directions and 8 in z direction, scale factor= 10.*



(i)



(j)

Figure 5.3: The computed flow field at (i)the 2^{th} *and (j) the* 1^{th} *level of the* 9^{th} *volume of* **sin** *(*$\alpha = 100.0$, $\gamma = 100.0$, *pyramid level=* 10, *outer iteration=* 1, *inner iteration=* 100, *with* $3*3*3$ *median filter), downsampling rate* 32 *in* x *and* y *directions and* 8 *in* z *direction, scale factor=* 10.

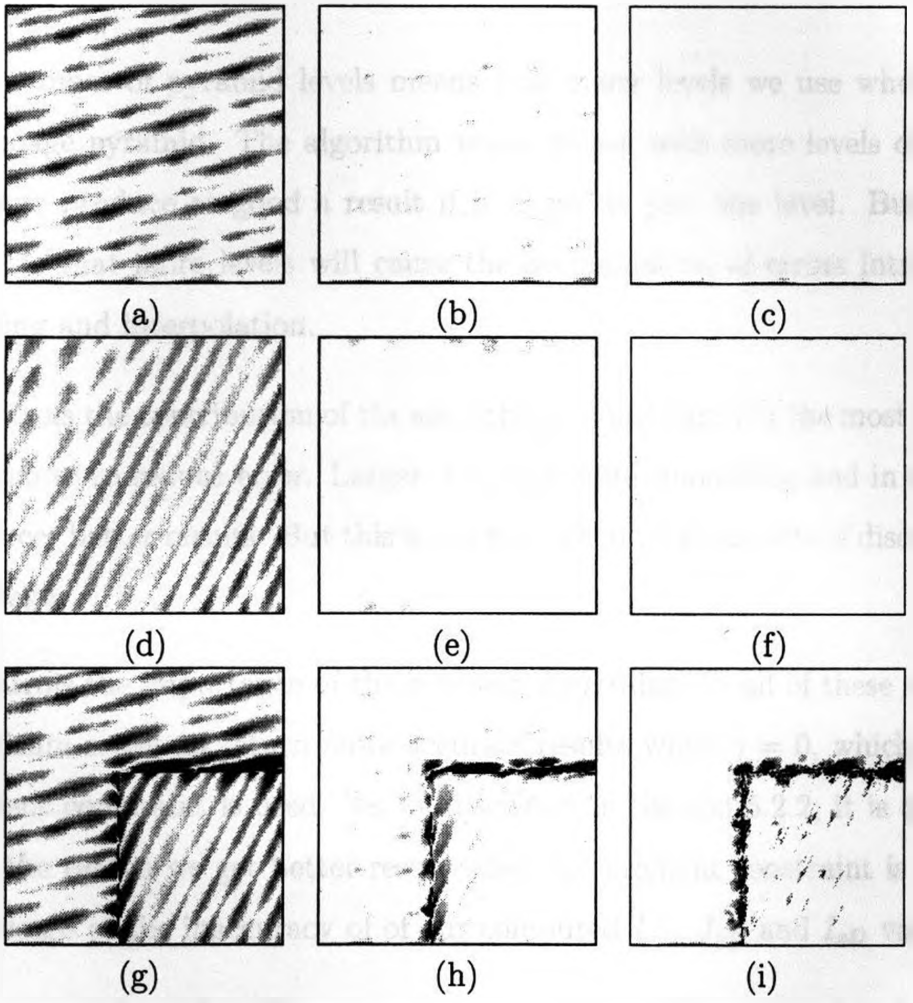


Figure 5.4: The difference between the 15^{th} slice of the left image and the right image for the 9^{th} volume of **sinL** *data at (a) the 10^{th}, (b) the 9^{th} and (c) the 1^{st} pyramid levels. The difference between the 15^{th} slice of the left image and the right image for the 9^{th} volume of* **sinR** *data at (d) the 10^{th}, (e) the 9^{th} and (f) the 1^{st} pyramid levels. The difference between the 15^{th} slice of the left image and the right image for the 9^{th} volume of* **sin** *data at (g) the 10^{th}, (h) the 9^{th}, and (i) the 1^{st} pyramid levels.*

5.2.5 Parameter Variation

We test the variation of parameters for the algorithm and present results for **sinL**, **sinR** and **sin** in Tables 5.29, 5.30 and 5.31. The parameters in the tables are:

- The number of pyramid levels means how many levels we use when building the image pyramid. The algorithm works better with more levels of pyramid. It never produce as good a result if it is run in just one level. But it is also possible that more levels will cause the accumulation of errors introduced by warping and interpolation.

- α controls the contribution of the smoothness constraint it's the most important parameter to reduce error. Larger α means more smoothing and in most cases produces better results. But this is not true when there are lots of discontinuities in images.

- γ controls the importance of the gradient constraint. In all of these results, the algorithm seems to return more accurate results when $\gamma = 0$, which means no gradient constraint is used. As we discussed in Section 5.2.2, it is quite likely that the reason we get better result when the gradient constraint is turned off is because of the inaccuracy of of our computed I_{xD}, I_{yD} and I_{zD} values.

- We employed median filtering so as to remove outliers from a neighborhood with size $N * N * N$. A median filter with larger size does make the result more accurate, but it is extremely time-consuming (see Table 5.36 for computation time).

Parameter				Fleet Error	
Pyramid levels	α	γ	filter size	AAE.	STD.
1	100	100	3	11.88°	2.11°
10	10	0	3	2.18°	2.60°
10	10	100	3	14.10°	11.63°
10	100	0	3	0.65°	1.10°
10	100	0	5	0.66°	1.11°
10	100	100	0	2.28°	2.69°
10	100	100	3	2.06°	2.56°
10	100	100	5	1.69°	2.08°
30	100	0	3	0.65°	1.11°
30	100	100	0	66.95°	31.32°

Table 5.29: Error analysis for parameter variation for **sinL**.

