Automatic Imagery Registration Toolkit: Concept, Implementation, and Analysis

May Arsenovic and Patrick North
Rochester Institute of Technology
Chester F. Carlson Center for Imaging Science
15 November 2007

1.0 Introduction

It is rare that a single remote sensing platform will have the capability to collect an entire range of data required for a given task. Often imagery is merged together from a variety of sensors to produce a data set with the desired resolution and bandwidth. The trade-off of using multiple modalities to collect imagery is that the resulting raw data is seldom registered. Two corresponding pixels in two different bands of data will most likely not correspond to the same location on the ground. The two un-registered bands can be fused together using either a brute-force manual registration technique or an automatic process. Obviously, having a robust automatic process is significantly less time consuming and potentially more consistent.

The process proposed automatically registers two image bands together without any preliminary knowledge of scene content. Un-registered image bands are first passed through a pre-processing filter to reduce noise and enhance the hard edges present in both bands. The resulting noise-reduced bands are then sent through a corner detection process to isolate image features points. Using a feature correspondence algorithm, the feature points in both bands are compared and matched together. The preliminary matching of feature points produces a rough-guess affine transformation matrix. This matrix represents the algorithm's best guess at the geometrical transform needed to register the bands together. Using this first matrix and the images' mutual information as a metric, the affine transform matrix is optimized using two methods, particle swarm optimization and a traditional Levenberg-Marquardt optimization.

2.0 Theory

2.1 Bilateral Filtering

With several traditional filtering methods for removing image noise, hard edges are often lost. For example, a Gaussian filter will do a satisfactory job of removing any noise from an image, but the resultant image will most likely be blurry. For this particular application, loosing the edges in the image would have been detrimental to the performance of the corner detector and therefore not acceptable. Bilateral filtering offers a method to remove the noise from the image while simultaneously preserving the edges.

Bilateral filtering uses a combination of traditional low-pass filtering and range filtering. A low-pass filter performs domain filtering. Each pixel is convolved with a filter kernel. Each element of that kernel is given a weight based on the kernel elements distances from the center pixel. Thus, the effect of each surrounding pixel on any given center pixel is based on its distance from the given center pixel. A range-filtering kernel will set each element's weight based on that element's difference in intensity from the center pixel. A bilateral filter performs domain filtering in conjunction with range filtering. One filter is based on a pixel's physical distance from the center pixel while the other is based on a pixel's photometric similarity to the center pixel. For example, given an image $\mathbf{f(x)}$, the resulting filtered image will be $\mathbf{h(x)}$ (2.1-1). To perform domain filtering, the geometrical kernel is represented by $c(\xi, \mathbf{x})$. The

center pixel is represented by **x** and the nearby point by ξ. The geometric kernel is shift-invariant and only a function of ξ-**x**.

$$\mathbf{h}(\mathbf{x}) = k_d^{-1}(\mathbf{x}) \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mathbf{f}(\xi) c(\xi, \mathbf{x}) d\xi \tag{2.1-1}$$

The normalization constant k_d is in place to preserve the DC component of the low-pass signals (2.1-2).

$$k_d = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} c(\xi, \mathbf{x}) d\xi \tag{2.1-2}$$

Similarly, range filtering is defined in equation (2.1-3). The photometric similarity kernel, s(**f**(ξ), **f**(**x**)), only depends on the difference **f**(ξ)-**f(**x**)**.

$$\mathbf{h}(\mathbf{x}) = k_r^{-1}(\mathbf{x}) \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mathbf{f}(\xi) s(\mathbf{f}(\xi), \mathbf{f}(\mathbf{x})) d\xi \tag{2.1-3}$$

The normalization constant for range filtering is shown in equation (2.1-4).

$$k_r(\mathbf{x}) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} s(\mathbf{f}(\xi), \mathbf{f}(\mathbf{x})) d\xi \tag{2.1-4}$$

The combination of both of these types of filters is a bilateral filter (2.1-5).

$$\mathbf{h}(\mathbf{x}) = k^{-1}(\mathbf{x}) \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mathbf{f}(\xi) c(\xi, \mathbf{x}) s(\mathbf{f}(\xi), \mathbf{f}(\mathbf{x})) d\xi \tag{2.1-5}$$

The normalization for the bilateral filter is defined below and is used to ensure that the bilateral kernel weights add up to one (2.1-6).

$$k(\mathbf{x}) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} c(\xi, \mathbf{x}) s(\mathbf{f}(\xi), \mathbf{f}(\mathbf{x})) d\xi \tag{2.1-6}$$

Borrowed from the text by Tomasi and Manduchi, the following graphic shows the results of a bilateral filter on a simple, noisy edge.

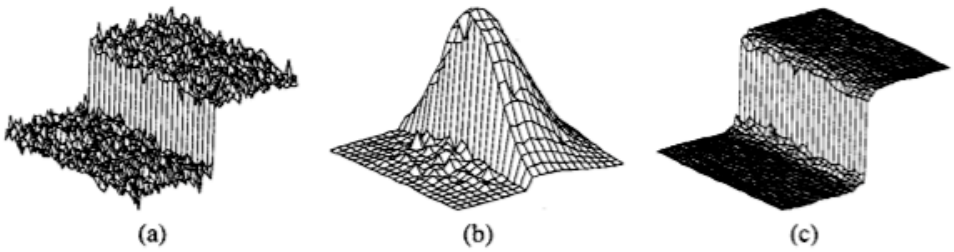


Figure 1: (a) A 100-gray-level step perturbed by Gaussian noise with $\sigma = 10$ gray levels. (b) Combined similarity weights $c(\xi, \mathbf{x}) s(\mathbf{f}(\xi), \mathbf{f}(\mathbf{x}))$ for a 23×23 neighborhood centered two pixels to the right of the step in (a). The range component effectively suppresses the pixels on the dark side. (c) The step in (a) after bilateral filtering with $\sigma_r = 50$ gray levels and $\sigma_d = 5$ pixels.

In this example, the kernel is centered over a pixel in the bright portion of the noisy edge (figure 1(a)). Figure 1(b) shows the resulting bilateral filter for this situation. In the bright region of this edge, the pixel values are relatively similar to one another and will be given weights close to one. In this manner, the bilateral filter behaves similar to a traditional low-pas filter and only removes the small discrepancies between pixels, or noise. However, any effect the dark side of the edge would normally have in the low-pass filtering of a bright pixel is negated by the range filter. The elements of the kernel positioned over the dark pixels obtain weights closer to zero because they are dissimilar to the center pixel in intensity. The resulting edge, figure 1(c), is preserved and the noise is removed. The bilateral filter used for the pre-processing in the registration technique used a Gaussian distribution for the weights of both the domain and range filters[1].

