An Efficient Motion Estimator with Application to Medical Image Registration *

B. C. Vemuri[1], S. Huang[1], S. Sahni[1],
C. M. Leonard[2], C. Mohr[3]
R. Gilmore[4] and J. Fitzsimmons[5]

[1]Department of Computer & Information Sciences
[2]Department of Neuroscience
[3] Department of Neuclear Engineering
[4]Department of Neurology
[5]Department of Radiology
University of Florida, Gainesville, Fl 32611

Keywords: Image Registration, Optical Flow, SSD, B-splines, Newton Iteration, Levenburg Marquardt, Brain Images, MRI.

*This work was supported in part by the grant NIH-R01-LM05944. Submitted to the journal of Medical Image Analysis.

Abstract

Image registration is a very important problem in computer vision and medical image processing. Numerous algorithms for registering single and multi-modal image data have been reported in these areas. Robustness as well as computational efficiency are prime factors of importance in image data registration. In this paper, a robust/reliable and efficient algorithm for estimating the transformation between two image data sets taken from the same modality over time is presented. Estimating the registration between two image data sets is formulated as a motion estimation problem. We use an optical flow motion model which allows for both global as well as local motion between the data sets. In this hierarchical motion model, we represent the flow field with a B-spline basis which implicitly incorporates smoothness constraints on the field. In computing the motion, we minimize the expectation of the squared differences energy function numerically via a modified Newton iteration scheme. The main idea in the modified Newton method is that we precompute the Hessian of the energy function at the optimum without explicitly knowing the optimum. This idea is used for both global and local motion estimation in the hierarchical motion model. We present examples of motion estimation on synthetic and real data and compare the performance of our algorithm with that of competing ones.

1 Introduction

Image registration is a very common and important problem in diverse fields such as medical imaging, computer vision, video compression, art, entertainment etc. The problem of registering two images, be they 2D or 3D, is equivalent to estimating the motion between them. There are numerous motion estimation algorithms in the computer vision literature (Aggarwal and Nandhakumar, 1989; Horn and Schunk, 1981; Barron *et al.*, 1994; Black and Anandan, 1993; Duncan and Chou, 1992; Nagel and Enkelmann, 1986; Szeliski and Coughlan, 1994; Lai and Vemuri, 1995) that could potentially be applied to the problem of registration of volume images, specifically, MR brain scans, which is the topic of focus in this paper. We draw upon this large body of literature of motion estimation techniques for problem formulation but *develop a new numerical algorithm for robustly and efficiently computing the motion parameters. Our usage of the word robust throughout this paper does not have any statistical connotation.*

A large body of literature exists on registration, especially for medical image data registration. Generally, they can be cast into two categories. One is feature-based approach. Pellizari *et al.* (Pelizarri *et al.*, 1989) have developed a method for the registration of two brains identified from two different data sets by minimizing the distance between the surfaces measured from a single central point. The minimization is carried out numerically

using Powell's method. Their measure of distance between surfaces is appropriate only for spherical shapes. Evans *et al.* (Evans *etl al.*, 1991) developed a registration scheme based on approximating the 3D warp between the model and target image by a 3D thin plate spline fitted to landmarks. Szeliski *et al.* (Szeliski and Coughlan, 1994) developed a fast matching algorithm for registration of 3D anatomical surfaces found in 3D medical image data. They use a distance metric minimization to find the optimal transformation between the surfaces. The key difference in their scheme from earlier methods is the use of tensor product splines to represent different registration transformations, namely rigid, affine, trilinear, quadratic etc. In addition, they also introduced the novel concept of octree splines for very fast computation of distance between surfaces. Davatzikos *et al.* (Davatzikos and Prince, 1994) introduced a two stage brain image registration algorithm. The first stage involved using active contours to establish a one-to-one mapping between cortical and ventricular boundaries in the brain and in the second stage, an elastic deformation transformation that determines the best correspondence between the identified contours in the two data sets is determined. Feldmar et al., (Feldmar and Ayache, 1994) developed a novel surface to surface nonrigid registration scheme, using locally affine transformation. Maintz et al., (Maintz *et al.*,) compared edge (Borgefors, 1989) and ridge-based (Gueziec and Ayache, 1992; Monga *et al.*,; van den Elsen, 1993) registration of CT and MR brain images. They describe a novel method that makes use of fuzzy edgeness and ridgeness images in achieving the registration between MR and CT data sets. All of these approaches have one commonality, i.e., they need to detect features/surface/contours in the images and hence the accuracy of registration is dictated by the accuracy of the feature detector. Also, additional computational time is required for detecting these features prior to application of the actual registration scheme. *In this paper, we propose a registration method that is robust and fast and is applicable directly to the raw data.*

The alternative class of registration algorithms involves using the direct approach, i.e., determining the registration transformation directly from the image data. One such straightforward approach is locally adaptive correlation window scheme (Okuomi and Kanade, 1992), which can achieve high accuracy. But it is computationally demanding (Szeliski and Coughlan, 1994). A more general scheme than window-based correlation approach is the optical

flow formulation, in which the problem of registering two images is treated as equivalent to computing the flow between the data sets. There are numerous techniques for computing the optical flow from a pair of images (Horn and Schunk, 1981; Barron *et al.*, 1994; Duncan and Chou, 1992; Lai and Vemuri, 1995; Szeliski and Coughlan, 1994; Black and Anandan, 1993; Gupta and Prince, 1995). Zhao et al., (Zhao, 1993) present an optical flow based registration scheme for aligning/registering coronal sections to form a 3D stack of registered auto-radiographic images. In their algorithm, they assume the flow is parameterized by a general affine parameters and use linear regularization constraints. The registration is performed in 2D yielding a stack of registered 2D coronal slices.

Another direct approach is based on the concept of maximizing mutual information reported in Viola and Wells (Viola and Wells, 1995) and Wells *et al.*, (Wells III *et al.*, 1996), Collignon et al., (Collignon *et al.,* 1995) and Studholme et al., (Studholme *et al.,*). Mutual information between the model and the image that are to be registered is maximized using a stochastic analog of the gradient descent method in (Wells III *et al.*, 1996) and other optimization methods such as the Powells method in (Collignon *et al.,* 1995) and a multiresolution scheme in (Studholme *et al.,*). The primary computational task in the mutual information based techniques is the estimation of probability density functions and their derivatives from the given data sets. The density function estimation has typically been performed using the Parzen window approach (Duda and Hart, 1973). Derivatives of the entropy whose definition involves densities are required when maximizing mutual information between the data sets. Once the densities are estimated, derivatives of the empirical entropy are easily estimated. Reported registration experiments in these works are quite impressive for the case of rigid motion. Most mutual information based algorithms for image registration in literature have been formulated for global parameterized motion with the exception of the recent work reported in Meyer et al., (Meyer97, 1997) wherein affine transformations as well as thin-plate spline warps are handled. The reported registration results are quite impressive but the CPU execution times are quite high – in the order of several hours for estimating thin-plate warps.

Bajscy and Kovacic (Bajscy and Kovacic, 1989) studied registration under nonrigid deformations using volumetric deformations based on elasticity theory of solids. Other direct approaches have used the so called "fluid registration" approach introduced by Christensen

et al. (Christensen *el al.*, 1996). In this approach the registration transformation is modeled by a viscous fluid flow model expressed as a partial differential equation (PDE). The model primarily describes the registration as a local fluid deformation expressed by a nonlinear PDE and more recently by a linear version (Bro-Nielsen and Gramkow, 1996). Thirion (Thirion, 1994) introduced an interesting demon-based registration that can be viewed as being similar to the fluid-based algorithm when the elastic filter in the latter is replaced by a Gaussian. In all these approaches, the transformation is not expressible in a parametric form and hence it is not possible to explicitly determine the individual components of the transformation no matter how simple that transformation may be. Moreover, the reported computational times for registration of 2D and 3D data sets are very large for implementation on uniprocessor workstations. The fastest implementation of the viscous fluid flow model by Schormann *et al.* (Schormann *et al.*, 1996) using a multi-grid scheme takes 200 mins. of CPU time on a Sparc-10 for registering two $(64, 128, 128)$ images.

In summary, the existing registration algorithms either can't be applied to raw image/volume data directly, or can't handle local deformation and global transformation at the same time, or require extensive computation. In this paper, a robust and fast algorithm is proposed, which incorporates a Modified Newton iterative method into a spline-based optical flow framework and achieves accurate registration results in registration applications involving medical data.

In our approach, we use the hierarchical motion model, a hybrid of local and global flow, to compute the registration, taking advantage of the best features of both approaches. This model consists of globally parameterized motion flow models at one end of the "spectrum" and a local motion flow model at the other end. The global motion model is defined by associating a single global motion transformation within each patch of a recursively subdivided input image, whereas, in the local flow model, the flow field is not parameterized. The flow field corresponds to the displacement at each pixel/voxel which is represented in a B-spline basis. Using this spline-based representation of the flow field (u_i, v_i) in the expectation of the squared differences error term, i.e., $E_{sd}(u_i, v_i) = E[I_2(x_i + u_i, y_i + v_i) - I_1(x_i, y_i)]^2$, the unknown flow field may be estimated at each pixel/voxel via numerical iterative minimization of E_{sd}. I_2 and I_1 denote the target and initial reference images, respectively. We use a combination of a

quasi-Newton and a modified Newton scheme. The quasi-Newton algorithms is well known in numerical analysis literature (Gill *et al.*, 1981) and the modified Newton technique developed by Diehl (Diehl and Burkhardt, 1989) will be described in a subsequent section. Briefly, this modification involves precomputing the Hessian matrix at the optimum – prior to starting the iteration – and using it at each iteration point rather than computing the Hessian or its approximation at each iteration point as is done in the standard and most modified Newton schemes. This precomputation of the Hessian leads to tremendous computational advantages and in addition, the convergence range of this modified Newton scheme is much larger than the standard Newton scheme (see (Diehl and Burkhardt, 1989) for details). When combined with the spline-based representation of the (u, v) field, the motion computation is very stable as is evident in the examples presented in this paper.

The main contributions of this paper are as follows: (a) a new formulation of the hierarchical flow field computation model based on the idea of precomputing the Hessian at the optimum prior to knowing the optimum, (b) development of a novel, fast and robust numerical solution technique using a Modified Newton iteration, wherein the computation of the Hessian matrix and gradients use a spline-based representation of the (u, v) field, (c) a novel application, namely, a volumetric flow-based registration of MRI brain scans.

The rest of the paper is organized as follows. In section 2, we describe the global/local flow field motion model briefly. Section 3 contains the new formulation of the flow field model describing the use of precomputed Hessian and the numerical algorithm used for computing the motion/registration. In Section 4, we present several experimental results on synthesized motion applied to real brain MRI data as well as results of registering MR brain scans of the same individual. Section 5 contains the conclusions.

2 Local/Global motion model

Optical flow computation has been a very active research area in the field of computer vision for the last two decades. This model of motion computation is very general especially when set in a hierarchical framework. In this framework, at one extreme, each pixel/voxel is assumed to undergo an independent displacement, thus producing a vector field of displacements over

the image. This is labeled as a local motion model. At the other extreme, we have global motion wherein the flow field is expressed parametrically by a small set of parameters, e.g., rigid motion, affine motion, and so on.

A general formulation of image registration (same modality data) can be posed as follows. Given a pair of images (possibly from a sequence) I_1 and I_2, we assume that I_2 was formed by locally displacing the reference image I_1 as given by

$$I_2(x_i + u_i, y_i + v_i) = I_1(x_i, y_i). \tag{1}$$

The problem of registering I_1 and I_2 is to recover the displacement field (u, v) for which the maximum likelihood solution is obtained by minimizing the error given by (see (Szeliski and Coughlan, 1994))

$$E_{ssd}\{(u_i, v_i)\} = \sum_i (I_2(x_i + u_i, y_i + v_i) - I_1(x_i, y_i))^2. \tag{2}$$

This formula is popularly known as the *sum of squared differences* (SSD) formula. In this motion model, the key underlying assumption is that intensity at corresponding pixels in I_1 and I_2 is unchanged and that I_1 and I_2 differ by local displacements. Other error criteria that take into account global variations in brightness and contrast between the two images and that are nonquadratic can be designed as (Szeliski and Coughlan, 1994; Black and Anandan, 1993),

$$E\prime_{ssd}\{(u_i, v_i)\} = \sum_i [I_2(x_i + u_i, y_i + v_i) - cI_1(x_i, y_i) + b]^2 \tag{3}$$

where b and c are the intensity and uniform contrast correction terms per-frame, which need to be recovered concurrently with the flow field. Expanding I_2 in a Taylor series in (u_i, v_i) to the first order yields the famous *image brightness constraint* of Horn and Schunk (Horn and Schunk, 1981). The above described objective function has been minimized in the past by several techniques, some of them using regularization on (u, v) (Horn and Schunk, 1981; Lai and Vemuri, 1995). In this paper, we tile a single image into several patches each of which can be described by either a local motion or a global parametric motion model. The tiling process is made recursive. The decision to tile a region further is made based on the error in computed motion/registration.

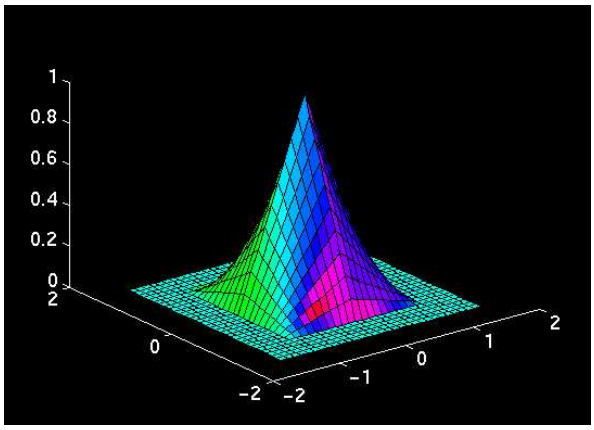


Figure 1: Bilinear basis function.