Parameter				Fleet Error	
Pyramid levels	α	γ	filter size	AAE.	STD.
1	100	100	3	26.35°	4.73°
10	10	0	3	1.84°	2.71°
10	10	100	3	16.02°	15.33°
10	100	0	3	0.75°	1.56°
10	100	100	0	2.22°	3.69°
10	100	100	3	1.97°	3.53°
30	100	0	3	0.72°	1.51°

Table 5.30: Error analysis for parameter variation for **sinR**.

Parameter				Fleet Error	
Pyramid levels	α	γ	filter size	AAE.	STD.
1	100	100	3	28.50°	25.08°
10	10	100	3	30.14°	32.89°
10	100	0	3	13.11°	28.71°
10	100	20	3	14.55°	28.27°
10	100	50	3	20.89°	33.53°
10	100	100	0	17.03°	30.40°
10	100	100	3	16.20°	29.69°
30	100	0	3	13.02°	28.73°
30	100	0	3	12.90°	28.83°

Table 5.31: Error analysis for parameter variation for **sin**.

Pyramid level	size_z	size_x	size_y	density(%)	off_z	off_x	off_y
10	19	161	161	73.1727	2	3	3
9	20	169	169	65.1178	3	3	3
8	21	178	178	66.6943	3	3	3
7	22	188	188	68.1592	3	3	3
6	23	198	198	69.5013	3	3	3
5	25	208	208	70.2663	3	4	4
4	26	219	219	71.4058	3	4	4
3	27	231	231	65.5806	4	4	4
2	29	243	243	67.7243	4	4	4
1	31	256	256	68.5104	4	5	5

Table 5.32: Border setting and volume sizes for a 10 levels pyramid with $\eta = 0.95$.

5.2.6 Border Error

To compute the derivatives for pixels at the image boundaries, we use reflection to pad the image. Therefore, we get some poor derivative values around the border for both the 1^{st}-order and 2^{nd}-order derivatives. This creates some error around the boundary. We examine the effect of cutting off the border at each level of the pyramid. Since volumes at different levels are of different sizes, we set offsets based on the size of the volume: $offset_{xy} = floor(pic_{xy} * 0.02)$, $offset_z = floor(pic_z * 0.15)$. Table 5.32 shows the volume size, offsets and flow density for a 10-level pyramid (on image data with size $256 \times 256 \times 31$).

Tables 5.33, 5.34 and 5.35 summarizes the with/without border cutting results for flow fields with the parameters given in Tables 5.29, 5.30 and 5.31. In most of case, border cutting makes the results better.

5.2.7 Running Time

Table 5.36 presented the time complexity of our algorithm running on **sinL** data. For **sinR** and **sin**, running times are similar to those of **sinL**. We run our program on newfie.csd.uwo.ca, which is a Linux (Fedora Core 9.3) machine at the University of Western Ontario, Computer Science department. It is a 64-bit machine with 2.8GHz

without border cutting		with border cutting	
Error	Density	Error	Density
11.88°	100.00	11.82°	68.51
2.18°	100.0	1.87°	68.51
14.10°	100.0	14.16°	68.51
0.65°	100.0	0.55°	68.51
0.66°	100.0	0.55°	68.51
2.28 °	100.0	1.90°	68.51
2.06°	100.0	1.70°	68.51
1.69°	100.0	1.39°	68.51
0.65°	100.0	0.55°	68.51
66.95 °	100.0	67.55°	68.51

Table 5.33: Border error analysis without and with border offsets for **sinL** *with the same parameter changes in Table 5.29.*

without border cutting		with border cutting	
Error	Density	Error	Density
26.35°	100.00	26.57°	68.51
1.84°	100.0	1.53°	68.51
16.02°	100.0	16.32°	68.51
0.75°	100.0	0.58°	68.51
2.22°	100.0	1.82°	68.51
1.97 °	100.0	1.57°	68.51
0.72°	100.0	0.58°	68.51

Table 5.34: Border error analysis without and with border offsets for **sinR** *with the same parameter changes in Table 5.30.*

clock speed and 8GB of main memory. The time is given in hours. As we can see from the table, the most time-consuming part is the $5 \times 5 \times 5$ median filtering, so usually we just use $3 \times 3 \times 3$ median filtering.

without border cutting		with border cutting	
Error	Density	Error	Density
28.50°	100.00	26.01°	68.51
30.14°	100.0	27.80°	68.51
13.11°	100.0	9.35°	68.51
14.55°	100.0	10.91°	68.51
20.89°	100.0	17.90°	68.51
17.03 °	100.0	13.65°	68.51
16.20°	100.0	12.65°	68.51
13.02 °	100.0	9.35°	68.51
12.90°	100.0	9.25°	68.51

Table 5.35: Border error analysis without and with border offsets for **sin** *with the same parameter changes in Table 5.31.*

Pyramid level	Outer-iter.	Inner-iter.	Filter size	Time
10	1	100	3	2.85h
10	1	300	5	25.41h
10	10	10	0	2.35h
10	10	10	3	3.11h
10	10	100	3	19.62h
30	1	100	3	3.60h
30	10	10	3	2.19h

Table 5.36: Time analysis for **sinL** *data(*$256 \times 256 \times 31$ *pixels).*

5.3 Gated MRI Cardiac Datasets Result

At the end of this chapter, we show results for the gated MRI cardiac datasets which we described in Section 4.4. Since the correct flow fields are not available for these datasets, we can only perform qualitative analysis of the flow fields. Figure 5.5(a)-(t) show the 40^{th} slice of volume 1 to volume 20 of the **10phase** dataset. We can see that different parts in heart are contracting/expanding at different rates and times and the heart is undergoing a twisting as a whole. Figure 5.6 shows the flow fields for (a) the 2^{nd}, (b) the 5^{th}, (c) the 9^{th} and (d) the 13^{th} volume of 10phase. Figure 5.7 shows the flow fields for (a) the 2^{nd}, (b) the 5^{th}, (c) the 9^{th} and (d) the 13^{th} volume of 5phase. The flows capture heart motions includes expansion and contraction. Most of the

contraction occurs in the first 4-5 images with the other images exhibiting expansion.