2.2 Harris Corner Detection

Finding sets of corresponding features between different bands and using these to determine the transformation between the two bands is a method of registration. The question of what a good feature is becomes a significant issue. Important properties of features are that they are fast to calculate, well defined in order to precisely locate them, and invariant under many different orientations, scales, modalities, illumination, and thermal conditions. Corners meet all of these criteria and are one of the most common features used in registration algorithms found in the literature.

The first step in calculating the "cornerness" involves taking the gradient of the image along the x- and y-axis, identified as the image gradients I_x and I_y respectively. These gradients can be calculated as seen in figure 2.2-1.

$$I_x(x,y) = \frac{\partial I(x,y)}{\partial x}$$
$$I_y(x,y) = \frac{\partial I(x,y)}{\partial y}$$

Figure 2.2-1

In order to reduce the effect of noise and local variations from texture the average gradient can be used rather than the local derivative, as is recommended in many corner detection papers. Convolving the image gradients with a small symmetric Gaussian weighting kernel can perform this process. In this suggested registration process a Gaussian kernel with a distance sigma value of 2 pixels is used to remove noise and calculate the average image gradient. The average image gradient in the x- and y-axis can be identified as $\overline{I_x(x,y)}$ and $\overline{I_y(x,y)}$ as shown in figure 2.2-2 where the weighting kernel is denoted as w.

$$\overline{I_x(x,y)} = I_x(x,y) * w$$
$$\overline{I_y(x,y)} = I_y(x,y) * w$$

Figure 2.2-2

Finally the cornerness can be measured by organizing the magnitudes of the image gradients into a 2x2 matrix, denoted as the **C** matrix in figure 2.2-3, and using eigen analysis. The idea of using this square cornerness matrix is that if the magnitudes of the image gradient are strong in orthogonal directions then this matrix will have two large eigenvalues. If the image gradient is strong in a single direction then this matrix will only have one large eigen value. If the image gradient is not very strong in any direction then the matrix will not have any large eigenvalues.

$$\mathbf{C}(x,y) = \begin{bmatrix} \overline{I_x(x,y)}^2 & \overline{I_x(x,y)I_y(x,y)} \\ \overline{I_x(x,y)I_y(x,y)} & \overline{I_y(x,y)}^2 \end{bmatrix} = \begin{bmatrix} a & c \\ c & b \end{bmatrix}$$

Figure 2.2-3

Since the **C** matrix is symmetric calculating the eigenvalues is a straightforward process. The derivation of these is given below if figure 2.2-4. An image of the second eigenvalue is displayed in figure 2.2-5 using a linear scale.

$$\left|\begin{bmatrix} a-\lambda & c \\ c & b-\lambda \end{bmatrix}\right| = 0$$

$$(a-\lambda)(b-\lambda)-c^2 = 0$$

$$ab-a\lambda-b\lambda+\lambda^2-c^2 = 0$$

$$\lambda^2-(a+b)\lambda+ab-c^2 = 0$$

$$t_1 = 1$$

$$t_2 = -(a+b)$$

$$t_3 = ab-c^2$$

$$\lambda_1 = \frac{-t_2+\sqrt{{t_2}^2-4t_1t_3}}{2t_1}$$

$$\lambda_2 = \frac{-t_2-\sqrt{{t_2}^2-4t_1t_3}}{2t_1}$$

Figure 2.2-4

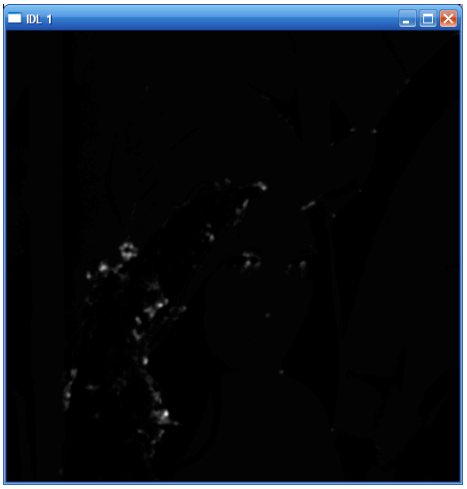


Figure 2.2-5

Once the eigenvalues λ_1 and λ_2 have been calculated then a corner metric must be used to differentiate corners from non-corners. Examples of possible corner metrics include the second eigenvalue (m_1), a ratio of the eigenvalues (m_2), subtracting the trace from the determinant (m_3), and finally subtracting a fraction of the squared trace from the determinant (m_4).

$$m_1 = \lambda_2$$

$$m_2 = \lambda_1\lambda_2 / (\lambda_1+\lambda_2)$$

$$m_3 = \lambda_1\lambda_2 - (\lambda_1+\lambda_2)$$

$$m_4 = \lambda_1\lambda_2 - k(\lambda_1+\lambda_2)^2$$

Figure 2.2-6

Once the corner metric has been selected, m_4 for this proposed process, this value can be used to select maximum corner points to use as features. The selection method should ensure that the strongest features are selected, the features are not too clustered together, the features are consistent for different modalities and phenomenology, and that a sufficient, but not excessive number, of corners are selected. In order to meet these four criteria the following method was used, written up as pseudo-code:

```
Sort corner metric values from image
Select threshold value a specified penetration depth into the sorted values
[default 80%]
While number of selected corners is less than max number of corners [default
100] and max corner value less than threshold value
       Select maximum remaining corner metric as a new corner feature point
       Mask out surrounding corner metric values from the selected corner within
       a certain number of pixels [default 10 pixels]
End While
```

This method enables a great deal of flexibility with the different parameters that control the corner selection method while still following the same set of rules. For example with drastically different modalities using a greater separation between corner points increased the corresponding corners selected between the two bands. Similarly by allowing two methods to limit the number of corner points selected

this ensures that too many corners will not be selected. The number of features in the band sets will be the limiting factor in computational performance in the next phase when finding feature correspondences between the bands.

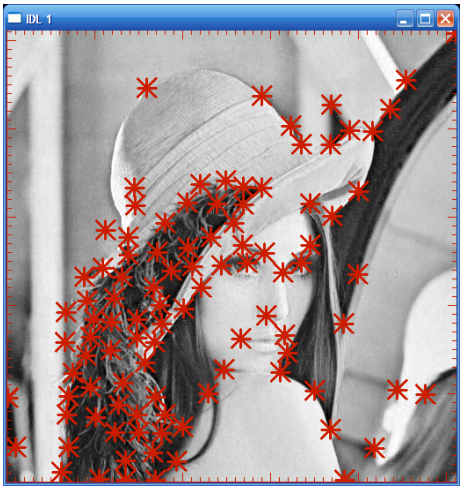


Figure 2.2-7

2.3 Feature Correspondence

The basis of this registration process is to quickly and efficiently find a coarse correspondence between the two bands using the features identified in the previous section. The transformation of the second target band to the first source band is based on an affine transformation (discussed in the following section 2.4). Transforming the target features to the source coordinate system and measuring the number of correspondences between features is the method of evaluating a transformation. In this registration process feature correspondence is determined from the Euclidian distance between a source feature and a transformed feature, if the distance is below a threshold distance then these features are considered corresponding features. A successful correspondence results from a transformation that produces a sufficient fraction of correspondences between the point sets.