2.1 2D Local Flow

We represent the displacement fields $u(x, y)$ and $v(x, y)$ by B-splines with a small number of control points $\hat{u}_j$ and $\hat{v}_j$ as in Szeliski (Szeliski and Coughlan, 1994). Then the displacement at a pixel location i is given by

$$\begin{aligned} u(x_i, y_i) &= \sum_j \hat{u}_j w_{ij} = \sum_j \hat{u}_j B_j(x_i, y_i) \\ v(x_i, y_i) &= \sum_j \hat{v}_j w_{ij} = \sum_j \hat{v}_j B_j(x_i, y_i) \end{aligned} \tag{4}$$

where the $w_{ij} = B_j(x_i, y_i)$ are the *basis functions* with finite support. In our implementation, we have used bilinear basis, $B(x, y) = (1-|x|)(1-|y|)$ for (x, y) in $[-1, 1]^2$ as shown in figure 1, and we also assumed that the spline control grid is a subsampled version of the image pixel grid $(\hat{x}_j = mx_i, \hat{y}_j = my_i)$, as in figure 2.

This spline-based representation of the motion field possesses several advantages. Firstly, it imposes a built-in smoothness on the motion field and thus removing the need for further regularization. Secondly, it eliminates the need for correlation windows centered at each pixel which are computationally expensive. In this scheme, the flow field $(\hat{u}_j, \hat{v}_j)$ is estimated by a weighted sum from all the pixels beneath the support of its basis functions. In the correlation window-based scheme, each pixel contributes to m^2 overlapping windows, where m is the size of the window. However, in the spline-based scheme, each pixel contributes its error only to

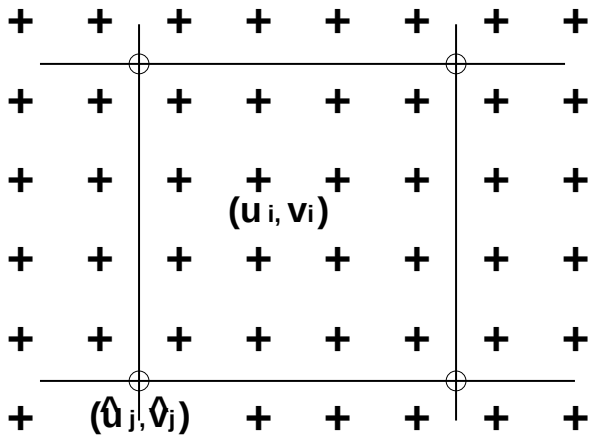


Figure 2: Spline-based flow field representation: the spline control points $\{(\hat{u}_j, \hat{v}_j)\}$ are indicated as circles and the pixel grid $\{(u_i, v_i)\}$ are shown as crosses.

its four neighboring control vertices which influence its displacement. Therefore, the latter achieves computation savings of $O(m^2)$ over the correlation window-based approaches.

In this paper, we use a slightly different error measurement from the one described above. Given two gray level images $I_1(x, y)$ and $I_2(x, y)$, where I_1 is the model, I_2 is the target, to compute an estimate $\hat{\mathbf{T}} = (\hat{u_1}, \hat{v_1}, ..., \hat{u_n}, \hat{v_n})^T$ of the true flow field $T = (u_1, v_1, ..., u_n, v_n)^T$ at n control points, first an intermediate image I_m is introduced and the motion is modeled in terms of the B-Spline control points as

$$I_m(\mathbf{X_i}, \hat{\mathbf{T}}) = I_1(x_i + \sum_j \hat{u}_j w_{ij}, y_i + \sum_j \hat{v}_j w_{ij}). \tag{5}$$

where $\mathbf{X}_i = (x_i, y_i)$ and w_{ij} are the basis functions as before. The expectation E of the Squared Difference, E_{sd}, is chosen to be the error criterion $J\{E_{sd}(\hat{\mathbf{T}})\}$

$$J\{E_{sd}(\hat{\mathbf{T}})\} = E\{(I_m(\mathbf{X_i}, \hat{\mathbf{T}}) - I_2(\mathbf{X_i}))^2\}. \tag{6}$$

Note that if we consider I_1 and I_2 as random fields, the above spline-based error criterion actually is the difference of the autocorrelation function of I_2 and the cross-correlation function of I_2 and I_m

$$J\{E_{sd}(\hat{\mathbf{T}})\} \quad = \quad E\{(I_m(\mathbf{X_i}, \hat{\mathbf{T}}) - I_2(\mathbf{X_i}))^2\}$$

$$
\begin{aligned}
&= E\{I_2^2 - 2I_m I_2 + I_m^2\} \\
&= 2(E\{I_2(\mathbf{X_i})^2\} - E\{I_2(\mathbf{X_i}) I_m(\mathbf{X_i}, \hat{\mathbf{T}})\}). \qquad (7)
\end{aligned}
$$

In the following, we describe the 2D/3D global flow motion models in the above described spline-based estimation framework. We will then discuss the numerical optimization of the error term to determine the motion parameters in each of the motion models discussed.

2.2 2D Rigid Flow

In many applications, the whole image undergoes the same type of motion, in which case, it is possible to parameterize the flow field by a small set of parameters describing the global motion, e.g., rigid, affine, quadratic and other types of transformations. The planar rigid flow model is defined as

$$
\begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix} = \begin{bmatrix} cos\phi & -sin\phi \\ sin\phi & cos\phi \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} - \begin{bmatrix} x \\ y \end{bmatrix} \qquad (8)
$$

where, the *global parameters* $\mathbf{T} = (\phi, d_1, d_2)^T$ are called *global motion parameters*, denoting the rotation angle and the translations along x and y direction respectively. Let $\mathbf{T'} = (cos\phi, -sin\phi, -d_1, sin\phi, cos\phi, -d_2)^T$. To compute an estimate of the global motion, we first define the displacements, $\hat{\mathbf{u}}_\mathbf{j} = (\hat{u}_j, \hat{v}_j)^T$, at the spline control vertices (indexed by the subscript j) in terms of the global motion parameters:

$$
\begin{aligned}
\hat{\mathbf{u}}_\mathbf{j} &= \begin{bmatrix} \hat{x_j} & \hat{y_j} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \hat{x_j} & \hat{y_j} & 1 \end{bmatrix} \mathbf{T'} - \begin{bmatrix} \hat{x_j} \\ \hat{y_j} \end{bmatrix} \\
&= \mathbf{T_j T'} - \hat{\mathbf{X}}_\mathbf{j} \qquad (9)
\end{aligned}
$$

where $\hat{\mathbf{X}}_\mathbf{j} = (\hat{x}_j, \hat{y}_j)^T$ is the coordinate of the control point j. We then define the flow at each pixel by interpolation using our spline representation. The error criterion $J\{E_{sd}(\hat{\mathbf{T}})\}$ for the rigid motion case becomes,

$$
J\{E_{sd}(\hat{\mathbf{T}})\} = E\{(I_1(\mathbf{X}_i + \sum_j w_{ij}(\mathbf{T_j T'} - \hat{\mathbf{X}}_\mathbf{j})) - I_2(\mathbf{X}_i))^2\}. \qquad (10)
$$

2.3 2D Affine Flow

A more general 2D motion is the affine transformation, which includes rotation, translation and scaling. The planar affine flow model is defined in the following manner

$$\begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix} = \begin{bmatrix} t_0 & t_1 \\ t_3 & t_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_2 \\ t_5 \end{bmatrix} - \begin{bmatrix} x \\ y \end{bmatrix} \tag{11}$$

where, $\mathbf{T} = (t_0, ..., t_5)^T$ forms the *global parameters*. Once again, to compute an estimate of the global motion, we first define the displacement at the spline control vertex j, $\hat{\mathbf{u}}_\mathbf{j} = (\hat{u}_j, \hat{v}_j)^T$, in terms of the global motion parameters

$$\begin{aligned} \hat{\mathbf{u}}_\mathbf{j} &= \begin{bmatrix} \hat{x_j} & \hat{y_j} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \hat{x_j} & \hat{y_j} & 1 \end{bmatrix} \mathbf{T} - \begin{bmatrix} \hat{x_j} \\ \hat{y_j} \end{bmatrix} \\ &= \mathbf{T_j T} - \hat{\mathbf{X}}_\mathbf{j} \end{aligned} \tag{12}$$

where $\hat{\mathbf{X}}_\mathbf{j} = (\hat{x}_j, \hat{y}_j)^T$ is the coordinate of the control point j. Substituting into equation (6), the error criterion $J\{E_{sd}(\hat{\mathbf{T}})\}$ for the affine motion case becomes

$$J\{E_{sd}(\hat{\mathbf{T}})\} = E\{(I_1(\mathbf{X}_i + \sum_j w_{ij}(\mathbf{T_j T} - \hat{\mathbf{X}}_\mathbf{j})) - I_2(\mathbf{X}_i))^2\}. \tag{13}$$

2.4 3D Affine Flow

Extending from 2D affine flow to 3D affine flow is quite straightforward. There are twelve motion parameters collected into a vector $\mathbf{T} = (t_0, ..., t_{11})^T$, and the control points in terms of the motion parameters are defined as

$$\hat{\mathbf{u}}_\mathbf{j} = \begin{bmatrix} \hat{x_j} & \hat{y_j} & \hat{z_j} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \hat{x_j} & \hat{y_j} & \hat{z_j} & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \hat{x_j} & \hat{y_j} & \hat{z_j} & 1 \end{bmatrix} \mathbf{T} - \begin{bmatrix} \hat{x_j} \\ \hat{y_j} \\ \hat{z_j} \end{bmatrix}, \tag{14}$$

which can be written as

$$\hat{\mathbf{u}}_\mathbf{j} = \mathbf{T_j T} - \hat{\mathbf{X}}_\mathbf{j}. \tag{15}$$

3 Numerical Solution

We now describe a novel adaptation of an elegant numerical method by Burkardt and Diehl (Burkhardt and Diehl, 1986) which is a modification of the standard Newton method for solving a system of nonlinear equations. The modification involves precomputation of the Hessian matrix at the optimum without starting the iterative minimization process. *Our adaptation of this idea to the framework of optical flow computation with spline-based flow field representations is new and involves nontrivial derivations of the gradient vectors and Hessian matrices.*

3.1 Modified Newton Method

In this section, we present the modified Newton method to minimize the error term $J\{E_{sd}\}$ given earlier. In the following, we will essentially adopt the notation from (Burkhardt and Diehl, 1986) to derive the modified Newton iteration and develop new notation as necessary. The primary structure of the algorithm is given by the following iteration formula

$$\hat{\mathbf{T}}^{k+1} = \hat{\mathbf{T}}^{k} - \mathbf{H}^{-1}(\hat{\mathbf{T}} = \hat{\mathbf{T}}^{*} = \mathbf{T})\mathbf{g}(\hat{\mathbf{T}}^{k}), \tag{16}$$

where $\mathbf{H}$ is the Hessian matrix and $\mathbf{g}$ is the gradient vector. Unlike in a typical Newton iteration, in the above equation, observe that the *Hessian* is always computed at the optimum $\hat{\mathbf{T}} = \hat{\mathbf{T}}^{*} = \mathbf{T}$ instead of the iteration point $\hat{\mathbf{T}}^{k}$. So, one of the key problems is how to compute the Hessian at the optimum prior to beginning the iteration, i.e., without actually knowing the optimum.

3.1.1 Computing the Hessian Matrix at the Optimum

Let the vector $\mathbf{X}$ denote the coordinates in any image and $h : \mathbf{X} \rightarrow \mathbf{X}'$ a transformation from $\mathbf{X}$ to another set of coordinates $\mathbf{X}'$, characterized by a set of parameters collected into a vector $\mathbf{T}$, i.e., $\mathbf{X}' = h(\mathbf{X}, \mathbf{T})$. The parameter vector $\mathbf{T}$ can represent any of rigid, affine, shearing, projective etc. transformations.

Normally, the Hessian at any iteration point $\hat{\mathbf{T}}^k$ is

$$\mathbf{H}(\hat{\mathbf{T}}^k) = \left.\frac{\partial^2 J\{E_{SD}(\hat{\mathbf{T}})\}}{\partial\hat{\mathbf{T}}\partial\hat{\mathbf{T}}}\right|_{\hat{\mathbf{T}}=\hat{\mathbf{T}}^k}$$

$$= 2E\left\{\frac{\partial I_m(\mathbf{X},\hat{\mathbf{T}})}{\partial\hat{\mathbf{T}}}\left(\frac{\partial I_m(\mathbf{X},\hat{\mathbf{T}})}{\partial\hat{\mathbf{T}}}\right)^T + (I_m(\mathbf{X},\hat{\mathbf{T}}) - I_2(\mathbf{X}))\frac{\partial^2 I_m(\mathbf{X},\hat{\mathbf{T}})}{\partial\hat{\mathbf{T}}\partial\hat{\mathbf{T}}}\right\}\Bigg|_{\hat{\mathbf{T}}=\hat{\mathbf{T}}^k}. \quad (17)$$

Note that at the optimum $I_m(\mathbf{X},\hat{\mathbf{T}}) = I_m(\mathbf{X},\mathbf{T}^*) = I_2(\mathbf{X})$, thus zeroing out the second term in the above equation yields the Hessian matrix at the optimum

$$\mathbf{H}(\mathbf{T}^*) = \frac{\partial^2 J\{E_{SD}(\mathbf{T}^*)\}}{\partial\hat{\mathbf{T}}\partial\hat{\mathbf{T}}} = 2E\left\{\frac{\partial I_m(\mathbf{X},\mathbf{T}^*)}{\partial\hat{\mathbf{T}}}\left(\frac{\partial I_m(\mathbf{X},\mathbf{T}^*)}{\partial\hat{\mathbf{T}}}\right)^T\right\}. \quad (18)$$