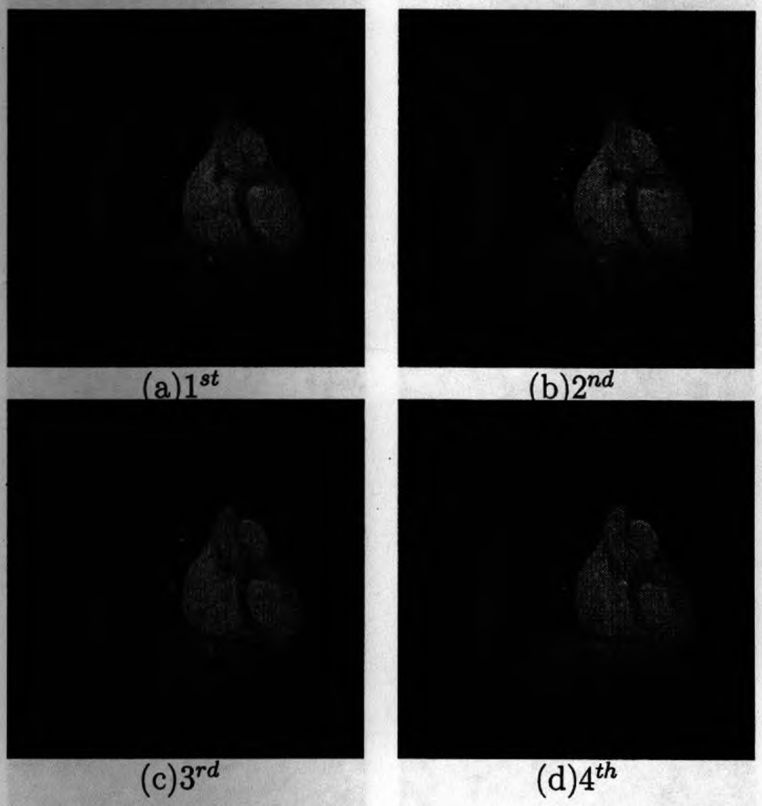


Figure 5.5: MRI cardiac data: (a)-(d) the 40^{th} slice of volume 1 to volume 4 of 10phase.