Transformations are calculated by selecting random sets of points between the source and target images. It is possible that these selections can be made as random sets of three points from the target and source image. However a smarter method of point selection can be used to drastically improve the performance of the resulting transformations and reduce the number of calculations. Some smart methods include selecting non-collinear point sets, points that are practically in a straight line. An acceptable point set can be determined by drawing a line between the first two points and measuring the perpendicular distance from this line to each other point using vector geometry ($d_{\square}$ in figure 2.3-1). Some other checks can be performed such as verifying that the distance between the points is not below some distance threshold and the angle between p_2 and p_3 from p_1 is above some threshold (angle $\square$ in figure 2.3-1). All of these verifications ensure that the points selected to be the foundation for the transformation are spread out and not tightly clustered to a local region or linear section. A poor point selection would result in a poor transformation, even though the correspondences may be very high this will result in an unrealistic transformation.

$$\mathbf{p}_1 = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

$$\mathbf{p}_2 = \begin{bmatrix} x_2 \\ y_2 \end{bmatrix}$$

$$\mathbf{p}_3 = \begin{bmatrix} x_3 \\ y_3 \end{bmatrix}$$

$$v_1 = p_2 - p_1$$

$$v_2 = p_3 - p_1$$

$$\alpha = \cos^{-1} \frac{v_1 \cdot v_2}{|v_1||v_2|}$$

$$d_\perp = |v_1| \sin \alpha$$

Figure 2.3-1

Another note on feature correspondence is that after several iterations if the fraction of corresponding features is not met then relaxing the correspondence distance threshold helps ensure convergence. Also after good transformations are found and the number of feature correspondences is large those corresponding features can be fed into the transformation calculation with more than just three features to help find the best average transformation for all of the points which will be less susceptible to noise and local distortion errors.

The pseudocode for finding the optimal feature correspondence can be summarized as the following.

```
While the number of corresponding features is less than the number of features
in the smaller set * correspondence threshold [default 0.50]
       Get random set 1 of non-collinear points from set 1
       Get random set 2 of non-collinear points from set 2
       Calculate transformation matrix from random set 1 and 2
       Transform set 2
       Calculate the number of feature correspondences between set 1 and the
              transformed set
       If number of feature correspondences is a new maximum then
              Recalculate transformation matrix from corresponding feature
              points
              If number of feature correspondences is a new maximum then
                     Save this transform as the best transformation
              End If
       End If
    If the number of iterations mod relaxation count [default 5,000] then
        Increase the distance threshold for determining feature correspondence
    End If
End While
```

When a transformation works well it is useful to plot the features in a graph to evaluate the results. An example is shown in figure 2.3-2 the results of using feature correspondence with the Lena image. This transformation also can be viewed by overlaying the source and transformed target bands on top of each other shown in figure 2.3-3.

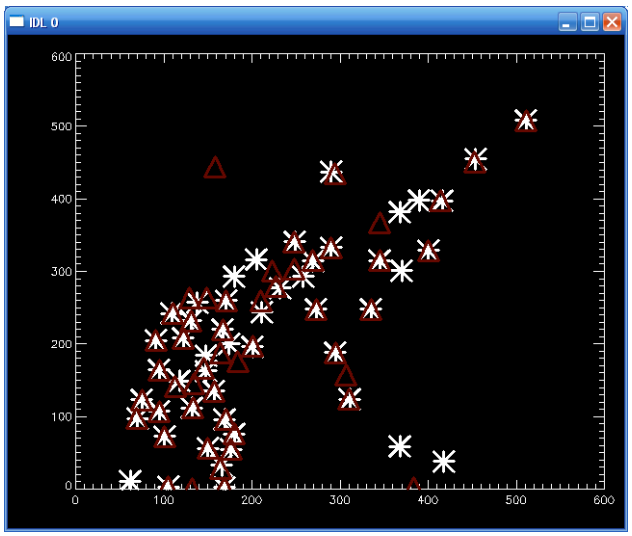


Figure 2.3-2

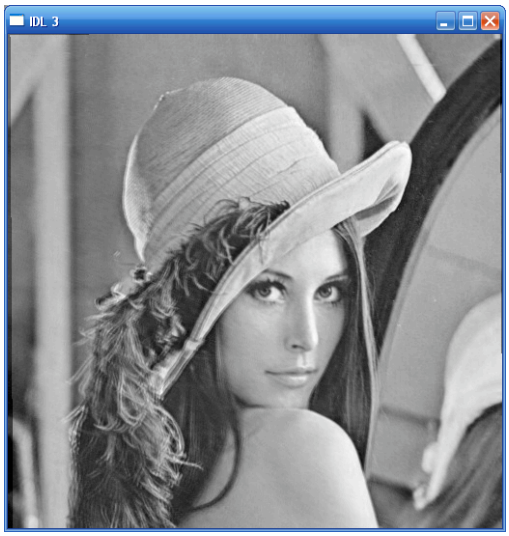


Figure 2.3-3

Also note that if insufficient constraints are placed on the feature correspondence then the resulting transform can have significant problems, as can be seen in figure 2.3-4. The feature correspondence threshold should be sufficiently large that a good portion of the features should match well but not be prohibitively large to make the computation time impractical. As mentioned in the feature detection stage, the number of features in each image is the primary limiting factor in this form of feature correspondence. This limitation occurs because as the number of points go up the feature correspondence becomes less likely when drawing random features to match. The converse to this is that if the two images are of different modalities and have significantly different phenomenology then if too few features are identified then they may not correspond well between images.