For the pure translation case, $\mathbf{T} = (d_1, d_2)^T$, $\partial I_m(\mathbf{X},\mathbf{T}^*)/\partial\hat{\mathbf{T}} = \partial I_2(\mathbf{X})/\partial\mathbf{X}$, then the Hessian at the optimum is independent of the optimum motion vector $(d_1^*, d_2^*)^T$

$$\mathbf{H}(\mathbf{T}^*) = \frac{\partial^2 J\{E_{SD}(\mathbf{T}^*)\}}{\partial\hat{\mathbf{T}}\partial\hat{\mathbf{T}}} = 2E\left\{\frac{\partial I_2(\mathbf{X})}{\partial\mathbf{X}}\left(\frac{\partial I_2(\mathbf{X})}{\partial\mathbf{X}}\right)^T\right\}. \quad (19)$$

But for other complicated types of motion, the Hessian at the optimum will explicitly depend on the optimum motion vector and hence cannot be precomputed directly. For example, for 2D rigid motion, $\mathbf{T} = (\phi, d_1, d_2)^T$, and we have

$$\begin{aligned}
\frac{\partial I_m(\mathbf{X},\mathbf{T}^*)}{\partial\hat{\phi}} &= -\frac{\partial I_2(\mathbf{X})}{\partial x}y + \frac{\partial I_2(\mathbf{X})}{\partial y}x \\
\frac{\partial I_m(\mathbf{X},\mathbf{T}^*)}{\partial\hat{d}_1} &= -\frac{\partial I_2(\mathbf{X})}{\partial x}cos\phi^* + \frac{\partial I_2(\mathbf{X})}{\partial y}sin\phi^* \\
\frac{\partial I_m(\mathbf{X},\mathbf{T}^*)}{\partial\hat{d}_2} &= -\frac{\partial I_2(\mathbf{X})}{\partial x}sin\phi^* - \frac{\partial I_2(\mathbf{X})}{\partial y}cos\phi^*.
\end{aligned} \quad (20)$$

Thus the Hessian at the optimum will depend on the optimum rotation angle ϕ^*, for example, $\mathbf{H}_{12}(\mathbf{T}^*) = \partial^2 I_m(\mathbf{X},\mathbf{T}^*)/\partial\hat{\phi}\partial\hat{d}_1$. However, for a slightly different formulation, the Hessian for general motion can be precomputed. In the following, we will describe such a formulation.

A clever technique was introduced in (Diehl and Burkhardt, 1989), involving the use of innovations (incremental transformations) with a moving coordinate system $\{\mathbf{X^k}\}$ and an intermediate motion vector $\tilde{\mathbf{T}}$ to develop the formulas for precomputing the Hessian. This

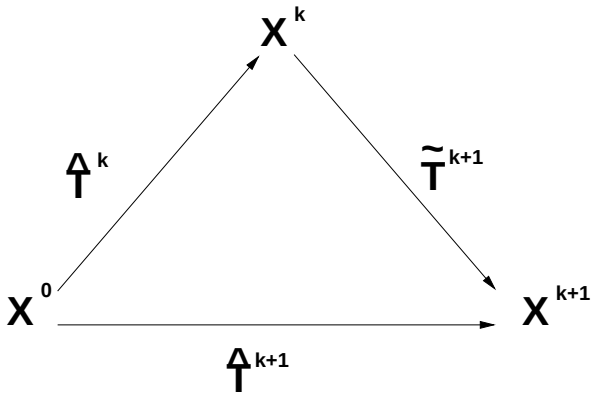


Figure 3: The relationship between $\hat{\mathbf{T}}^{\mathbf{k}}$, $\hat{\mathbf{T}}^{\mathbf{k+1}}$ and $\tilde{\mathbf{T}}^{\mathbf{k+1}}$

intermediate motion vector gives the motion from $\{\mathbf{X}^{\mathbf{k}}\}$ of iteration step k to $\{\mathbf{X}^{\mathbf{k+1}}\}$ of iteration step $k+1$, as shown in figure 3.

$$\mathbf{X}^k = h(\mathbf{X}^0, \hat{\mathbf{T}}^k) \tag{21}$$

$$\mathbf{X}^{k+1} = h(\mathbf{X}^k, \tilde{\mathbf{T}}^{k+1}) = h(h(\mathbf{X}^0, \hat{\mathbf{T}}^k), \tilde{\mathbf{T}}^{k+1}) = h(\mathbf{X}^0, \hat{\mathbf{T}}^{k+1}) \tag{22}$$

$$I_m(\mathbf{X}^0, \hat{\mathbf{T}}^{k+1}) = I_m(\mathbf{X}^0, \hat{\mathbf{T}}^k \circ \tilde{\mathbf{T}}^{k+1}) \tag{23}$$

Instead of differentiating the error criterion with respect to $\hat{\mathbf{T}}$, the error criterion now has to be differentiated with respect to $\tilde{\mathbf{T}}$. The innovation $\tilde{\mathbf{T}}^{k+1}$ is defined as

$$\tilde{\mathbf{T}}^{k+1} = -\tilde{\mathbf{H}}^{-1}\tilde{\mathbf{g}}(\hat{\mathbf{T}}^k), \tag{24}$$

where, $\tilde{\mathbf{H}}$ is the Hessian matrix at the optimum, $\tilde{\mathbf{g}}$ is the gradient vector at the iteration point $\hat{\mathbf{T}}^{\mathbf{k}}$. The estimate at step k+1 is then updated by

$$\hat{\mathbf{T}}^{k+1} = \tilde{\mathbf{T}}^{k+1} \circ \hat{\mathbf{T}}^k = f(\hat{\mathbf{T}}^k, \tilde{\mathbf{T}}^{k+1}). \tag{25}$$

Where f is a function that depends on the type of motion model used.

In the following, we will show how to compute the Hessian matrix at the optimum with respect to $\tilde{\mathbf{T}}$. Assuming that the optimum is reached at step N, i.e., $I_m(\mathbf{X}, \hat{\mathbf{T}}^N) = I_2(\mathbf{X})$.

Then the estimate $\tilde{\mathbf{T}}^{N+1}$ of iteration step $N+1$ will be zero, $\tilde{\mathbf{T}}^{N+1} = \mathbf{0}$. This leads to the following equations (see appendix):

$$\left.\frac{\partial I_m(\mathbf{X}, \mathbf{T}^N \circ \tilde{\mathbf{T}}^{N+1})}{\partial \tilde{\phi}^{N+1}}\right|_{\tilde{T}^{N+1}=0} = -\frac{\partial I_1(\mathbf{X}^N)}{\partial x^N} y^N + \frac{\partial I_1(\mathbf{X}^N)}{\partial y^N} x^N$$

$$\left.\frac{\partial I_m(\mathbf{X}, \mathbf{T}^N \circ \tilde{T}^{N+1})}{\partial \tilde{d}_1}^{N+1}\right|_{\tilde{T}^{N+1}=0} = -\frac{\partial I_1(\mathbf{X}^N)}{\partial x^N}. \tag{26}$$

Where, differentiation with respect to X^N involves first applying the motion $\hat{\mathbf{T}}^N$ to the initial image $I_1(\mathbf{X}^0)$, to get the new image $I_1(\mathbf{X}^N)$. Then, we apply a small motion $\tilde{\mathbf{T}}^{N+1}$ to this image and then differentiate with respect to $\mathbf{X}^N$.

Using these equations to compute the Hessian at the optimum, the entry $\tilde{\mathbf{H}}_{12}(\mathbf{T}^*)$ is given by

$$\begin{aligned}\tilde{\mathbf{H}}_{12}(\mathbf{T}^*) = \frac{\partial^2 J\{E_{SD}(\mathbf{T}^*)\}}{\partial \tilde{d}_1^{N+1} \partial \tilde{\phi}^{N+1}} &= 2E\left\{\frac{\partial I_1(\mathbf{X^N})}{\partial x^N}\left(\frac{\partial I_1(\mathbf{X^N})}{\partial x^N} y^N - \frac{\partial I_1(\mathbf{X^N})}{\partial y^N} x^N\right)\right\} \\ &= 2E\left\{\frac{\partial I_1(\mathbf{X})}{\partial x}\left(\frac{\partial I_1(\mathbf{X})}{\partial x} y - \frac{\partial I_1(\mathbf{X})}{\partial y} x\right)\right\}. \end{aligned} \tag{27}$$

This equation was arrived by assuming that expectation value is independent of the coordinate system and therefore all the derivatives are given in terms of the original image I_1. This shows that the Hessian matrix at the optimum is only related to I_1 and can be computed prior to starting the iterations.

Extending from 2D rigid motion to general motion is quite straightforward. Starting from the term $\partial I_m(\mathbf{X}, \mathbf{T}^*)/\partial \tilde{\mathbf{T}}$ in equation (18), let $\mathbf{X}^* = \mathbf{T}^*(\mathbf{X})$ be the coordinate system at the optimum then, $I_m(\mathbf{X}, \mathbf{T}^*) = I_1(\mathbf{X}^*)$. Then, make use of the same idea as in the rigid motion example, i.e., apply a small motion $\tilde{\mathbf{T}}$ to $I_1(\mathbf{X}^*)$, differentiate with respect to $\tilde{\mathbf{T}}$ and take the value at $\tilde{\mathbf{T}} = 0$ to get the equation (28) (see appendix for details).

$$\left.\left(\frac{\partial I_1(\mathbf{X}^*)}{\partial \tilde{\mathbf{T}}}\right)^T\right|_{\tilde{\mathbf{T}}=0} = \left.\left(\frac{\partial I_1(\mathbf{X}^*)}{\partial \mathbf{X}^*}\right)^T \frac{\partial \mathbf{X}^*}{\partial \tilde{\mathbf{T}}}\right|_{\tilde{\mathbf{T}}=0}. \tag{28}$$

Substituting into equation (18), yields (see appendix for details)

$$\tilde{\mathbf{H}} = 2E\left\{\frac{\partial I_1(\mathbf{X}^*)}{\partial \tilde{\mathbf{T}}}\left(\frac{\partial I_1(\mathbf{X}^*)}{\partial \tilde{\mathbf{T}}}\right)^T\right\}\Bigg|_{\tilde{\mathbf{T}}=\mathbf{0}}$$

$$= \quad 2E\left\{\left(\frac{\partial \mathbf{X}^*}{\partial \tilde{\mathbf{T}}}\right)^T \frac{\partial I_1(\mathbf{X}^*)}{\partial \mathbf{X}^*}\left(\frac{\partial I_1(\mathbf{X}^*)}{\partial \mathbf{X}^*}\right)^T \frac{\partial \mathbf{X}^*}{\partial \tilde{T}}\right\}\Bigg|_{\tilde{\mathbf{T}}=\mathbf{0}}. \tag{29}$$

Finally, the Hessian at the optimum is given by (see appendix for details)

$$\tilde{\mathbf{H}} = 2E\left\{\left(\frac{\partial h(\mathbf{X},\mathbf{T})}{\partial \mathbf{T}}\right)^{\mathbf{T}} \frac{\partial I_1(\mathbf{X})}{\partial \mathbf{X}}\left(\frac{\partial I_1(\mathbf{X})}{\partial \mathbf{X}}\right)^{\mathbf{T}} \frac{\partial h(\mathbf{X},\mathbf{T})}{\partial \mathbf{T}}\right\}\Bigg|_{\mathbf{T}=\mathbf{0}}. \tag{30}$$

3.1.2 Gradient Vector Computation

After the calculation of the Hessian matrix at the optimum, we still need to provide the gradient vector $\tilde{\mathbf{g}}(\hat{\mathbf{T}}^k)$. *Since the Hessian matrix can be precomputed, the numerical cost per iteration step of the Modified Newton scheme is determined by the calculation of the gradient vector.* A significant reduction of the calculation can be reached if at each step of the iteration, instead of expressing the gradient vector, the partial derivative of the transformed model image $I_m(\mathbf{X}, \hat{\mathbf{T}}^k, \tilde{\mathbf{T}}^{k+1})$ with respect to the motion vector $\tilde{\mathbf{T}}^{k+1}$, we now express it in terms of the corresponding derivatives of the image $I_2(\mathbf{X})$ with respect to coordinates $\{\mathbf{X}\}$. Therefore, the partial derivatives can be determined in advance and within each iteration they only need to be multiplied with the difference of the image $I_m(\mathbf{X}, \hat{\mathbf{T}}^k)$ and $I_2(\mathbf{X})$.

The gradient vector in terms of first derivatives of the error function with respect to the motion vector $\tilde{\mathbf{T}}$ is

$$\tilde{\mathbf{g}}^(\hat{\mathbf{T}}^k) = 2E\left\{(I_m(\mathbf{X}, \hat{\mathbf{T}}^k) - I_2(\mathbf{X}))\frac{\partial I_m(\mathbf{X}, \hat{\mathbf{T}}_k \circ \tilde{\mathbf{T}}^{k+1})}{\partial \tilde{\mathbf{T}}^{k+1}}\right\}\Bigg|_{\tilde{\mathbf{T}}^{k+1}=0} \tag{31}$$

where,

$$\left(\frac{\partial I_m(\mathbf{X}, \hat{\mathbf{T}}_k \circ \tilde{\mathbf{T}}^{k+1})}{\partial \tilde{\mathbf{T}}^{k+1}}\right)^T = \left(\frac{\partial I_m(\mathbf{X}, \hat{\mathbf{T}}_k \circ \tilde{\mathbf{T}}^{k+1})}{\partial \mathbf{X}}\right)^T \left(\frac{\partial \mathbf{X}^k}{\partial \mathbf{X}}\right)^{-1} \left(\frac{\partial \mathbf{X}^{k+1}}{\partial \mathbf{X}^k}\right)^{-1} \frac{\partial \mathbf{X}^{k+1}}{\partial \tilde{\mathbf{T}}^{k+1}}. \tag{32}$$

At the optimum, $\tilde{\mathbf{T}}^{\mathbf{K+1}} = \mathbf{0}$, $I_m(\mathbf{X}, \hat{\mathbf{T}}_k) = I_2(\mathbf{X})$, $\mathbf{X^{k+1}} = \mathbf{X^k}$, thus $\partial \mathbf{X^{k+1}}/\partial \mathbf{X^k} = \mathbf{I}$. Hence

$$\left(\frac{\partial I_m(\mathbf{X}, \hat{\mathbf{T}}_k \circ \tilde{\mathbf{T}}^{k+1})}{\partial \tilde{\mathbf{T}}^{k+1}}\right)^T\Bigg|_{\tilde{\mathbf{T}}^{k+1}=0} = \left(\frac{\partial I_2(\mathbf{X})}{\partial \mathbf{X}}\right)^T \left(\frac{\partial \mathbf{X}^k}{\partial \mathbf{X}}\right)^{-1} \left(\frac{\partial \mathbf{X}_{k+1}}{\partial \tilde{\mathbf{T}}^{k+1}}\right)\Bigg|_{\tilde{\mathbf{T}}^{k+1}=0}. \tag{33}$$

Finally, substituting into equation (31), the gradient vector with respect to $\tilde{\mathbf{T}}$ is given by

$$\tilde{\mathbf{g}}(\hat{\mathbf{T}}^k) = 2E\left\{e\left(\left(\frac{\partial I_2(\mathbf{X})}{\partial \mathbf{X}}\right)^{\mathbf{T}} \left(\frac{\partial \mathbf{X}^k}{\partial \mathbf{X}}\right)^{-1} \left(\frac{\partial \mathbf{X}^{k+1}}{\partial \tilde{\mathbf{T}}^{k+1}}\right)\Bigg|_{\tilde{\mathbf{T}}^{k+1}=0}\right)^{\mathbf{T}}\right\}, \tag{34}$$

where, $e = I_m(\mathbf{X}, \hat{\mathbf{T}}^k) - I_2(\mathbf{X})$.