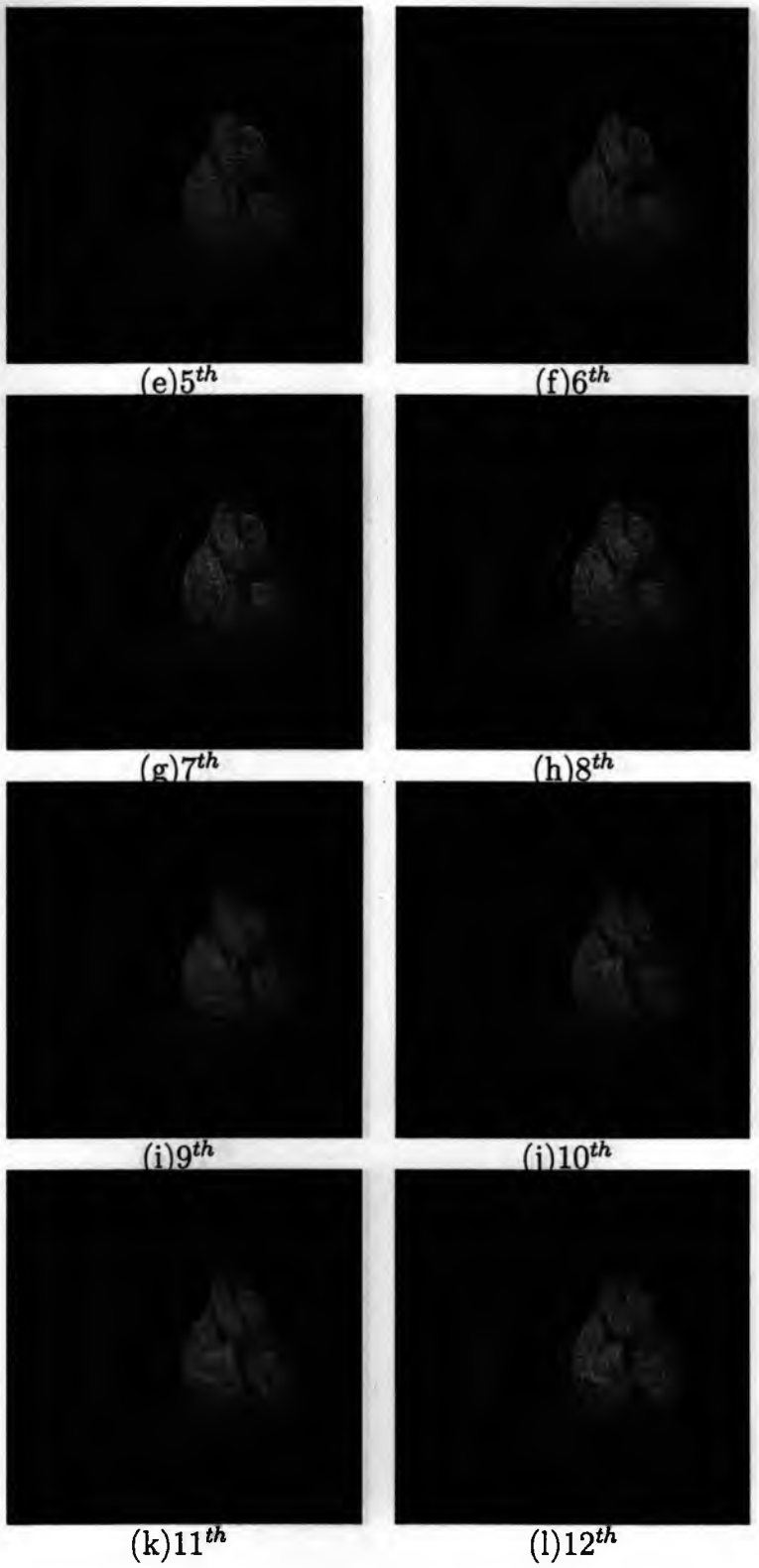


Figure 5.5: MRI cardiac data: (e)-(l) the 40^{th} *slice of volume* 5 *to volume* 12 *of 10phase.*

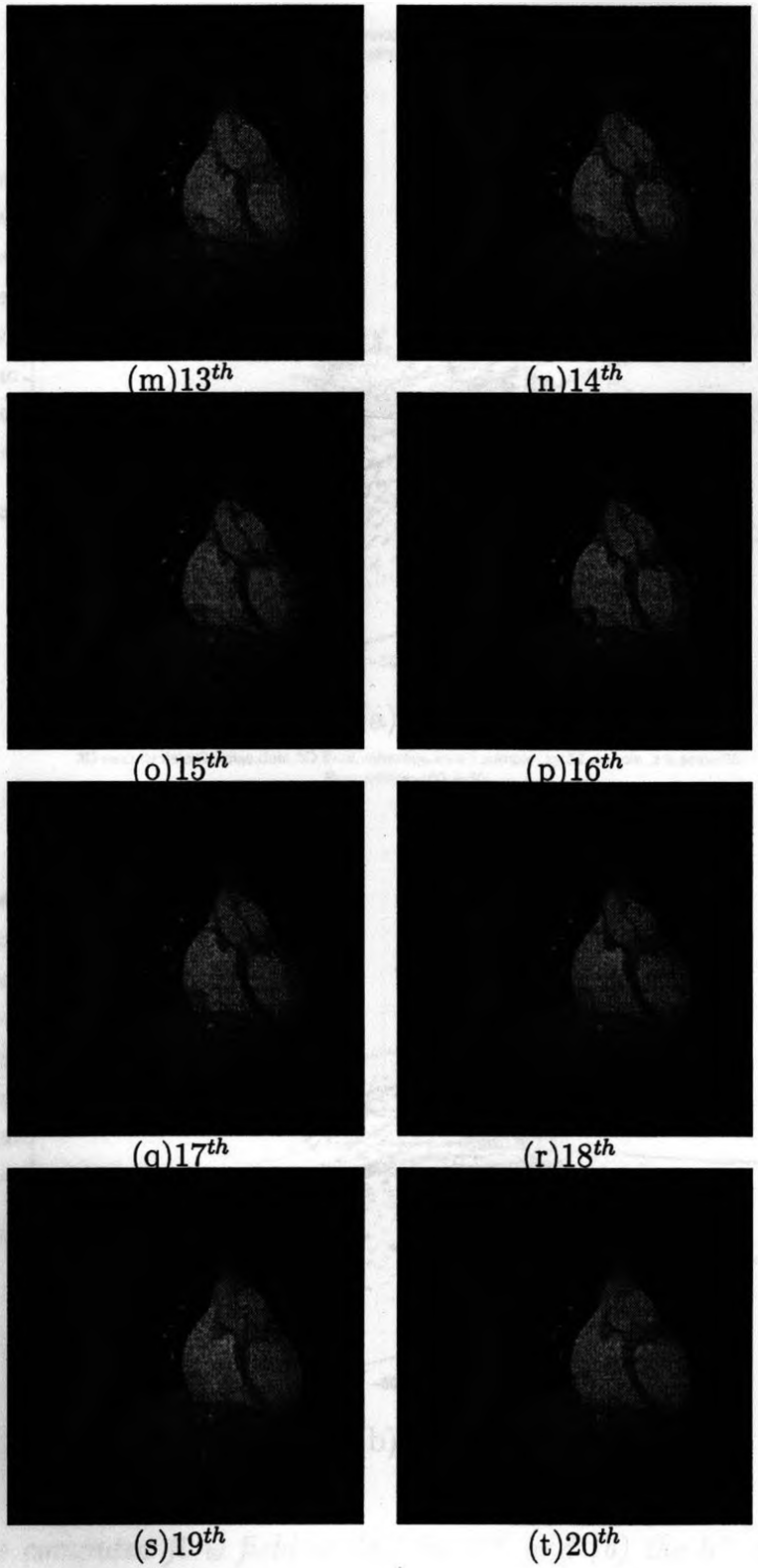
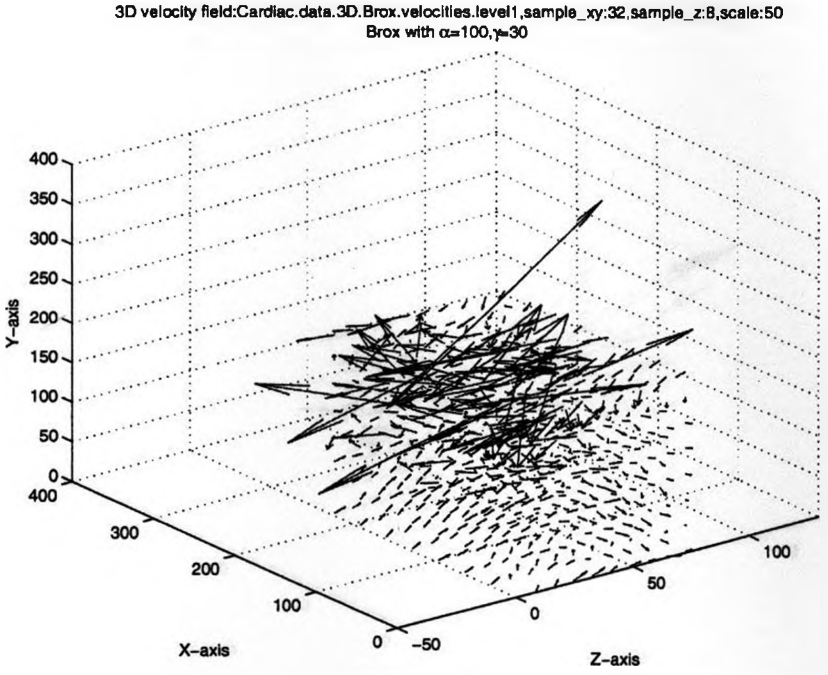
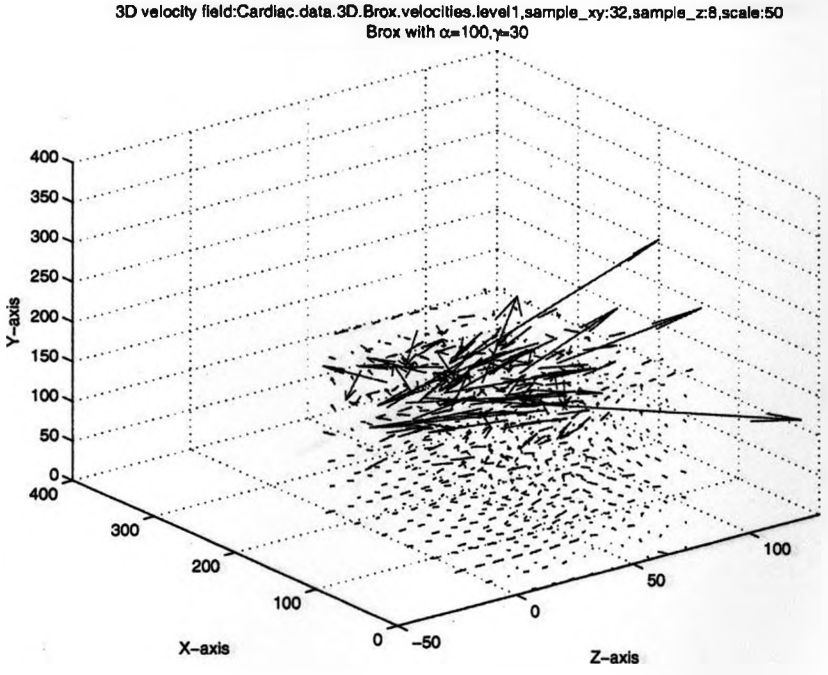