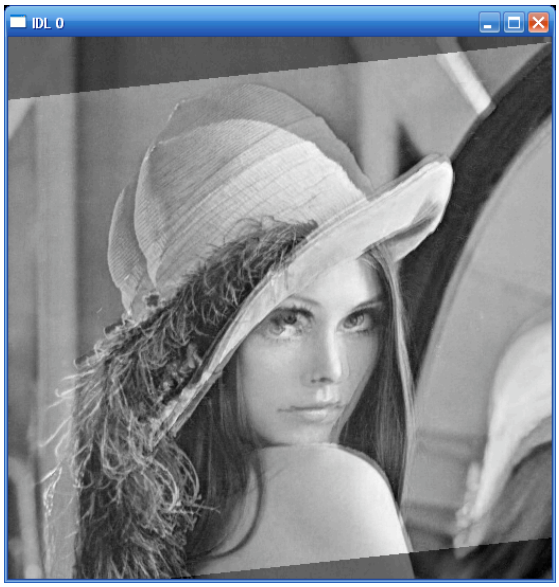


Figure 2.3-4

2.4 Affine Transforms

An affine transform is a geometrical transform used to map one image's coordinates to the coordinates of a second image. The affine transform takes into account any rotation, scale, and translation differences that can occur between the two un-registered bands. This type of transformation

was chosen for this process because there are fewer degrees of freedom than other transforms (i.e. perspective transforms) and it accounts for the type of discrepancies that can be present in aerial imagery.

To transform a coordinate from one band [x, y] into another band [x′, y′], the original coordinates are simply multiplied by the affine transform matrix that takes into account all the possible motions: translation, scale, and rotation (2.4-1).

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x \cos(\theta) & -s_x \sin(\theta) & t_x \\ s_y \sin(\theta) & s_y \cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.4.1}$$

When performing this type of transformation to imagery it is unlikely that a transformed coordinate will directly correspond to a whole pixel from the original band. To account for this likelihood, a form of interpolation is required when a transformed coordinate falls between multiple pixels. For this particular application, bilinear interpolation was used.

2.5 Mutual Information

Mutual information is a common metric to evaluate the quality of registration between two images. Mutual information is based on the probability of two intensity values occurring at the same registered point in the two images, shown in equation provided in figure 2.5-1. This metric can also be summarized over an entire image to give the overall image registration metric, which is the equation provided in figure 2.5-2.

$$MI(A,B) = P(A,B)\log\frac{P(A,B)}{P(A)P(B)}$$

Figure 2.5-1

$$MI = \sum_A \sum_B P(A,B)\log\frac{P(A,B)}{P(A)P(B)}$$

Figure 2.5-2

The general concept of mutual information is that it provides a statistical measure of how two events are related. For this particular application, the event is the occurrence of a particular set of intensity values in a registered pair of images. Experimentally it has been shown that mutual information is maximized when two images are well registered. The mutual information will gradually fall off as images become mis-registered. This phenomenon is displayed in figure 2.5-3 where the Lena image is misregistered via shifts along the x- and y-axis. The mutual information is plotted in an image with the zero shifts at the center, and the values along the x- and y- axis corresponding to the shifts in those corresponding axes.

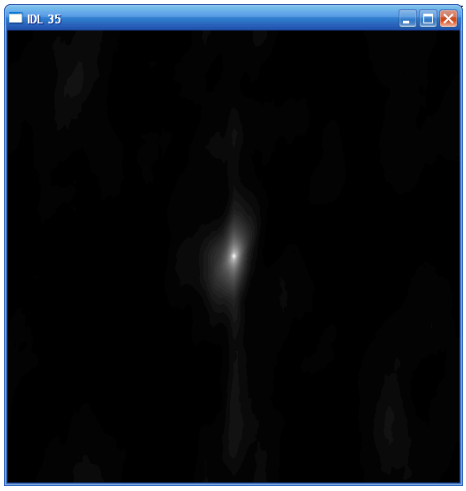
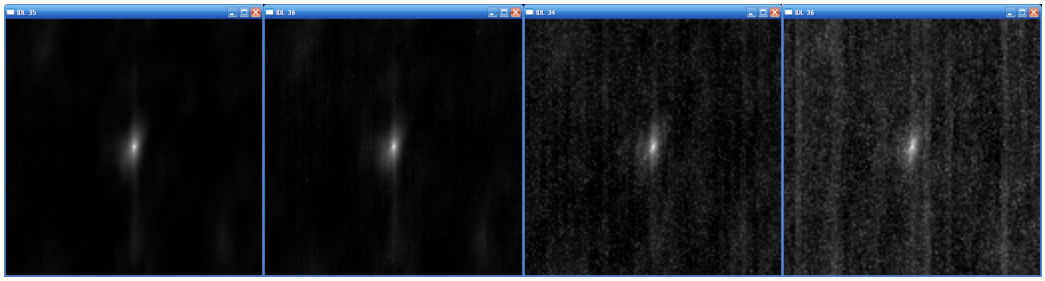


Figure 2.5-3

Because this calculation can be very computationally expensive two key approximations can be made to significantly improve performance while maintaining the desired characteristics of mutual information: intensity quantization and spatial sampling. Because the number of intensity levels and range in an image is arbitrary, it can be difficult to determine what intensity levels to use when calculating mutual information in each image. For this application the top and bottom 5% of the histogram was trimmed off of the image and then rescaled into just 10 levels. This approximation produced satisfactory and consistent results and sped up the calculation significantly. The second approximation, the spatial subsampling, was empirically measured. It was found that there was a small, but not prohibitive, degradation in the fidelity when sampling 100% of the corresponding pixel values vs. just sampling 50%. The results of this approximation can be visualized by mapping the mutual information for the Lena red and blue bands shifted relative to one another, but with varying sampling rates shown next to one another in 2.5-4.



100% **50%** **10%** **5%**

Figure 2.5-4

2.6 Particle Swarm Optimization

Particle Swarm Optimization, or PSO, is a genetic algorithm introduced by Eberhart and Kennedy in 1995[2]. The algorithm itself was inspired by the social behavior of birds. It is useful to imagine a flock of birds circling above some location. Ultimately, the goal of the flock is to find food somewhere on the ground below. If a single bird discovers food and dives to the ground, the entire flock will shift whatever course they were previously pursuing, and follow the bird diving to the food.

By extension, this behavior can be applied to solution optimization. In PSO, a given solution to a function is represented by a particle (analogous to a bird). A group of possible solutions, or particles, is referred to as a swarm (analogous to a flock). Each particle is defined by a set of parameters. These parameters are initially unique to a particle. For example, if one were trying to find the optimum solution

to fit a straight line to a set of points, the two parameters to consider would be the slope and intercept. Using PSO, each particle would be initialized with two parameters, a guess at slope and intercept values.