3.1.3 Summary of the Modified Newton Method

Thus the Modified Newton algorithm consists of the following steps:

1. Precompute the Hessian at the optimum $\tilde{\mathbf{H}}$ using equation (30).
2. At iteration k, compute the gradient vector using equation (34).
3. Compute the innovation $\tilde{\mathbf{T}}^{k+1}$ using equation (24).
4. Update the motion parameter $\hat{\mathbf{T}}^{k+1}$ using equation (25).

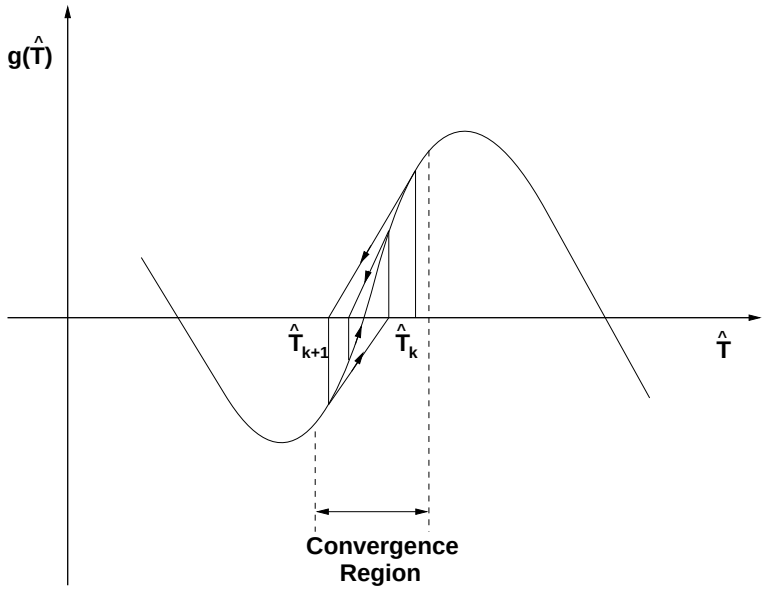
The Modified Newton method has several advantages over the standard Newton iteration scheme for solving nonlinear equations. The major advantage is a computational one, which stems from the fact that the Hessian is precomputed . In addition, the Modified Newton method has a broader range of convergence than the standard Newton scheme, as was shown in (Burkhardt and Diehl, 1986). This allows the algorithm to cope with large motions even when the initial guess is far from the optimum. An example depicting the convergence range for the conventional Newton and Modified Newton scheme is shown in figure 4. Finally, the Modified Newton schemes is at least quadratically convergent (Burkhardt and Diehl, 1986).

3.2 Application of the Modified Newton Method to the Local/Global Motion Model

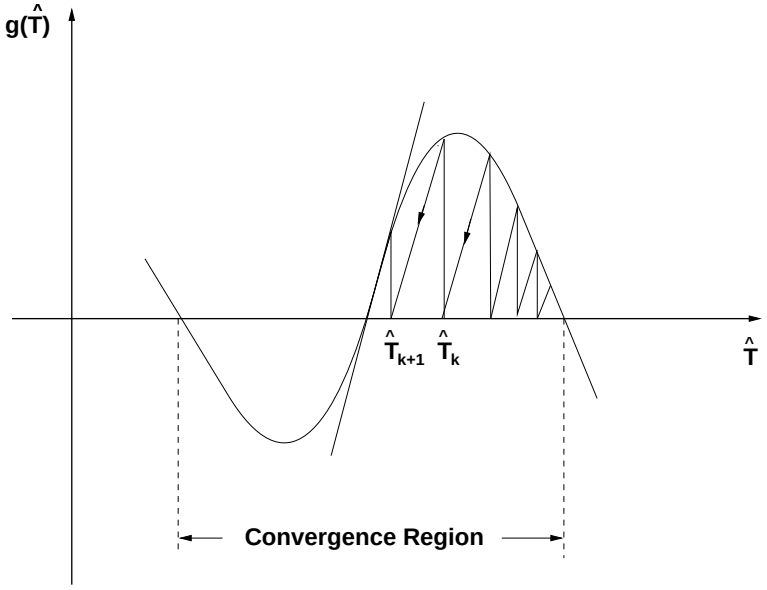
In this section, we describe a robust way of pre-computing the Hessian matrix at the optimum and the gradient vector at each iteration step for motion models including local and global ones.

3.2.1 H & g for 2D Local Flow

Let $\hat{\mathbf{X}}_j, (j = 1, 2, ..., n)$ be a vector of control points. Then the flow vector is given by $\mathbf{T} = (\hat{u}_1, \hat{v}_1, ..., \hat{u}_n, \hat{v}_n)^T$. Actually, local flow is equivalent to pure translation at each pixel



(a)



(b)

Figure 4: Convergence range comparison between standard Newton algorithm and Modified Newton algorithm. (a) Standard Newton algorithm (b) Modified Newton algorithm

and hence, the Hessian at the optimum is only related to the derivatives with respect to the original coordinates and does not depend on the motion vector. Therefore, it can be calculated without introducing $\tilde{\mathbf{T}}$ as shown below

$$Let\ \partial = (w_1\frac{\partial}{\partial x}, w_1\frac{\partial}{\partial y}, ...w_n\frac{\partial}{\partial x}, w_n\frac{\partial}{\partial y}). \tag{35}$$

The Hessian at the optimum is then given by

$$\mathbf{H}^*_{ij} = \tilde{\mathbf{H}}_{ij} = 2E\{\partial_i I_1(\mathbf{X})\partial_j I_1(\mathbf{X})\}. \tag{36}$$

Whereas the gradient vector is,

$$\tilde{\mathbf{g}}(\hat{\mathbf{T}}^k) = 2E\left\{(I_m - I_2)\mathbf{N}^T\mathbf{M}^{-T}\frac{\partial I_2(\mathbf{X})}{\partial \mathbf{X}}\right\}, \tag{37}$$

where, the matrices $\mathbf{M}$ and $\mathbf{N}$ are given by

$$\mathbf{M} = \frac{\partial \mathbf{X^k}}{\partial \mathbf{X}} = \begin{bmatrix} 1 + \sum_j (w_j)_x{}'\hat{u}_j^k & \sum_j (w_j)_y{}'\hat{u}_j^k \\ \sum_j (w_j)_x{}'\hat{v}_j^k & 1 + \sum_j (w_j)_y{}'\hat{v}_j^k \end{bmatrix} \tag{38}$$

and

$$\mathbf{N} = \left(\frac{\partial \mathbf{X^{k+1}}}{\partial \tilde{\mathbf{T}}^{k+1}}\right)\Bigg|_{\tilde{\mathbf{T}}^{k+1}=0} = \begin{bmatrix} w_1 & 0 & ... & w_n & 0 \\ 0 & w_1 & ... & 0 & w_n \end{bmatrix}, \tag{39}$$

respectively. We can now substitute equations (36) and (37) into equation (24) yielding $\tilde{\mathbf{T}}^{k+1}$ which upon substitution into equation (25) results in $\hat{\mathbf{T}}^{k+1}$. Hence, the numerical iterative formula (25) used in computing the local motion becomes

$$\hat{\mathbf{T}}^{k+1} = \hat{\mathbf{T}}^k - \tilde{\mathbf{H}}^{-1}\tilde{\mathbf{g}}(\hat{\mathbf{T}}^k). \tag{40}$$

The size of $\tilde{\mathbf{H}}$ is determined by how many control points are used in representing the flow field (u, v). For 3D problems, $\tilde{\mathbf{H}}$ is $(3n \times 3n)$ where n is the number of control points. For large n, numerical iterative solvers are quite attractive and we use a preconditioned conjugate gradient (PCG) algorithm (Lai and Vemuri, 1995; Szeliski and Coughlan, 1994) to solve the linear system $\tilde{\mathbf{H}}^{-1}\tilde{\mathbf{g}}(\hat{\mathbf{T}}^k)$. The specific preconditioning we use in our implementation of the local flow is a simple diagonal Hessian preconditioning. More sophisticated preconditioners can be used in place of this simple preconditioner and we refer the reader to (Lai and Vemuri, 1995) for more preconditioning.

3.2.2 H & g for 2D Rigid Flow

We now derive the Hessian matrix and the gradient vector for the case when the flow field is expressed using a global parametric form specifically, rigid motion parameterization.

Let $\hat{\mathbf{X}}_j, (j = 1, 2, ..., n)$ be the control points and let

$$\mathbf{A} = \left(\frac{\partial h(\mathbf{X}, \mathbf{T})}{\partial \mathbf{T}}\right)^T = \begin{bmatrix} -\sum_j w_j \hat{y}_j & \sum_j w_j \hat{x}_j \\ -\sum_j w_j & 0 \\ 0 & -\sum_j w_j \end{bmatrix}. \tag{41}$$

The Hessian at the optimum can then be written as

$$\tilde{\mathbf{H}} = 2E\left\{\mathbf{A}\frac{\partial I_1(\mathbf{X})}{\partial \mathbf{X}}\left(\frac{\partial I_1(\mathbf{X})}{\partial \mathbf{X}}\right)^T \mathbf{A}^T\right\}. \tag{42}$$

The gradient vector $\tilde{\mathbf{g}}(\hat{\mathbf{T}}^k)$ at the optimum is given by

$$\tilde{\mathbf{g}}(\hat{\mathbf{T}}^k) = 2E\left\{(I_m - I_2)\mathbf{N}\mathbf{M}^{-T}\frac{\partial I_2(\mathbf{X})}{\partial \mathbf{X}}\right\}, \tag{43}$$

where the matrices $\mathbf{M}$ and $\mathbf{N}$ are

$$\mathbf{M} = \frac{\partial \mathbf{X^k}}{\partial \mathbf{X}} = \begin{bmatrix} 1 + \sum_j (w_j)_x{}' \hat{u}_j^k & \sum_j (w_j)_y{}' \hat{u}_j^k \\ \sum_j (w_j)_x{}' \hat{v}_j^k & 1 + \sum_j (w_j)_y{}' \hat{v}_j^k \end{bmatrix} \tag{44}$$

and

$$\mathbf{N} = \left(\frac{\partial \mathbf{X^{k+1}}}{\partial \tilde{\mathbf{T}}^{k+1}}\right)^T \Bigg|_{\tilde{\mathbf{T}}^{k+1}=0} = \begin{bmatrix} -\sum_j w_j \hat{y}_j^k & \sum_j w_j \hat{x}_j^k \\ -\sum_j w_j & 0 \\ 0 & -\sum_j w_j \end{bmatrix}, \tag{45}$$

respectively. The basic steps in our algorithm for computing the global rigid flow are,

1. Precompute the Hessian at the optimum $\tilde{\mathbf{H}}$ using equation (42).
2. At iteration k, compute the gradient vector using equation (43).
3. Compute the innovation $\tilde{\mathbf{T}}^{k+1}$ using equation (24).
4. Update the motion parameter $\hat{\mathbf{T}}^{k+1}$ using the following equation

$$\begin{bmatrix} \hat{\phi} \\ \hat{d}_1 \\ \hat{d}_2 \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\tilde{\phi} & -sin\tilde{\phi} \\ 0 & sin\tilde{\phi} & cos\tilde{\phi} \end{bmatrix}_{k+1} \begin{bmatrix} \hat{\phi} \\ \hat{d}_1 \\ \hat{d}_2 \end{bmatrix}_k + \begin{bmatrix} \tilde{\phi} \\ \tilde{d}_1 \\ \tilde{d}_2 \end{bmatrix}_{k+1}. \tag{46}$$

3.2.3 H & g for 2D Affine Flow

We now derive the Hessian matrix and the gradient vector for the case when the flow field is expressed using a global parametric form specifically, an affine parameterization.