Figure 5.5: MRI cardiac data: (m)-(t) The 40^{th} *slice of volume* 13 *to volume* 20 *of 10phase.*

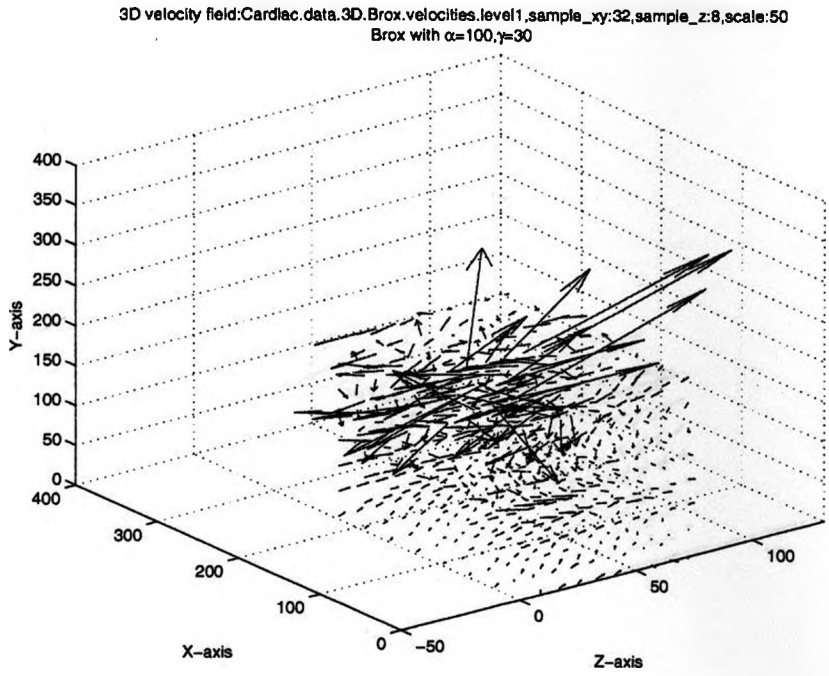


(a)