A swarm is composed of N solutions that each have j-number of parameters. Each ith particle, x_i, (i=1, …, N) is initialized with random parameter values, bounded by each parameter condition. The quality of each particle's offered solution is judged using a fitness function. At each iteration, or generation, all the solutions are changed using a velocity vector. The velocity vector is a factor of several different variables and is shown below (2.6-1).

$$v_i(k+1) = \phi(k)v_i(k) + \alpha_1\left[\gamma_{1i}\left(p_i - x_i(k)\right)\right] + \alpha_2\left[\gamma_{2i}\left(G - x_i(k)\right)\right] \tag{2.6-1}$$

The two most important features affecting the velocity vector for each generation (k) are each particle's personal best solution (p_i) and the global best solution (G). A particle's best solution is the set of parameters that from that particle's history have produced the best fitness function value. The global best solution is the set of parameters that produce the best fitness function value obtained so far, out of the entire population. The velocity vector for each particle, in each generation, is calculated using these two values. The incorporation of these two 'best' values will push all of the particles in the swarm to follow the one solution getting the best answer. Thus, all the particles are following the best solution; just like the birds, following the one bird that's located the food. After an appropriate amount of time, all of the particles will converge to the same solution.

Equation (2.6-1) also includes the previous velocity of the ith particle (v_i), the position of the ith particle (x), random numbers between [0, 1] ($\gamma_{1,2}$), and the acceleration constants ($\alpha_{1,2}$, usually set to 2.0). The inertial weight of the particle is designated by ϕ(k) and is calculated using the following equation (2.6-2). The total number of generations is represented by K, the minimum allowable inertial weight by ϕ_{min}, and the maximum allowable inertial weight by ϕ_{max}.

$$\phi(k+1) = \phi(k) + \left(\frac{\phi_{\min} - \phi_{\max}}{K}\right) \tag{2.6-2}$$

The position of each particle for the next generation is calculated using equation (2.6-3).

$$x_i(k+1) = x_i(k) + v_i(k+1) \tag{2.6-3}$$

For this particular application, each solution will be made up of five parameters, the variables of the affine transform: t_x, t_y, s_x, s_y, and θ. The pseudo-code for achieving this particular application is shown below[3,4].

```
WHILE maximum iterations or minimum error criteria is not attained

      FOR each particle
            Initialize particle [Initialize N number of particles with random
            values for all RTS parameters within designated range]
      END

      FOR each particle

            Calculate fitness value [Using initial RTS values, perform N
            transforms and calculated N mutual information values (fitness values)]


            IF the fitness value is better than the best fitness value
            (pBest) in history
```

```
                set current value as the new pBest [If generated MI value is
                better than any recorded MI value for that particle, save those
                RTS vales as the particle's best solution (pbest)
        ENDIF
    ENDFOR

    Choose the particle with the best fitness value of all the particles
    as the gBest [Choose the particle whose RTS values produced the highest MI
    value]

    FOR each particle
        Calculate particle velocity according equation (a)
        Update particle position according equation (b)
    ENDFOR

ENDWHILE
```

2.7 Levenberg-Marquardt Optimization

Levenberg-Marquardt's method of nonlinear optimization can be applied to iteratively optimize the parameters (denoted as p_j or as the collective matrix $\boldsymbol{p}$ for all parameters) of an affine transformation matrix (shown in figure 2.7-7) with the goal of maximizing the overall mutual information of the registered images. The key notion here is that at each iteration the mutual information partial derivative can be calculated for each parameter at each registered pixel, shown in figure 2.7-1. When these are combined into an array with the partial derivative for each pixel along the rows and each parameter along the columns the Jacobian matrix is produced, shown in figure 2.7-2. The Jacobian matrix is a good indication of how the registration changes at each point for each parameter.

$$\frac{\partial MI(x_i)}{\partial p_j}$$

Figure 2.7-1

$$\mathbf{J} = \begin{bmatrix} \frac{\partial MI(x_1)}{\partial p_1} & \cdots & \frac{\partial MI(x_1)}{\partial p_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial MI(x_m)}{\partial p_1} & \cdots & \frac{\partial MI(x_m)}{\partial p_n} \end{bmatrix}$$

Figure 2.7-2

When the Jacobian matrix is transposed and matrix multiplied by itself the resulting matrix is referred to as the Hessian matrix. The diagonal elements can be multiplied by the one plus the Levenberg-Marquardt parameter (□) to give us the augmented Hessian, shown in figure 2.7-3 (the # symbol is used to indicate element by element multiplication). This matrix is commonly used to solve for the change in parameters needed to solve for the optimal parameter set. This calculation is done by first computing the gradient matrix(**G** shown in figure 2.7-4), which is the error at each pixel (or the difference between the desired mutual information and the current mutual information given the current parameter set shown in figure 2.7-5) and matrix multiplying the transpose of the Jacobian by the error at each pixel to get the gradient. To produce the proposed change in the parameters the pseudo inverse of the augmented Hessian is matrix multiplied by the gradient matrix. These steps are shown in figure 2.7-4.

$$\mathbf{H}' = \mathbf{J}^T\mathbf{J}\#(1+\lambda\mathbf{I})$$

Figure 2.7-3

$$\mathbf{G} = (\mathbf{J}^T\varepsilon)$$

Figure 2.7-4

$$\mathbf{å} = \begin{bmatrix} MI_G - MI(x_1) \\ \vdots \\ MI_G - MI(x_m) \end{bmatrix}$$

Figure 2.7-5

$$\Delta\mathbf{p} = \mathbf{H}'^{-1}(\mathbf{J}^T\mathbf{å})$$

Figure 2.7-6

This process is a local approximation method that follows the derivatives to reduce the error, or improve the mutual information in this case, relative to its current position. After a change in parameters, dp, is calculated and constrained to reasonable step sizes. The mutual information can be calculated for the transformation $\boldsymbol{p}+\square\boldsymbol{p}$. If the resulting mutual information is better than the change in parameters is accepted and the Levenberg-Marquardt parameter, lambda, is decreased (i.e. by a factor of 10). Otherwise the Levenberg-Marquardt parameter, lambda, in increased (i.e. by a factor of 10).

These steps are repeated until the solution converges. Cconvergence is declared when 10 successive iterations fail to improve mutual information. However, a very important modification to the traditional Levenberg-Marquardt process is to coarsely evaluate changes to the individual parameters to make certain the solution is not a local optima point. By adding in this step the search for solutions becomes much more robust and less prone to premature cessation. If an improvement to mutual information can be found through the coarse search then the new parameters are accepted and the Levenberg-Marquardt processing is restarted.