Let $\hat{\mathbf{X}}_j, (j = 1, 2, ..., n)$ be the control points and let

$$\mathbf{A} = \left(\frac{\partial h(\mathbf{X}, \mathbf{T})}{\partial \mathbf{T}}\right)^T = \begin{bmatrix} \sum_j w_j \hat{x}_j & 0 \\ \sum_j w_j \hat{y}_j & 0 \\ \sum_j w_j & 0 \\ 0 & \sum_j w_j \hat{x}_j \\ 0 & \sum_j w_j \hat{y}_j \\ 0 & \sum_j w_j \end{bmatrix}. \tag{47}$$

The Hessian at the optimum can then be written as

$$\tilde{\mathbf{H}} = 2E\left\{\mathbf{A}\frac{\partial I_1(\mathbf{X})}{\partial \mathbf{X}}\left(\frac{\partial I_1(\mathbf{X})}{\partial \mathbf{X}}\right)^T \mathbf{A}^T\right\}. \tag{48}$$

The gradient vector $\tilde{\mathbf{g}}(\hat{\mathbf{T}}^k)$ at the optimum is given by

$$\tilde{\mathbf{g}}(\hat{\mathbf{T}}^k) = 2E\left\{(I_m - I_2)\mathbf{N}\mathbf{M}^{-T}\frac{\partial I_2(\mathbf{X})}{\partial \mathbf{X}}\right\}, \tag{49}$$

where the matrices $\mathbf{M}$ and $\mathbf{N}$ are:

$$\mathbf{M} = \frac{\partial \mathbf{X^k}}{\partial \mathbf{X}} = \begin{bmatrix} 1 + \sum_j (w_j)_x{}' \hat{u}_j^k & \sum_j (w_j)_y{}' \hat{u}_j^k \\ \sum_j (w_j)_x{}' \hat{v}_j^k & 1 + \sum_j (w_j)_y{}' \hat{v}_j^k \end{bmatrix} \tag{50}$$

and

$$\mathbf{N} = \left(\frac{\partial \mathbf{X^{k+1}}}{\partial \tilde{\mathbf{T}}^{k+1}}\right)^T \Bigg|_{\tilde{\mathbf{T}}^{k+1}=0} = \begin{bmatrix} \sum_j w_j \hat{x}_j^k & 0 \\ \sum_j w_j \hat{y}_j^k & 0 \\ \sum_j w_j & 0 \\ 0 & \sum_j w_j \hat{x}_j^k \\ 0 & \sum_j w_j \hat{y}_j^k \\ 0 & \sum_j w_j \end{bmatrix}, \tag{51}$$

respectively. The basic steps in our algorithm for computing the global 2D affine flow are,

1. Precompute the Hessian at the optimum $\tilde{\mathbf{H}}$ using equation (48).
2. At iteration k, compute the gradient vector using equation (49).
3. Compute the innovation $\tilde{\mathbf{T}}^{k+1}$ using equation (24).
4. Update the motion parameter $\hat{\mathbf{T}}^{k+1}$ using the following equations,

$$\begin{bmatrix} \hat{t}_0 & \hat{t}_1 \\ \hat{t}_3 & \hat{t}_4 \end{bmatrix}_{k+1} = \begin{bmatrix} \tilde{t}_0 + 1 & \tilde{t}_1 \\ \tilde{t}_3 & \tilde{t}_4 + 1 \end{bmatrix}_{k+1} \begin{bmatrix} \hat{t}_0 & \hat{t}_1 \\ \hat{t}_3 & \hat{t}_4 \end{bmatrix}_k \tag{52}$$

$$\begin{bmatrix} \hat{t}_2 \\ \hat{t}_5 \end{bmatrix}_{k+1} = \begin{bmatrix} \tilde{t}_0 + 1 & \tilde{t}_1 \\ \tilde{t}_3 & \tilde{t}_4 + 1 \end{bmatrix}_{k+1} \begin{bmatrix} \hat{t}_2 \\ \hat{t}_5 \end{bmatrix}_k + \begin{bmatrix} \tilde{t}_2 \\ \tilde{t}_5 \end{bmatrix}_{k+1}. \tag{53}$$

3.2.4 H & g for 3D Affine Flow

In this section, we derive the Hessian matrix and the gradient vector for the case when the flow field is in 3D and is expressed using a global parametric form specifically, a 3D affine parameterization. Let $\hat{\mathbf{X}}_j, (j = 1, 2, ..., n)$ be the control points and let

$$\mathbf{A} = \left(\frac{\partial h(\mathbf{X}, \mathbf{T})}{\partial \mathbf{T}}\right)^T = \begin{bmatrix} \sum_j w_j \hat{x}_j & 0 & 0 \\ \sum_j w_j \hat{y}_j & 0 & 0 \\ \sum_j w_j \hat{z}_j & 0 & 0 \\ \sum_j w_j & 0 & 0 \\ 0 & \sum_j w_j \hat{x}_j & 0 \\ 0 & \sum_j w_j \hat{y}_j & 0 \\ 0 & \sum_j w_j \hat{z}_j & 0 \\ 0 & \sum_j w_j & 0 \\ 0 & 0 & \sum_j w_j \hat{x}_j \\ 0 & 0 & \sum_j w_j \hat{y}_j \\ 0 & 0 & \sum_j w_j \hat{z}_j \\ 0 & 0 & \sum_j w_j \end{bmatrix}. \tag{54}$$

The Hessian at the optimum is then given by

$$\tilde{\mathbf{H}} = 2E\left\{\mathbf{A}\frac{\partial I_1(\mathbf{X})}{\partial \mathbf{X}}\left(\frac{\partial I_1(\mathbf{X})}{\partial \mathbf{X}}\right)^T \mathbf{A}^T\right\}. \tag{55}$$

and the gradient vector at the optimum is

$$\tilde{\mathbf{g}}(\hat{\mathbf{T}}^k) = 2E\left\{(I_m - I_2)\mathbf{N}\mathbf{M}^{-T}\frac{\partial I_2(\mathbf{X})}{\partial \mathbf{X}}\right\}, \tag{56}$$

where, the matrices $\mathbf{M}$ and $\mathbf{N}$ are as follows

$$\mathbf{M} = \frac{\partial \mathbf{X^k}}{\partial \mathbf{X}} = \begin{bmatrix} 1 + \sum_j (w_j)_x{}' \hat{u}_j^k & \sum_j (w_j)_y{}' \hat{u}_j^k & \sum_j (w_j)_z{}' \hat{u}_j^k \\ \sum_j (w_j)_x{}' \hat{v}_j^k & 1 + \sum_j (w_j)_y{}' \hat{v}_j^k & \sum_j (w_j)_z{}' \hat{v}_j^k \\ \sum_j (w_j)_x{}' \hat{q}_j^k & \sum_j (w_j)_y{}' \hat{q}_j^k & 1 + \sum_j (w_j)_z{}' \hat{q}_j^k \end{bmatrix} \tag{57}$$

and

$$\mathbf{N} = \left(\frac{\partial \mathbf{X^{k+1}}}{\partial \tilde{\mathbf{T}}^{k+1}} \right)^T \Bigg|_{\tilde{\mathbf{T}}^{k+1}=0} = \begin{bmatrix} \sum_j w_j \hat{x}_j^k & 0 & 0 \\ \sum_j w_j \hat{y}_j^k & 0 & 0 \\ \sum_j w_j \hat{z}_j^k & 0 & 0 \\ \sum_j w_j & 0 & 0 \\ 0 & \sum_j w_j \hat{x}_j^k & 0 \\ 0 & \sum_j w_j \hat{y}_j^k & 0 \\ 0 & \sum_j w_j \hat{z}_j^k & 0 \\ 0 & \sum_j w_j & 0 \\ 0 & 0 & \sum_j w_j \hat{x}_j^k \\ 0 & 0 & \sum_j w_j \hat{y}_j^k \\ 0 & 0 & \sum_j w_j \hat{z}_j^k \\ 0 & 0 & \sum_j w_j \end{bmatrix}. \tag{58}$$

Our algorithm for computing the global 3D affine flow consists of the following steps,

1. Precompute the Hessian at the optimum $\tilde{\mathbf{H}}$ using equation (55).

2. At iteration k, compute the gradient vector using equation (56).

3. Compute the transformation $\tilde{\mathbf{T}}^{k+1}$ using equation (24).

4. Update the motion parameter $\hat{\mathbf{T}}^{k+1}$ using the following update equations,

$$\begin{bmatrix} \hat{t}_0 & \hat{t}_1 & \hat{t}_2 \\ \hat{t}_4 & \hat{t}_5 & \hat{t}_6 \\ \hat{t}_8 & \hat{t}_9 & \hat{t}_{10} \end{bmatrix}_{k+1} = \begin{bmatrix} \tilde{t}_0 + 1 & \tilde{t}_1 & \tilde{t}_2 \\ \tilde{t}_4 & \tilde{t}_5 + 1 & \tilde{t}_6 \\ \tilde{t}_8 & \tilde{t}_9 & \tilde{t}_{10} + 1 \end{bmatrix}_{k+1} \begin{bmatrix} \hat{t}_0 & \hat{t}_1 & \hat{t}_2 \\ \hat{t}_4 & \hat{t}_5 & \hat{t}_6 \\ \hat{t}_8 & \hat{t}_9 & \hat{t}_{10} \end{bmatrix}_k \tag{59}$$

$$\begin{bmatrix} \hat{t}_3 \\ \hat{t}_7 \\ \hat{t}_{11} \end{bmatrix}_{k+1} = \begin{bmatrix} \tilde{t}_0 + 1 & \tilde{t}_1 & \tilde{t}_2 \\ \tilde{t}_4 & \tilde{t}_5 + 1 & \tilde{t}_6 \\ \tilde{t}_8 & \tilde{t}_9 & \tilde{t}_{10} + 1 \end{bmatrix}_{k+1} \begin{bmatrix} \hat{t}_3 \\ \hat{t}_7 \\ \hat{t}_{11} \end{bmatrix}_k + \begin{bmatrix} \tilde{t}_3 \\ \tilde{t}_7 \\ \tilde{t}_{11} \end{bmatrix}_{k+1}. \tag{60}$$

Computing the gradient vector using equation 56 requires the multiplication of matrices $\mathbf{N}(12,3)$, $\mathbf{M}^{-T}(3,3)$ and a vector $\frac{\partial I_2(\mathbf{X})}{\partial \mathbf{X}}$ of size 3. This matrix multiplication if carried out left to right will require 144 arithmetic operations whereas only 45 arithmetic operations are needed when the multiplication is carried out right to left, a factor of three savings in computational time. In our implementation of the above algorithm, we carried out code optimization by eliminating all redundant computations in the gradient computation routine. Note that the gradient vector is computed at every iteration of the algorithm. Using such code optimization, we achieved an additional ten-fold improvement in the computational

performance of our code. Timing (execution time) results for various image registration examples are reported in the next section.

3.3 The combination of Modified Newton and Quasi-Newton methods

When there is significant noise, the second order derivatives term in equation (17) cannot be ignored. Therefore, equation (18) cannot exactly give the Hessian at the optimum. In addition, the Modified Newton method requires that the motion transformations be invertible and closed. These conditions cannot be met when the noise is significant or some parts of the two data sets are inherently quite different. Fortunately, these problems can be successfully solved when the Modified Newton scheme is used in conjunction with the Quasi-Newton method. The Quasi-Newton method computes the approximate Hessian matrix at each iteration point $\hat{\mathbf{T}}^k$ by using the first order derivative $\tilde{\mathbf{g}}(\hat{\mathbf{T}}^k)$. The Quasi-Newton matrix $\mathbf{Q_k}$ will converge to the Hessian matrix at the optimum if the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update (Gill *et al.*, 1981) is used:

$$\mathbf{Q_{k+1}} = \mathbf{Q_k} + \frac{\tilde{\mathbf{g}}(\hat{\mathbf{T}}^k)\tilde{\mathbf{g}}(\hat{\mathbf{T}}^k)^T}{\tilde{\mathbf{g}}(\hat{\mathbf{T}}^k)^T \mathbf{s_k}} + \frac{1}{\mathbf{y_k}^T \mathbf{s_k}} \mathbf{y_k}\mathbf{y_k}^T, \tag{61}$$

where $\mathbf{y_k} = \tilde{\mathbf{g}}(\hat{\mathbf{T}}^{k+1}) - \tilde{\mathbf{g}}(\hat{\mathbf{T}}^k)$, $\mathbf{s_k} = \hat{\mathbf{T}}^{k+1} - \hat{\mathbf{T}}^k$.

Combining the Modified Newton and Quasi-Newton schemes gives the governing iteration as

$$\hat{\mathbf{T}}^{\mathbf{k+1}} = \hat{\mathbf{T}}^k - \mathbf{C_k}^{-1} \tilde{\mathbf{g}}(\hat{\mathbf{T}}^k) \tag{62}$$

where, matrix $\mathbf{C_k}$ is the mean of Hessian matrix at the optimum $\tilde{\mathbf{H}}$ and the Quasi-Newton matrix $\mathbf{C}^{BFGS}$

$$\mathbf{C_k} = \frac{\tilde{\mathbf{H}} + \mathbf{C}_k^{BFGS}}{2}, \tag{63}$$

with $\mathbf{C}_k^{BFGS}$ being the BFGS-Quasi Newton update of $\mathbf{C_{k-1}}$.

The intuitive interpretation of this algorithm is, when the estimate is far away from the optimum, the approximation in equation (61) is poor. Therefore, in order to stabilize this algorithm, we let $\mathbf{C_k}^{BFGS} = \mathbf{0}$ whenever $\mathbf{y}_k^T \mathbf{s}^k < 0$ (implying that C_k^{BFGS} is not positive

definite), yielding the Modified Newton algorithm but with double the step size. This may accelerate the algorithm convergence when far away from the optimum. Whereas, when the estimate is close to the optimum, the approximation in equation (61) is reasonably good and aids the convergence behavior. When the estimate is very close to the optimum, $\mathbf{C_k}^{BFGS}$ will converge to the Hessian at the optimum, thus the combined algorithm in fact automatically switches back to the Modified Newton method. Note that there is no additional overhead in computation of the BFGS-Quasi Newton update as $\tilde{\mathbf{g}}$ has already been computed.

The limitations of this algorithm are that even though the convergence range of the algorithm is quite large to a local optimum, there is always a possibility that the algorithm may diverge as in any local search techniques especially, if the initial guess is too far from the solution. On the other hand, due to the increased convergence range of the modified Newton+quasi-Newton scheme, we have demonstrated (in the next section) the practicality of this algorithm via examples involving large scaling and rotation transformations.