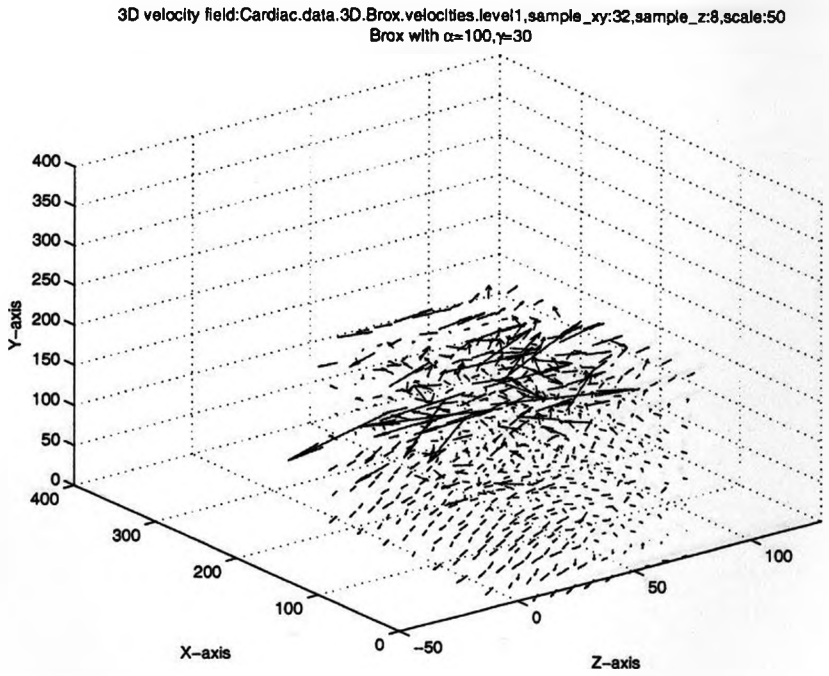


(b)

Figure 5.6: The computed flow field at (a) the 2^{nd} and (b) the 5^{th} volume of gated MRI cardiac data **5phase** *(α = 100, γ = 30,pyramid level= 10,outer iteration= 1, inner iteration= 100,with $3 * 3 * 3$ median filter),downsampling rate 32 in x and y directions, and 8 in z direction, scale factor= 50.*



(c)



(d)

Figure 5.6: The computed flow field at (c) the 9^{th} *and (d) the* 13^{th} *volume of gated MRI cardiac data* **5phase** *(*$\alpha = 100$*,* $\gamma = 30$*,pyramid level=* 10*,outer iteration=* 1*, inner iteration=* 100*,with* $3 * 3 * 3$ *median filter),downsampling rate* 32 *in* x *and* y *directions, and* 8 *in* z *direction, scale factor=* 50*.*

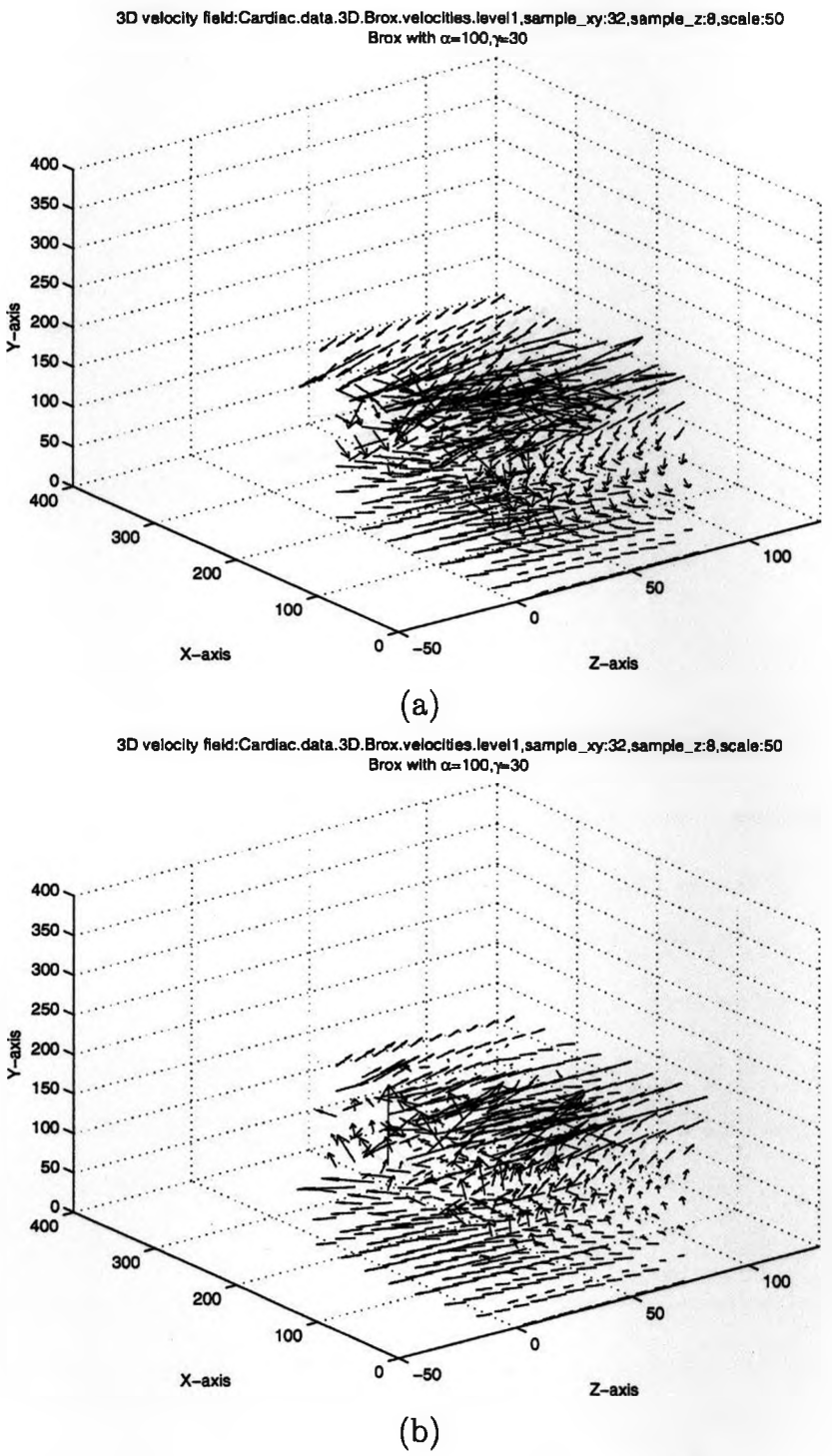
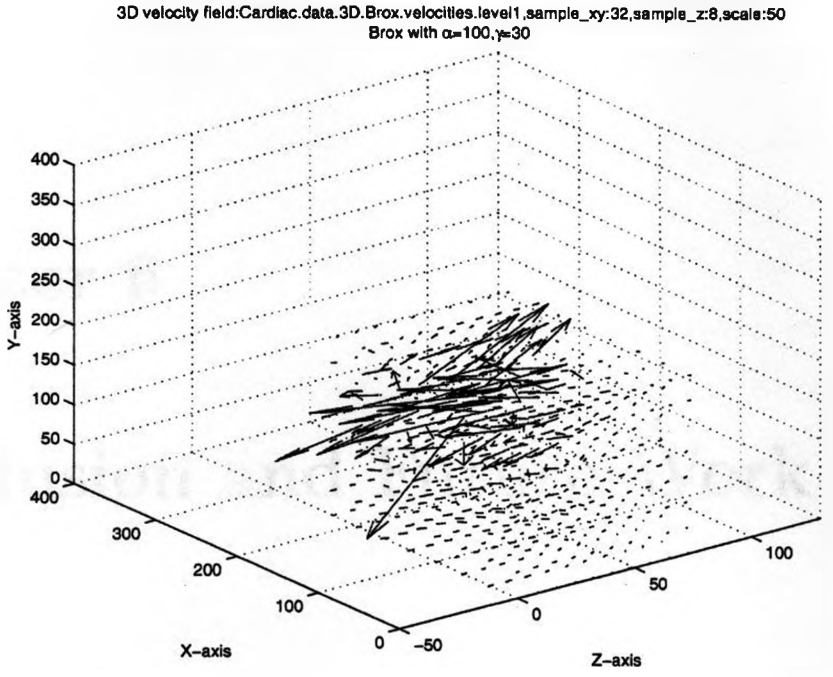
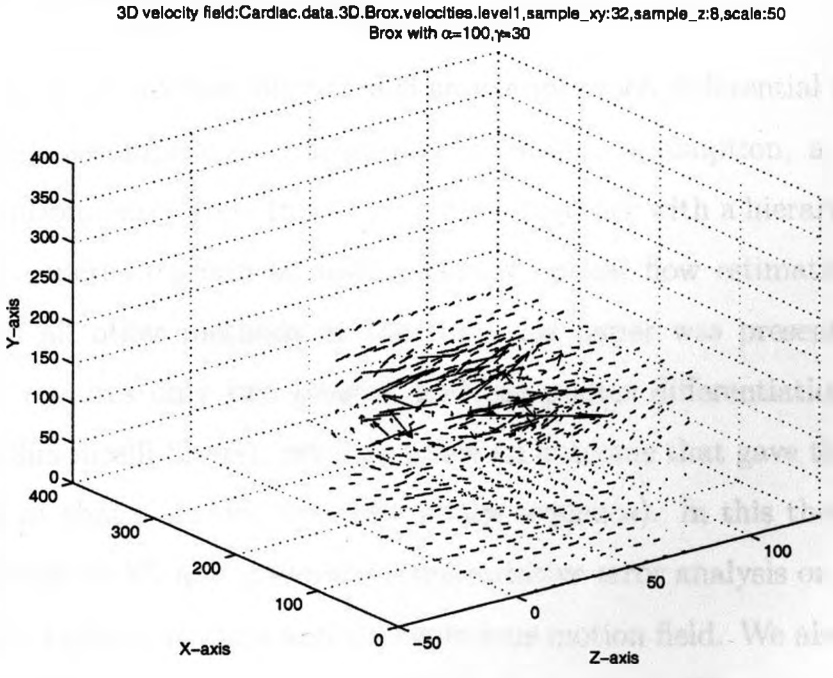