Some implementation specific details were crucial to ensuring successful convergence on good solutions. The first is establishing the step size for each parameter in the search space. This step size must be appropriate for the position of the parameter is the affine transformation matrix: p_1, p_2, p_3, p_4, p_5, and p_6 as in figure 2.7-4, with the step sizes used in this registration process of 0.01 for p_1, p_2, p_4, and p_5 and step sizes of 0.5 for p_3 and p_6. The step size is used in the discrete calculation of the partial derivatives as well as in the coarse searching. A poorly chosen large step size can result in erroneous derivatives for the particular local conditions. Also too small a step size may not result in significant improvements to mutual information and may require more iterations to converge than are necessary.

$$\begin{bmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \end{bmatrix}$$

Figure 2.7-7

Also because the optimum overall mutual information and the pixel-by-pixel mutual information are not known an approximation must be made. The method selected for this registration technique has been to multiply the maximum pixel-by-pixel mutual information by a scalar greater than or equal to one and subtracting the current parameter's pixel-by-pixel mutual information, shown in figure 2.7-5. This operation provides a simple error function ($\mathbf{å}$ defined in figure 2.7-5) that allows us to solve for $d\boldsymbol{p}$ (as shown in 2.7-6).

$MI_G = s\ MAX(MI)$

Figure 2.7-8

Although good results were found when the initial parameters were within a small range from the correct solution the iterative process is very computationally expensive. Methods of improvement may be to perform LM at coarser scales of resolution and work up to higher resolutions, finding alternative registration metrics or speeding up the mutual information calculation to increase the computation speed, investigating dynamically sizing the steps, and smart ways to coarsely search. To reduce the risk of diverging and getting away from the optimal region the step sizes were chosen to be very small and the coarse searching was performed with 10 steps in the positive and negative direction for each parameter when the LM method had converged.

3.0 Process

The goal of this software package is to allow the user to input two un-registered, noisy bands and produce noise-less, registered versions of the same bands, with no knowledge of scene content or acquisition parameters. Initially the two raw bands are passed through a bilateral filter to reduce noise and enhance edges. The user has an option to only perform a traditional Gaussian low-pass filtering on the bands, but there would be a substantial loss in edge quality. The two resulting, noise-reduced bands are then passed through a Harris corner detection process. The resulting corner lists from both bands are referred to as feature lists. These two lists of features are then compared using a feature correspondence algorithm to determine which points are corresponding to the same image content. After a certain percentage of features have been matched, a rough guess affine matrix is calculated. The rough guess affine matrix represents the best guess at the affine transformation required to transform the second band into the first band's coordinate system, based on the matched feature points. The quality of this initial transform is gauged using the mutual information between the original first band and the mapped second band. Next, the two noise-reduced bands and the rough guess matrix are then passed to two methods of optimization; Particle Swarm Optimization and Levenberg-Marquardt Optimization. Both of these optimizations will produce an optimized version of the affine transform and apply them to the two noise-reduced bands. Once again, the quality of each resulting transformed is measured using the mutual information between the two resulting bands. The algorithm flow is depicted in figure 3.0-1 below.

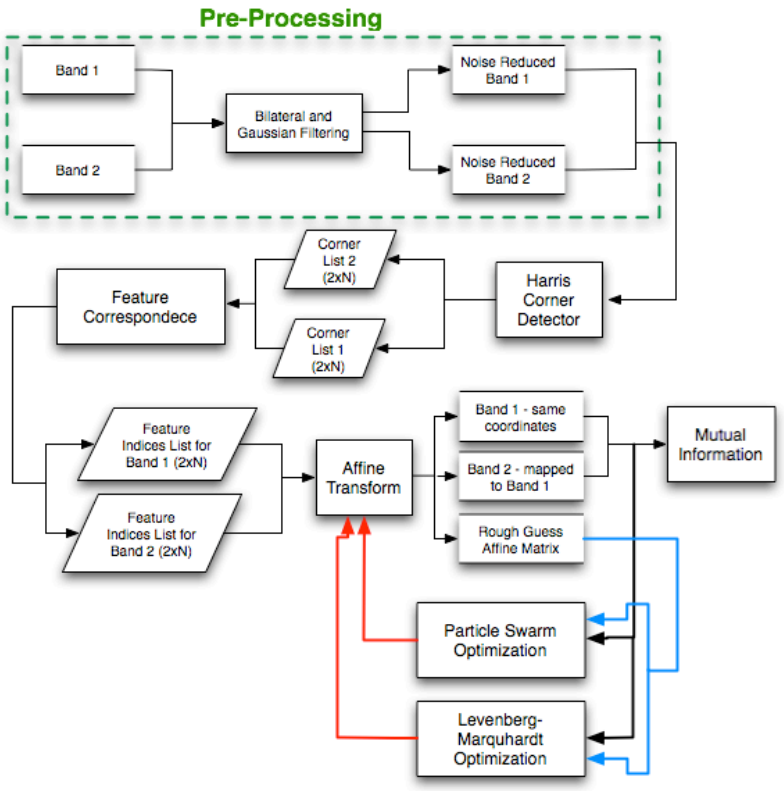


Figure 3.0-1: Software flow chart

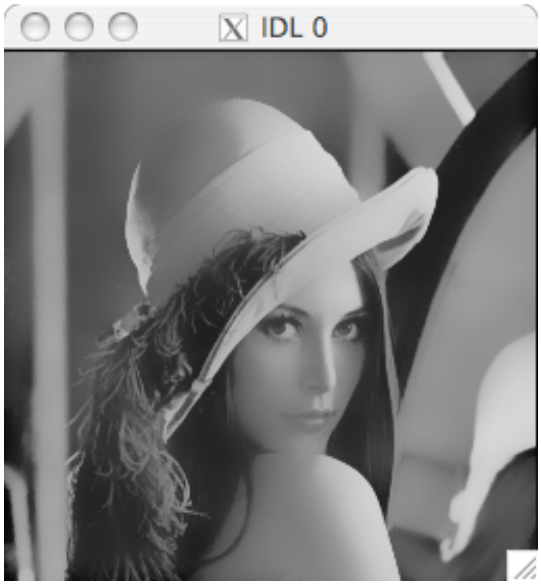
4.0 Results

4.1 Bilateral filtering

Preliminary testing was performed using a bilateral filtering implementation in IDL. These tests were conducted to examine the effect of bilateral filtering under various conditions with different image scenes. The first test implemented used a controlled example to determine the validity of the implemented algorithm. A simulated edge, with the addition of uniformly distributed noise, was passed through the bilateral filter and the result displayed. Both the original and resulting images are shown below in figures 4.1-1 and 4.1-2.

Figures 4.1-1and 4.1-2: Simulated edge with uniformly distributed noise (on left) and bilateral filtering result (on right)

From the results, one can see that bilateral filtering clearly removes the noise and preserves the edge. Additionally, comparisons were made between the effect of bilateral filtering and the effect of traditional Gaussian low-pass filtering. Both types of filters were processed with two different sigma values. For the bilateral case, the supplied sigma value was used for both the range and domain filtering.