4 Implementation Results

We tested our algorithm with the various types of motions discussed in earlier sections. Our test data are a slice of an MR brain scan for the 2D case and the entire scan for the 3D case. To measure the accuracy of the registration, we used the relative error between the optimum motion vector and computed motion vector defined by using the vector 2-norm. Note that the optimum motion vector is only known for the synthesized data. In the real data case, our test for accuracy currently involves a measure of discrepancy between the AC-PC (anterior and posterior commosure) lines of the registered data sets. Let the optimum motion vector be denoted by $\mathbf{T_{opt}}$, and the computed motion vector by $\hat{\mathbf{T}}$, then the error is defined as

$$error = \frac{\|\hat{\mathbf{T}} - \mathbf{T}_{opt}\|_2}{\|\mathbf{T}_{opt}\|_2}. \tag{64}$$

This measure is used in all our parametric motion experiments to quantify the accuracy of the registration. The measure is more meaningful than the average angle error between computed and true flow vectors used by Barron *et al.* (Barron *et al.*, 1994). In addition, it can be used for (piecewise) global as well as local flow models of motion.

In all the experiments, we always use zero motion as the initial guess to start the minimization iterations.

4.1 Testing with 2D Rigid Motion

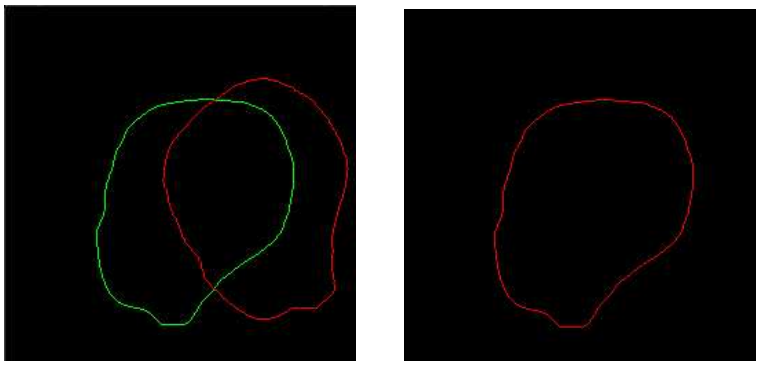
To test our algorithm with 2D rigid motion, we applied a known rigid motion to a slice of size $(256, 256)$ from the MR brain scan and generated a transformed data set. We then used these two data sets as input to the motion estimation algorithm. Table 1 summarizes the results wherein the true and computed motions are depicted in adjacent columns. As evident, the computed motion is extremely accurate. Note that the algorithm can handle large rotations (70°) and translations (30 pixels). In all the rigid motion examples the number of control points used is 25 control points and the number of patches used is 16. One of the advantages of the Modified Newton method is an increase in the size of the region of convergence. Note that normally, the Newton method requires that the initial guess for starting the iteration be reasonably close to the optimum. However, in the modified Newton scheme described earlier, we always used the zero vector as the initial guess for the motion vector to start the iterations. For more details on the convergence behavior of this method, we refer the reader to (Diehl and Burkhardt, 1989).

For visualization purpose, we apply the inverse of the computed transformation to the second image I_2 and then superimpose it on I_1. For clarity of presentation, we choose to depict this process via super-imposition of the head extracted from each of the image data sets. This procedure is used in all the experiments presented in this paper.

Figure 5 depicts images of a pair of heads extracted **only for visualization purposes**. Our registration algorithm was applied to the corresponding intensity image data sets. The subfigure (a) shows a rigid motion consisting of 40° rotation and a translation of $(30, 30)$ pixels. The subfigure (b) shows a super-imposition of the head extracted from the original image and the transformed head using the inverse of the computed transformation. As evident, the registration is perfect.

Table 1: Global 2D rigid motion tests: true and computed motion

True Motion	Computed Motion
$(20°, 4, 2)$	(20.0077, 4.0062, 2.0082)
$(60°, 4, 2)$	(60.0112, 3.9897, 2.0088)
$(70°, 4, 2)$	(69.9816,4.012, 2.0031)
$(40°, 10, 10)$	(40.0092,10.0012,10.0114)
$(40°, 20, 20)$	(40.0119,19.9995,20.0184)
$(40°, 30, 30)$	(40.0246,30.00219,30.024)



(a) (b)

Figure 5: Experimental result for 2D rigid motion. (a) Before registration, where the green head is the model, we applied a 40 degree rotation and $(30, 30)$ translation to the model to generate the red head. (b) After the registration, $\hat{\mathbf{T}}^{-1}$(red head) is superimposed on the original green model, where $\hat{\mathbf{T}}$ is the computed motion vector.

4.2 Testing with 2D Affine Motion

In the next experiment, we tested the (piecewise) global 2D affine motion model by using a similar data generation scheme as used in the rigid motion case. Table 2 summarizes the relative errors for different affine motions. The computed motion results are shown for three different methods namely, the Levenberg-Marquardt method used in Szeliski et al., (Szeliski and Coughlan, 1994) for minimization of the spline-based representation SSD error and the modified Newton method of Burkhardt (Burkhardt and Diehl, 1986) for minimizing the expectation of the squared differences error and our method. This experimental study involved comparing the performance of our algorithm with the performance of our implementation of the published algorithms in Szeliski et al., (Szeliski and Coughlan, 1994) and Burkhardt (Burkhardt and Diehl, 1986). Our implementation of the algorithm in (Szeliski and Coughlan, 1994) was able to reproduce their published results of estimated motion for public domain image data thereby lending credence to our implementation.

Our method is a combination of these two methods and inherits their best features. Note that the 2D affine motion has six global motion parameters and the first column of table 2 depicts the values used for these parameters in our 2D affine motion tests. The values used for the motion parameters in the true motion column included large expansive and contractive scaling as well as rotation and translations. Although Burkhardt's algorithm depicts better accuracy in some of the tests, we observe that it lacks consistent performance in accuracy. Note that our method is quite robust compared to the other two methods and is able to handle large scaling fairly accurately. The double horizontal line in the table has been used to separate the results of testing with different types of affine motion. The first three rows depict a motion which includes an expansive scaling and translation. The next two rows depicts a contractive scaling and a translation while the last two rows depict the full affine motion. When an algorithm yielded over 100% relative error between the optimal motion vector and the computed motion vector after reaching a maximum iteration count, it was deemed to have diverged. Our algorithm takes 13.0 seconds of CPU time on an Ultrasparc-1 to achieve the registration for the affine motion shown in the last row of the table 2. The number of control points and the number of patches used in our algorithm for this experiment

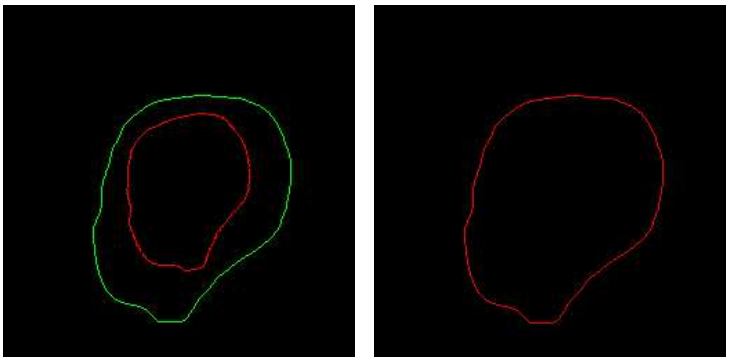
Table 2: Comparison of computed motion using the Szeliski's, Burkhardt's and our method for the affine motion model.

True Motion	Error from Szeliski's	Error from Burkhardt's	Error from ours
(1.1,0.0,0.0,0.0,1.1,0.0)	1.018%	0.60%	0.68%
(1.5,0.0,0.0,0.0,1.5,0.0)	3.114%	19.54%	1.16%
(2.5,0.0,0.0,0.0,2.5,0.0)	diverge	1.45%	1.34%
(0.8,0.0,0.0,0.0,0.8,0.0)	26.32 %	0.75%	0.88%
(0.6,0.0,0.0,0.0,0.6,0.0)	diverge	diverge	4.25%
(1.128,-0.410,2,0.410,1.128,4)	0.69%	0.25%	0.27%
(1.449,-0.388,2,0.388,1.449,4)	1.109%	16.58%	0.54%

are 25 and 16 respectively. Figure 6 is a depiction of the result in the last row of the table 2.

4.3 Testing with 3D Affine Motion

Table 3 summarizes the accuracy comparisons of Szeliski's method, Burkhard's method and our method for the 3D affine motion applied to a volume data set namely, an MR brain scan of size $(128, 128, 35)$. The true motion column contains angle of rotation about a pre-specified arbitrary axis, with uniform scaling in (x,y,z) axis and translation. In our current implementation, the rotation axis is chosen to be the line inclined approximately 10° from the z-axis within the $x = y$ plane. Once again, our method is the most robust of the three. The Szeliski's method here refers to a 3D version of the scheme described in (Szeliski and Coughlan, 1994). In this experiment, we used a single patch with 8 control points leading to a fairly accurate estimate of the synthesized motion. Figure 7 visualizes the result in the first row of the table 3.

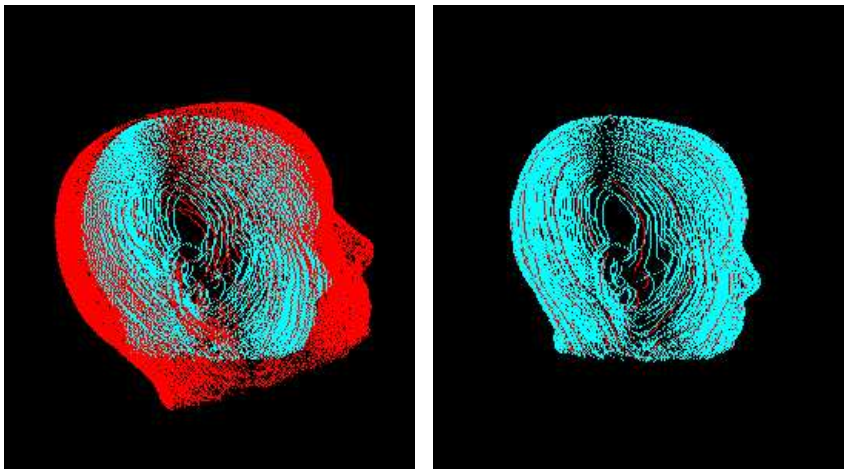


(a) (b)

Figure 6: Experimental result for 2D affine motion. (a) Before registration, where the green head is the model. The red head is generated by applying a 15 degree rotation, (2,4) translation and a contraction by a factor of 1.5. (b) After registration, $\hat{\mathbf{T}}^{-1}$(red head) is superimposed on the green head, where $\hat{\mathbf{T}}$ is the computed motion vector.

Table 3: 3D affine motion test: accuracy comparisons of three methods

True Motion	Error from Szeliski's	Error from Burkhardt's	Error from ours
$(20°, 1.2, 2, 2, 2)$	4.12%	12.3%	0.56%
$(30°, 1.2, 2, 2, 2)$	2.87%	13.3%	0.51%
$(40°, 1.2, 2, 2, 2)$	91.5%	21.5%	2.47%



(a) (b)

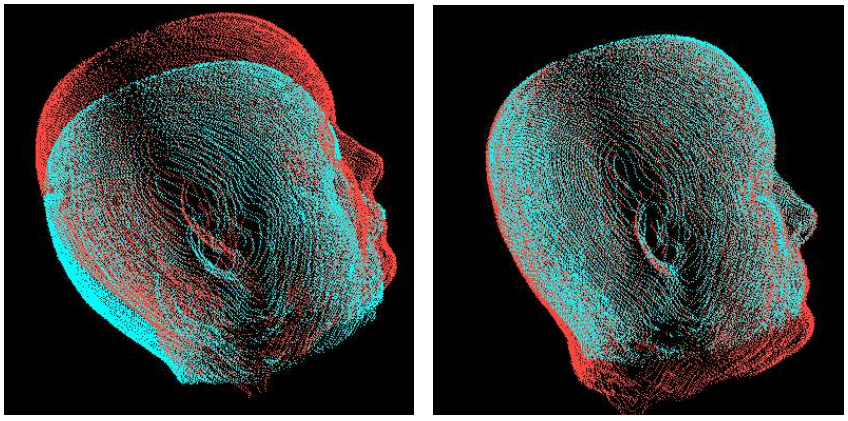
Figure 7: Experimental result for 3D synthesized affine motion. (a) Before registration, where the blue head is an MR scan considered as the model. The red head is generated by applying 20° rotation about an axis inclined approximately 10° from Z-axis within the $x = y$ plane, an expansion factor of 1.2 in (x,y,z) directions and a translation of $(2, 2, 2,)$. (b) After registration. $\hat{\mathbf{T}}^{-1}$(red head) is superimposed on the blue head, where $\hat{\mathbf{T}}$ is the computed motion vector

4.4 Testing with 3D Real Data Sets

Finally, we applied our algorithm in 3D to register three pairs of MR brain scans, each pair belonging to the same individual before and after surgery. The three pairs of data sets are of size $(256, 256, 122) \times (256, 256, 124)$, $(256, 256, 119) \times (256, 256, 119)$, and $(256, 256, 105) \times (256, 256, 105)$ respectively. We applied our algorithm to the raw 3D data, computed the motion using 8 control points and a single patch in the (u, v, w) representation, and then applied the inverse of the computed transformation to the second data set. For visualization purposes, we extracted the heads from the first and transformed second data sets and superimposed them as shown in figure 8. The registration appears to be visually quite accurate and the accuracy was verified by observing that the AC-PC lines of the two data sets were fully aligned. The evident mis-registration could be attributed to the error in the head shape extraction process performed for the visualization purposes.