Figure 5.7: The computed flow field at (a) the 2^{nd} *and (b) the* 5^{th} *volume of gated MRI cardiac data* **10phase** *(*$\alpha = 100$*,* $\gamma = 30$*,pyramid level=* 10*,outer iteration=* 1*, inner iteration=* 100*,with* $3 * 3 * 3$ *median filter),downsampling rate* 32 *in* x *and* y *directions, and* 8 *in* z *direction, scale factor=* 50*.*



(c)



(d)

Figure 5.7: The computed flow field at (c) the 9^{th} and (d) the 13^{th} volume of gated MRI cardiac data **10phase** *($\alpha = 100$, $\gamma = 30$,pyramid level= 10,outer iteration= 1, inner iteration= 100,with $3 * 3 * 3$ median filter),downsampling rate 32 in x and y directions, and 8 in z direction, scale factor= 50.*

Chapter 6

Conclusion and Future Work

6.1 Conclusion

The Brox et al.'s optical flow algorithm is an energy-based differential method which integrates three assumptions: a brightness constancy assumption, a gradient constancy assumption, and a smoothness constraint together with a hierarchical warping strategy. This method produced high accuracy optical flow estimation result that out-performed all other methods at the time this paper was presented (in 2004). This method requires only two images, and use poorer differentiation (than those produced by Simoncelli filters), yet it produce an outcome that gave the "best" published results at that time (for Yosemite image sequence). In this thesis, we extend this 2D algorithm to 3D and presented a quantitative error analysis on 3D sinusoidal sequences with both continuous and discontinuous motion field. We also implement a hierarchical 3D Horn and Schunck method for comparison. We implemented our 3D Brox et al's algorithm in MATLAB, and vectorized our code to get high efficiency, since MATLAB is is optimized for matrix operations. In our evaluation, we tested the evolution of flow fields at each level in the multiscale approach. We also tested the influence of parameters that controls the contribution of single constraints to the

total energy. For qualitative evaluation on real data, we produce flow fields for gated MRI cardiac datasets which capture some essential motions of a beating heart.