Figures 4.1-3 and 4.1-4: Results of bilateral filtering using a sigma of 3 pixels (on left) and 6 pixels (on right)

Figures 4.1-5 and 4.1-6: Results of Gaussian low-pas filtering using a sigma of 3 pixels (on left) and 6 pixels (on right)

Through comparison, it can be seen that bilateral filtering surpasses traditional Gaussian low-pass filtering with its ability to remove noise and preserve edges. Based on these results, bilateral filtering with a three-sigma window size was used in preprocessing.

4.2 Corner detection

Testing of the Harris corner detection was performed using each of the 4 defined metrics section 2.2. Emperical results showed the Harris metric worked very well and the default settings of FILL IN DEFAULTS. For the real WASP data the found corners are plotted for the two bands and displayed in figure 4.2-1, the synthetic WASP data corners plotted in figure 4.2-2, and the Lena image was used in section 2.2 and the results displayed in that previous section. Examples of what the raw corner responses are shown for the real WASP data in figure 4.2-3 and the synthetic WASP data in figure 4.2-4, both with a square root stretch on the positive values and the values below zero truncated to zero.

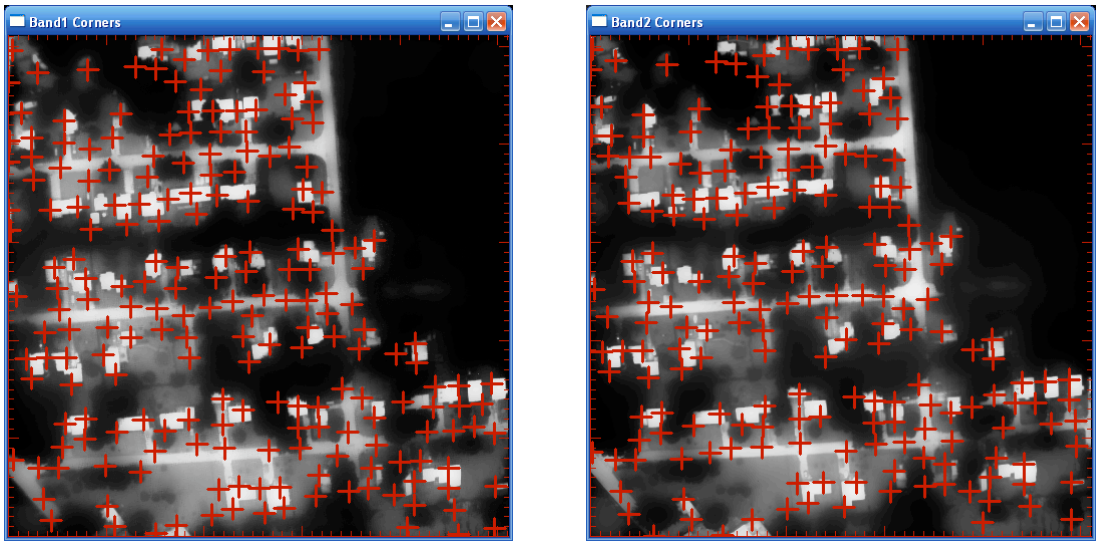


Figure 4.2-1

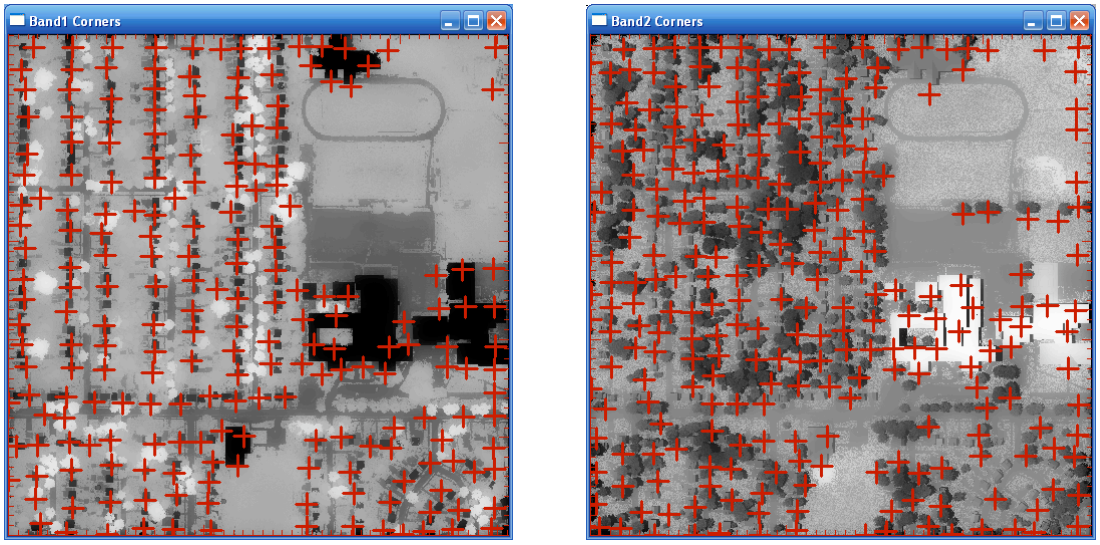


Figure 4.2-2

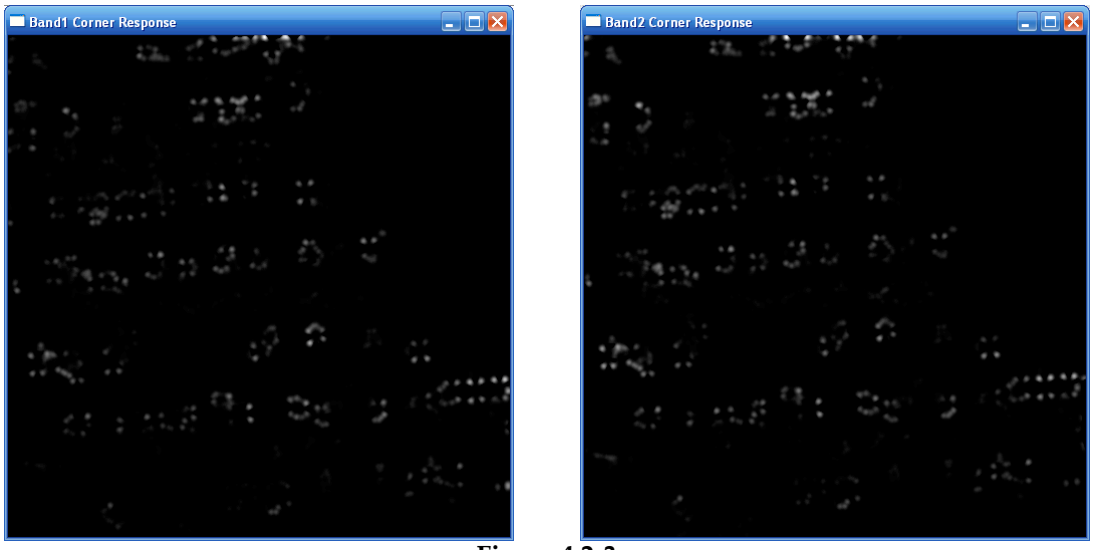


Figure 4.2-3

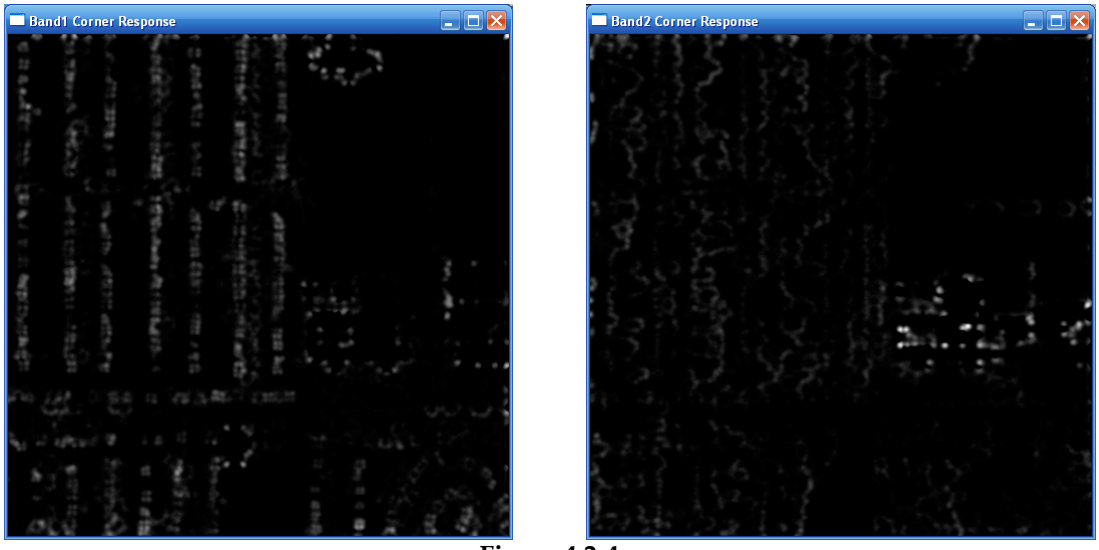


Figure 4.2-4

The corner response metric does a very good job of identifying the corners in all of the images, however the number of corners can become very large becoming prohibitive to speed and when the modalities are very different the maximum corners may not correspond very well.