We can use the global registration results as input to the local flow estimation process to refine the registration if and when needed. In our examples however, we did not get any significant improvement in the registration after applying the local flow estimator and hence we do not depict the local flow results. Our global flow estimation algorithm takes 9.2 *mins.*, 1.8 *mins* and 1.35 *mins*. CPU time – on an Ultra-Sparc-1 to register the above mentioned pairs of MR brain scans respectively.

Figure 8 only gives a rough idea of how good the registration is. A more meaningful registration validation is presented in figure 9, showing the 3D registration at an arbitrary cross-section. We used a slice from the pre-operative MR scan as the underlay and superimposed the corresponding slice from the 3D edge map of the transformed post-operative MR scan on it to show the registration effect. The registration is visually perfect and the missing part(the surgically removed tissue in the middle of the head) is well localized. Figures 10 and 11 depict the 3D registration at an arbitrary slice for two additional pairs of MR brain scans.



(a) (b)

Figure 8: Experimental result for 3D real data. (a) Before registration, where the red head is a pre-operative scan considered as the model. The blue head is the post-operative scan. (b) After registration. $\hat{\mathbf{T}}^{-1}$(blue head) is superimposed on the red head, where $\hat{\mathbf{T}}$ is the computed motion vector

5 Conclusion

In this paper, we presented a novel image registration algorithm that incorporates the modified Newton method into the hierarchical spline-based optical flow framework. The modified Newton method described here has a larger region of convergence and is computationally more efficient - since the Hessian matrix is precomputed - in comparison to the traditional Newton scheme. In addition, the spline-based representation of the flow field possesses the property of built-in smoothness. All of these features lead to our algorithm having the following strengths, namely, it can be applied directly to raw image data, it is very robust/reliable, it can cope with large motion including scaling and is computationally efficient – it takes 1.8mins. register two MRI scans each of size $(256, 256, 119)$ on an Ultra Sparc-1. Our algorithm has the capability to handle global as well as local motions and proceeds to first globally register the data sets and then refines the registration locally if necessary. The effectiveness of the registration was measured quantitatively and the percentage error in registration achieved by using our algorithm was compared with competing methods (Szeliski and Coughlan, 1994;

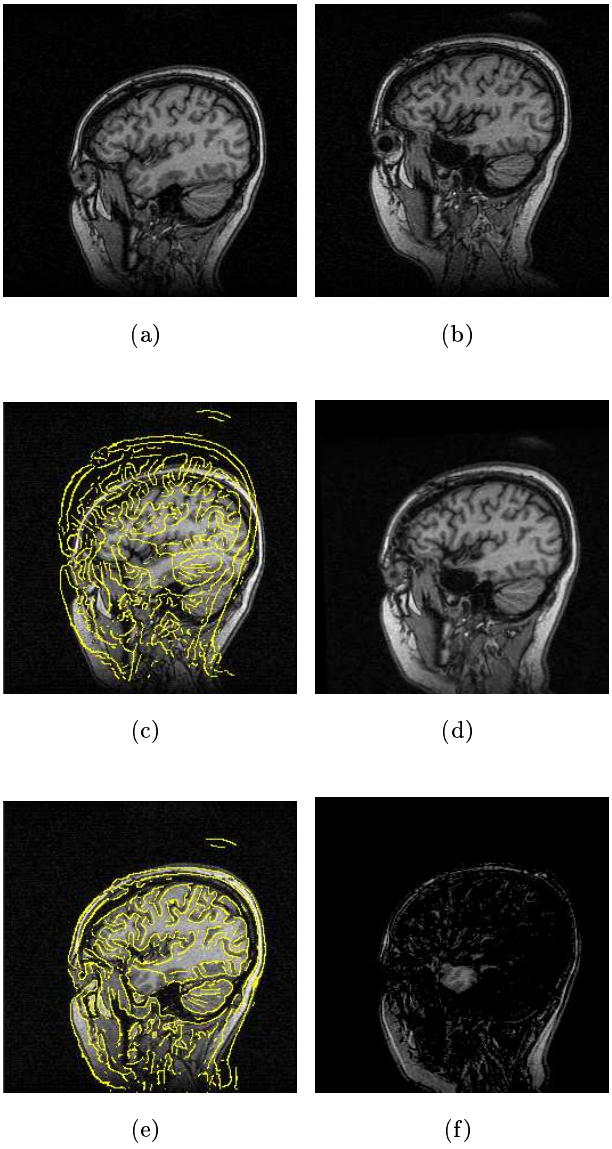


Figure 9: 3D registration depicted along an arbitrary slice $z = 30$. (a) A slice from pre-operative MR scan before registration. (b) A slice from post-operative MR scan before registration. (c) A slice from the 3D edge map of post-operative MR scan superimposed on (a) prior to registration. (d) A slice from $\hat{\mathbf{T}}^{-1}$(post-operative MR scan) after registration. (e)A slice of the 3D edge map of the transformed post-operative MR scan superimposed on (a) after registration. (f) A slice representing the difference between (a) and (d).

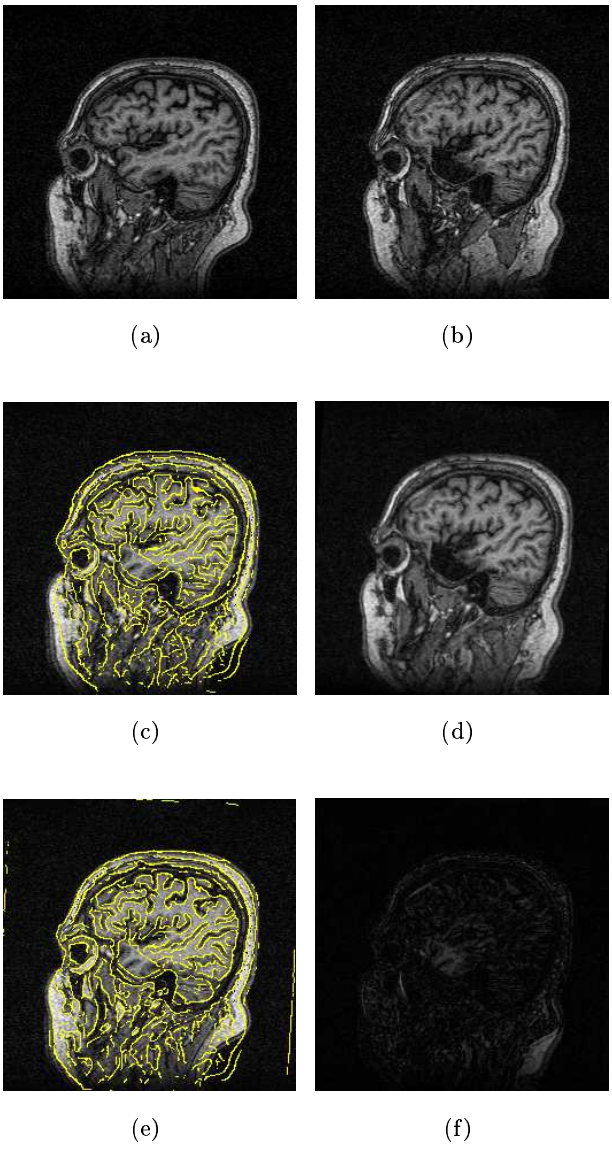
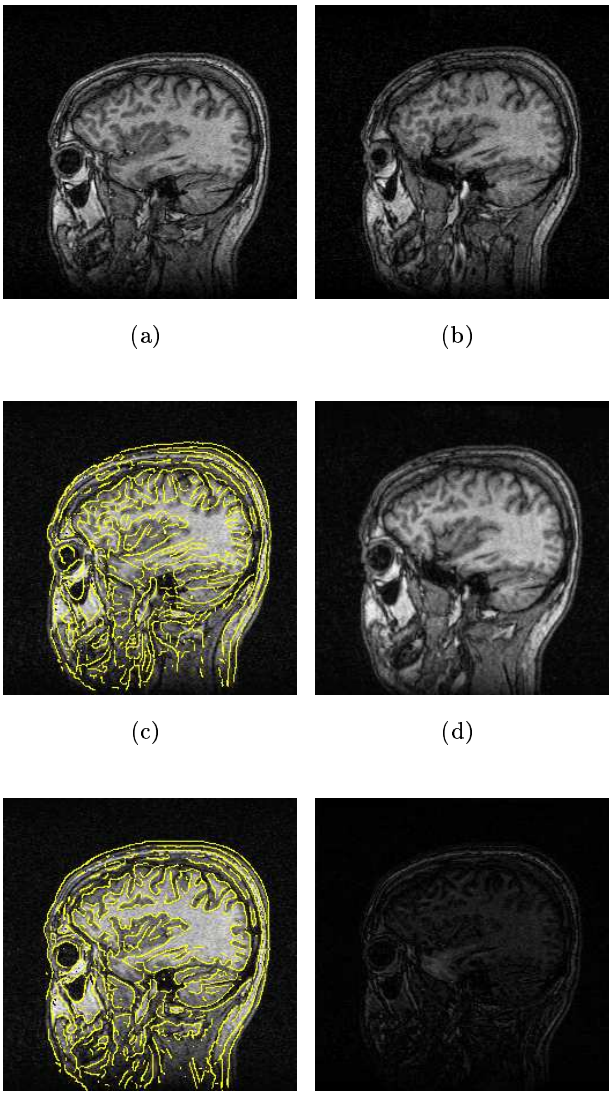


Figure 10: 3D registration depicted along an arbitrary slice $z = 34$. (a) A slice from pre-operative MR scan before registration. (b) A slice from post-operative MR scan before registration. (c) A slice from the 3D edge map of post-operative MR scan superimposed on (a) prior to registration. (d) A slice from $\hat{\mathbf{T}}^{-1}$(post-operative MR scan) after registration. (e)A slice of the 3D edge map of the transformed post-operative MR scan superimposed on (a) after registration. (f) A slice representing the difference between (a) and (d).



(a) (b)

(c) (d)

(e) (f)

Figure 11: 3D registration depicted along an arbitrary slice $z = 27$. (a) A slice from pre-operative MR scan before registration. (b) A slice from post-operative MR scan before registration. (c) A slice from the 3D edge map of post-operative MR scan superimposed on (a) prior to registration. (d) A slice from $\hat{\mathbf{T}}^{-1}$(post-operative MR scan) after registration. (e)A slice of the 3D edge map of the transformed post-operative MR scan superimposed on (a) after registration. (f) A slice representing the difference between (a) and (d).

Burkhardt and Diehl, 1986). For the various motion models described in this paper, it was shown that our algorithm outperforms the competing methods in robustness. Our future efforts will be focused on making it more practical by involving more testing with clinical data and implementing a user friendly interface.

6 Appendix

In this appendix, we first present the details of the derivations for equation (27) and then the derivation for equations (30) and (31).

Let $I_T(\mathbf{X}, \mathbf{T})$ result from reference image $S(\mathbf{X})$ under the rigid moion $\mathbf{T} = (\phi, d_1, d_2)$

$$I_T(\mathbf{X}, \mathbf{T}) = S(x_1\ cos\phi - y_1\ sin\phi - d_1,\ x_1\ sin\phi + y_1\ cos\phi - d_2) \tag{65}$$

Let

$$\left\{ \begin{array}{lcl} x_2 & = & x_1\ cos\phi - y_1\ sin\phi - d_1 \\ y_2 & = & x_1\ sin\phi + y_1\ cos\phi - d_2 \end{array} \right. \tag{66}$$

$\Longrightarrow I_T(\mathbf{X}, \mathbf{T}) = S(x_2, y_2)$

$$\stackrel{chain\ rule}{\Longrightarrow} \left\{ \begin{array}{lcl} \frac{\partial s}{\partial x_2} & = & \frac{\partial s}{\partial x_1}\frac{\partial x_1}{\partial x_2} + \frac{\partial s}{\partial y_1}\frac{\partial y_1}{\partial x_2} \\ \frac{\partial s}{\partial y_2} & = & \frac{\partial s}{\partial x_1}\frac{\partial x_1}{\partial y_2} + \frac{\partial s}{\partial y_1}\frac{\partial y_1}{\partial y_2} \end{array} \right. \tag{67}$$

inverting (66) we can get:

$$\left\{ \begin{array}{lcl} x_1 & = & (x_2 + d_1)\ cos\phi + (y_2 + d_2)\ sin\phi \\ y_1 & = & -(x_2 + d_1)\ sin\phi + (y_2 + d_2)\ cos\phi \end{array} \right. \tag{68}$$

$$\Longrightarrow \left\{ \begin{array}{lcl} \frac{\partial x_1}{\partial x_2} & = & cos\phi \\ \frac{\partial x_1}{\partial y_2} & = & sin\phi \end{array} \right. \quad and \quad \left\{ \begin{array}{lcl} \frac{\partial y_1}{\partial x_2} & = & -sin\phi \\ \frac{\partial y_1}{\partial y_2} & = & cos\phi \end{array} \right. \tag{69}$$

Substituting (69) into (67), we get,

$$\left\{ \begin{array}{lcl} \frac{\partial s}{\partial x_2} & = & cos\phi\ \frac{\partial s}{\partial x_1} - sin\phi\ \frac{\partial s}{\partial y_1} \\ \frac{\partial s}{\partial y_2} & = & sin\phi\ \frac{\partial s}{\partial x_1} + cos\phi\ \frac{\partial s}{\partial y_1} \end{array} \right. \tag{70}$$