Our quantitative results shows that 3D Brox et al.'s algorithm obtains best result when the weight γ (for gradient constraint) is set to 0, which means gradient constraint is turned off. As we discussed in Section 5.2.2, the main reason for this may be the poor temporal derivatives we use for I_{xt}, I_{yt}, and I_{zt}, which simply use the difference between the spatial derivatives of the left and right images for the temporal part of the differentiation. One other thing we should mention here is that the Box et al. method without gradient constraint is actually based on two same assumptions as the Horn and Schunck algorithm. However, its results still superior results for hierarchical Horn and Schunck algorithm. This can result from the use of the robust estimator in Brox et al. method.

6.2 Future Work

Future work includes finding out how to improve the temporal derivatives we compute and/or how to get the gradient constraint to work under poor temporal differentiation. Right now our extension is based on the variant method presented by Faisal and Barron [9] in which Crammer's rule is employed as the iterative schema. It would be good to get Brox et al.'s iterative scheme to work. In addition, Papenberg et al. [23] have combined more non-linearized constraints into the model and obtain even better (but not significantly so) results for Yosemite image sequence. We could also try these constraints in our 3D algorithm.

Bibliography

[1] L. Alvarez, J. Weickert, and J. Sanchez. Reliable estimation of dense optical flow fields with large displacements. *International Journal of Computer Vision*, 39(1):41–56, August 2000.

[2] T. Amiaz and N. Kiryati. Piecewise-smooth dense optical flow via level sets. *International Journal of Computer Vision*, 68(2):111–124, 2006.

[3] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2:283–310, 1989.

[4] A. Bab-Hadiashar and D. Suter. Robust optic flow computation. *International Journal of Computer Vision*, 29(1):59–77, 1998.

[5] J. Barron. Experience with 3d optical flow on gated mri cardiac datasets. In *Proceedings of the 1st Canadian Conference on Computer and Robot Vision*, pages 370–377, 2004.

[6] J. Barron, D. Fleet, and S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77, 1994.

[7] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *Proceedings of the 8th European Conference on Computer Vision*, pages 25–36, 2004.

[8] A. Bruhn, J. Weickert, and C. Schnörr. Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61(3):1–21, 2005.

[9] M. Faisal and J. Barron. High accuracy optical flow method based on a theory for warping: Implementation and qualitative/quantitative evaluation. In *Proceedings of the 4th International Conference on Image Analysis and Recognition*, pages 513–525, 2007.

[10] G. Färneback. Very high accuracy velocity estimation using orientation tensors, parametric motion, and simultaneous segmentation of the motion field. In *Proceedings of the 8th IEEE International Conference on Computer Vision*, volume I, pages 171–177, July 2001.

[11] D. J. Fleet and A. D. Jepson. Computation of component image velocity from local phase information. *International Journal of Computer Vision*, 5(1):77–104, 1990.

[12] D. Hanselman and B. Littlefield. *Mastering MATLAB 7*. Pearson Prentice Hall, 2005.

[13] D. J. Heeger. Model for the extraction of image flow. *Journal of the Optical Society of America*, 4(8):1455–1471, 1987.

[14] B. Horn and B. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1-3):185–203, 1981.

[15] S. X. Ju, M. J. Black, and A. D. Jepson. Skin and bones: Multi-layer, locally affine, optical flow and regularization with transparency. In *Proceedings of Computer Vision and Pattern Recognition Conference*, pages 307–314, June 1996.

[16] S.-H. Lai and B.C. Vemuri. Reliable and efficient computation of optical flow. *International Journal of Computer Vision*, 29(2):87–105, 1998.

[17] H.C. Longuet-Higgins and K. Prazdny. The interpretation of a moving retinal image. *Proceedings of the Royal Society of London B, Biology Sciences*, 208(1173):385–397, July 1980.

[18] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[19] E. Mémin and P. Pérez. Hierarchical estimation and segmentation of dense motion fields. *International Journal of Computer Vision*, 46(2):129–155, 2002.

[20] J. Moore, M. Drangova, M. Wiergbicki, J. Barron, and T. Peters. A high resolution dynamic heart model. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI), LNCS 2878*, volume 1, pages 549–555, 2003.

[21] H. H. Nagel. Displacement vectors derived from second-order intensity variations in image sequences. *Computer Vision, Graphics, and Image Processing*, 21:85–117, 1983.

[22] T. Nir, A. M. Bruckstein, and R. Kimmel. Over-parameterized variational optical flow. *International Journal of Computer Vision*, 76(2):205–216, 2008.

[23] N. Papenberg, A. Bruhn, T. Brox, S. Didas, and J. Weickert. Highly accurate optic flow computation with theoretically justified warping. *International Journal of Computer Vision*, 67(2):141–158, April 2006.

[24] R. Woods R. Gonzalez and S. Eddins. *Digital Image Processing Using MATLAB*. Prentice-Hall, Inc., 2003.

[25] S. Roth and M. J. Black. On the spatial statistics of optical flow. In *Proceedings of the Tenth IEEE International Conference on Computer Vision*, volume 1, pages 42–49, 2005.

[26] P. Sand and S. J. Teller. Particle video: Long-range motion estimation using point trajectories. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2195–2202, 2006.

[27] A. Singh. An estimation-theoretic framework for image-flow computation. In *Proceedings of the 3rd International Conference on Computer Vision*, pages 168–177, 1990.

[28] S. Uras, F. Girosi, A. Verri, and V. Torre. A computational approach to motion perception. *Biological Cybernetics*, 60:79–97, 1988.

[29] A. M. Waxman, J.Wu, and F. Bergholm. Convected activation profiles and receptive fields for real time measurement of short range visual motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 717–723, 1988.