Making use of eqn (70), gives

$$\begin{aligned}\frac{\partial I_T}{\partial d_1} &= \frac{\partial s}{\partial x_2}\,\frac{\partial x_2}{\partial d_1} + \frac{\partial s}{\partial y_2}\,\frac{\partial y_2}{\partial d_1} \\ &= \frac{\partial s}{\partial x_2}\cdot(-1) + \frac{\partial s}{\partial y_2}\cdot 0 \\ &= -\frac{\partial s}{\partial x_2} \\ &= -cos\phi\ \frac{\partial s}{\partial x_1} + sin\phi\ \frac{\partial s}{\partial y_1}\end{aligned} \tag{71}$$

$$\begin{aligned}\frac{\partial I_T}{\partial d_2} &= \frac{\partial s}{\partial x_2}\,\frac{\partial x_2}{\partial d_2} + \frac{\partial s}{\partial y_2}\,\frac{\partial y_2}{\partial d_2} \\ &= \frac{\partial s}{\partial x_2}\cdot 0 + \frac{\partial s}{\partial y_2}\cdot(-1) \\ &= -\frac{\partial s}{\partial y_2} \\ &= -sin\phi\ \frac{\partial s}{\partial x_1} - cos\phi\ \frac{\partial s}{\partial y_1}\end{aligned} \tag{72}$$

$$\begin{aligned}and\ \ \frac{\partial I_T}{\partial \phi} &= \frac{\partial s}{\partial x_2}\,\frac{\partial x_2}{\partial \phi} + \frac{\partial s}{\partial y_2}\,\frac{\partial y_2}{\partial \phi} \\ &= \frac{\partial s}{\partial x_2}(-x_1\ sin\phi - y_1\ cos\phi) + \frac{\partial s}{\partial y_2}(x_1\ cos\phi - y_1\ sin\phi) \\ &= -y_1\ \frac{\partial s}{\partial x_1} + x_1\ \frac{\partial s}{\partial y_1}\end{aligned} \tag{73}$$

The above is the general derivation of derivatives with respect to the rigid motion $\mathbf{T}$ for any given reference image and rigid motion. Now, let us derive the conditions at the optimum. Suppose optimum is reached at iteration step N, then $\tilde{\mathbf{T}}^{N+1} = (\tilde{\phi}^{N+1}, \tilde{d_1}^{N+1}, \tilde{d_2}^{N+1}) = 0$

$$\begin{aligned}\Longrightarrow\ \ & \frac{\partial I_m(\mathbf{X}, \hat{\mathbf{T}}^N \circ \tilde{\mathbf{T}}^{N+1})}{\partial \tilde{d_1}^{N+1}}\bigg|_{\tilde{\mathbf{T}}^{N+1}=0} \\ &= \frac{\partial I_1(\mathbf{X}^N, \tilde{\mathbf{T}}^{N+1})}{\partial \tilde{d_1}^{N+1}}\bigg|_{\tilde{\mathbf{T}}^{N+1}=0} \\ &= -cos\tilde{\phi}^{N+1}\ \ \frac{\partial I_1(\mathbf{X}^N)}{\partial x_1^N}\ +\ sin\tilde{\phi}^{N+1}\ \ \frac{\partial I_1(\mathbf{X}^N)}{\partial y_1^N}\bigg|_{\tilde{\mathbf{T}}^{N+1}=0} \\ &= -\frac{\partial I_1(\mathbf{X}^N)}{\partial x_1^N}\end{aligned} \tag{74}$$

$$\begin{aligned}\Longrightarrow\ \ & \frac{\partial I_m(\mathbf{X}, \hat{\mathbf{T}}^N \circ \tilde{\mathbf{T}}^{N+1})}{\partial \tilde{\phi}^{N+1}}\bigg|_{\tilde{\mathbf{T}}^{N+1}=0} \\ &= \frac{\partial I_1(\mathbf{X}^N, \tilde{\mathbf{T}}^{N+1})}{\partial \tilde{\phi}^{N+1}}\bigg|_{\tilde{\mathbf{T}}^{N+1}=0} \\ &= -y_1^N\ \ \frac{\partial I_1(\mathbf{X}^N)}{\partial x_1^N}\ +\ x_1^N\ \ \frac{\partial I_1(\mathbf{X}^N)}{\partial y_1^N}\bigg|_{\tilde{\mathbf{T}}^{N+1}=0} \\ &= -y_1^N\ \ \frac{\partial I_1(\mathbf{X}^N)}{\partial x_1^N}\ +\ x_1^N\ \ \frac{\partial I_1(\mathbf{X}^N)}{\partial y_1^N}\end{aligned} \tag{75}$$

This is the eqn (27) on Page 14, where $\mathbf{X}^N = (x_1^N, y_1^N),\ \mathbf{X} = (x_1, y_1)$.

We now present the derivations of equations (30) and (31). For a better understanding, we can change eqn. (29) to,

$$\begin{aligned} \frac{\partial I_1(\mathbf{X}^*)}{\partial \tilde{\mathbf{T}}} &= (\frac{\partial \mathbf{X}^*}{\partial \tilde{\mathbf{T}}})^T \ \frac{\partial I_1(\mathbf{X}^*)}{\partial \mathbf{X}^*} \\ &= (\frac{\partial \mathbf{X}}{\partial \tilde{\mathbf{T}}})^T \ \frac{\partial \mathbf{X}^*}{\partial \mathbf{X}} \ \frac{\partial I_1(\mathbf{X}^*)}{\partial \mathbf{X}^*} \ . \end{aligned} \tag{76}$$

Substituting into eqn. (19), yields

$$\begin{aligned} \tilde{H} &= 2E\{\frac{\partial I_1(\mathbf{X}^*)}{\partial \tilde{\mathbf{T}}} \ (\frac{\partial I_1(\mathbf{X}^*)}{\partial \tilde{\mathbf{T}}})^T\} \mid_{\tilde{\mathbf{T}}=0} \\ &= 2E\{(\frac{\partial \mathbf{X}}{\partial \tilde{\mathbf{T}}})^T \ \frac{\partial \mathbf{X}^*}{\partial \mathbf{X}} \ \frac{\partial I_1(\mathbf{X}^*)}{\partial \mathbf{X}^*} \ (\frac{\partial I_1(\mathbf{X}^*)}{\partial \mathbf{X}^*})^T \ (\frac{\partial \mathbf{X}^*}{\partial \mathbf{X}})^T \ \frac{\partial \mathbf{X}}{\partial \tilde{\mathbf{T}}}\} \mid_{\tilde{\mathbf{T}}=0} \end{aligned}$$

which is in the form of eqn. (30).

$$\begin{aligned} &= 2E\{(\frac{\partial \mathbf{X}}{\partial \tilde{\mathbf{T}}})^T \ \frac{\partial I_1(\mathbf{X})}{\partial \mathbf{X}} \ (\frac{\partial I_1(\mathbf{X})}{\partial \mathbf{X}})^T \ \frac{\partial \mathbf{X}}{\partial \tilde{\mathbf{T}}}\} \mid_{\tilde{\mathbf{T}}=0} \\ &= 2E\{(\frac{\partial h(\mathbf{X},\mathbf{T})}{\partial \mathbf{T}})^T \ \frac{\partial I_1(\mathbf{X})}{\partial \mathbf{X}} \ (\frac{\partial I_1(\mathbf{X})}{\partial \mathbf{X}})^T \ \frac{\partial h(\mathbf{X},\mathbf{T})}{\partial \mathbf{T}}\} \mid_{\mathbf{T}=0} \end{aligned} \tag{77}$$

which is eqn. (31).

References

Aggarwal, J. K. and Nandhakumar, N. (1988) On the Computation of Motion from a Sequences of Images – a Review. *Proc. of the IEEE*, 76(8):917-935.

Bajscy, R. and Kovacic, S. (1989) Multiresolution Elastic Matching. *Computer Vision, Graphics and Image Processing*, vol. 46, pages 1-21.

Barron, J. L., Fleet, D. J. and Beauchemin, S. S.(1994) Performance of Optical Flow Techniques. *Intern. J. Comput. Vision*, 1(12):43-77.

Black, M. J. and Anandan, P. (1993) A Framework for Robust Estimation of Optical Flow. *Proc. Intl. Conf. on Compu. Vision (ICCV)*, pages 231-236, Berlin, Germany.

Borgefors, G., (1988) Hierarchical chamfer matching: a parametric edge matching algorithm. *IEEE Trans. on PAMI*, 10, 849-865.

Bro-Nielsen, M. and Gramkow, C. (1996) Fast Fliud Registration of Medical Images. *Proc. of the IV Intl. Conf. on Visualization in Biomedical Computing*, pages 267-284, Hamburg, Germany.

Burkhardt, H. and Diehl, N. (1986) Simultaneous Estimation of Rotation and Translation in Image Sequences. *Proc. of the European Signal Processing Conference*, Elsevier Science Publishers B.V. (North-Holland), pages 821-824.

Christensen, G. E., Miller, M. I. and Vannier, M. (1996) Individualizing Neuroanatomical Atlases Using a Massively Parallel Computer. *IEEE Computer*, 1(29):32-38.

Collignon, A., Maes, F., Delaere, D., Vandermeulen, D., Suetens P., and Marchal., G., (1995) Automated multimodality image registration using information theory, In Bizais, Y., Barillot, C., and Di paoloa, R., (eds.), *Proc. Information Processing in Medical Imaging (Brest)*, pp. 263-274. Kluwer, Dordrecht.

Davatzikos, C. and Prince, J, L. (1994) Brain Image Registration Based on Curve Mapping. *IEEE Workshop on Biomedical Image Analysis*, pages 245-254, Seattle, WA.

Diehl, N. and Burkhardt, H. (1989) Motion Estimation in Image Sequences. *High Precision Navigation*, pages 297-312, Springer-Verlag, New York.

Duda, R. O. and Hart P. E. (1973) *Pattern Classification and Scene Analysis*, John Wiley and Sons.

Duncan, J. H. and Chou, T. C. (1992) On the Detection of Motion and the Computation of Optical Flow. *IEEE Trans. Patt. Anal. Mach. Intell.*, PAMI-14(3):346-352.

Evans, A. C., Dai, W., Collins, L., Neeling P. and Marett S. (1991) Warping of Computerized 3D Atlas to Match Brain Image Volumes for Quantitative Neuroanatomical and Functional Analysis. *Proc. SPIE Medical Imaging V*, vol. 1445, pages 236-246.

Feldmar, J. and Ayache, N. (1994) Locally Affine Registration of Free-form Surfaces. *Proc. of IEEE CVPR*, pp. 496-501, Seattle, WA.

Gill, P.E., Wright, W. and H., M. (1981) *Practical Optimization.* Academic Press, London.

Gueziec, A. and Ayache, N. (1992) Smoothing and matching of 3D space curves. In Robb. R. (ed.), *Visualization in Biomedical Computing*, *Proc. SPIE*, 1808, pp. 259-273. SPIE Press, Bellingham, WA.

Meyer, C. R., Boes, J. L., Kim, B., Bland, P. H., Zasadny, K. R., Kison, P. V., Koral, K., Frey, K. A., Wahl, R. L. (1997) Demonstrating the accuracy and clinical versatility of mutual information for automatic multimodality image fusion using affine and thin-plate spline warped geometric deformations. *Medical Image Analysis*, Vol. 1., No. 3, pp. 195-206.

Monga, O., Benayoun, S. and Faugeras, O. D. (1992) From partial derivatives of 3D density images to ridge lines. Robb. R. A. (ed.), *Visualization in Biomedical Computing*, *Proc. SPIE*, 1808, pp. 118-127. SPIE Press, Bellingham, WA.

Gupta, S. and Prince, J. (1995) On Variable Brightness Optical Flow for Tagged MRI. *Proc. of IPMI*, pp. 323-334, Dordredcht, Kluwer.

Horn, B. P. K. and Schunk, B. G. (1981) Determining optical flow. *Artificial Intelligence*, Vol. 17, pp. 185-203.

Ju, S. X., Black, M. J. and Jepson, A. D. (1996) Skin and Bones: Multi-layer, Locally Affine, Optical Flow and Regularization With Transparency. *Proc. of CVPR'96*, pages 307-314, San Fransico, CA.

Lai, S. H. and Vemuri, B. C. (1995) Robust and Efficient Computation of Optical Flow. *IEEE Symposium on CVS*, pages 455–460, Miami, Florida.

Maintz, J. B. A., van den Elsen P. A., and Viergever M. A., (1996) Comparison of edge-based and ridge-based registration of CT and MR brain images. *Medical Image Analysis*, 1(2), pp. 151-161.

Nagel, H. H. and Enkelmannm, W. (1986) An Investigation of Smoothness Constraints for the Estimation of Displacement Vector Fields from Image Sequences. *IEEE Trans. Patt. Anal. Mach. Intell.*, 8(5), pp. 565–593.

Okuomi, M. and Kanade, T. (1992) A Locally Adaptive Window for Signal Matching. *International Journal of Computer Vision*, 7(2), pp. 143-162.

Pelizarri, C. A., Chen, G. T. Y. and Spelbring, D. R. (1989) Accurate Three-Dimensional Registration of CT, PET and MR Images of the Brain. *J. Comput. Assist. Tomogr.*, 13(1), pp.20-26.

T. Schormann, S. Henn and K. Zilles. A New Approach to Fast Elastic Alignment with Application to Human Brains. *Visualization in Biomedical Computing*, pp. 338-342, Hamburg, Germany.

Studholme , C., Hill and Hawkes, D. J. Automated 3D registration of MR and CT images in the head. *Medical Image Analysis*, 1(2), pp. 163-175.

Szeliski, R. and Coughlan, J. (1994) Hierarchical Spline-Based Image Registration. *IEEE Conf. Comput. Vision Patt. Recog.*, pages 194-201, Seattle, WA.

Thirion, J-P. (1994) Extremal Points: Definition and Application to 3D Registration. *Proc. of IEEE CVPR*, pp. 587-592, Seattle, WA.

van den Elsen, P. A. (1993) Multimodality matching of brain images. *Ph.D. Thesis*, Utrecht University, The Netherlands.

Alignment by maximization of mutual information, In *Proc. of the IEEE Intl. Conf. on Computer Vision*, (Boston), pp. 15-23.

Wells III, W. M., Viola, P. and Atsumi, H. (1996) Multi-modal Volume Registration by Maximization of Mutual Information. *Medical Image Analysis*, 1(1), pp. 35-51.

Zhao, W., Young, T. Y., and Ginsberg, M. D. (1993) Registration of three-dimensional autoradiographic images by the disparity analysis method. *IEEE Transactions on Medical Imaging*, Vol. 12, No. 4, pp. 782-791.