4.3 Feature correspondence

In order to test feature correspondence sets of random points were generated, and a random selection was selected and copied to be the second set with random perturbations in the coordinates. Then feature correspondence was performed to determine what level of correspondence to achieve good results. Without stringent non-collinearity requirements just requiring 50% of the features to correspond resulted in consistently poor transformations, as can be seen in figure 4.3-1 and 4.3-2.

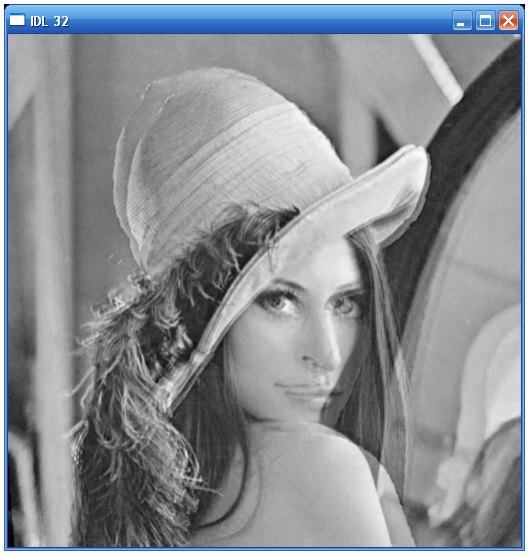


Figure 4.3-1

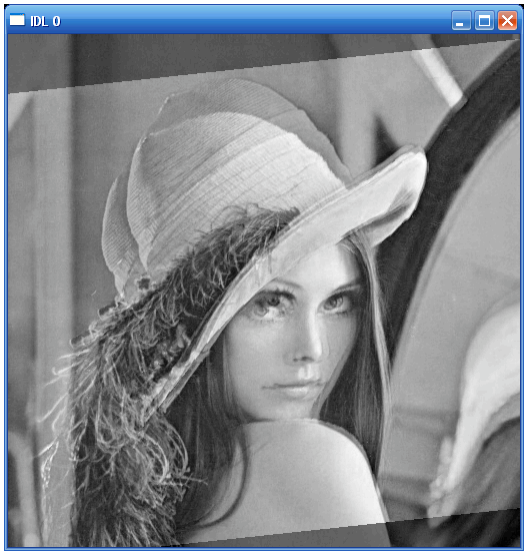


Figure 4.3-2

However after the non-collinear requirements and angular requirements a 50% correspondence performed significantly better and determined to be sufficient to feed to the optimization algorithms and was fast enough to be appropriate for this registration process. Images of 50% correspondence with the additional point set requirements showing good registrations are show in figure 4.3-3.

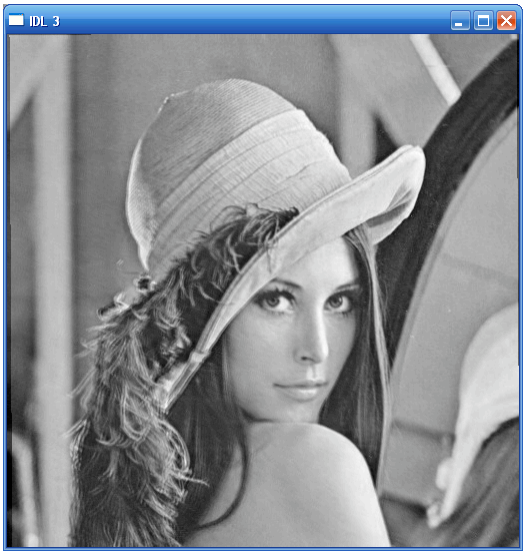


Figure 4.3-3

4.4 Mutual Information

The mutual information map used to optimize the affine transformation for the Lena image is displayed in figure 4.4-1 with the full range of intensity values and figure 4.4-2 with just 10 levels of intensity. From this analysis the fewer levels provided a smoother MI map to optimize and 10 levels was chosen for this registration process. The MI maps for the real and synthetic WASP images are shown in figures 4.4-3 and 4.4-4 respectively.

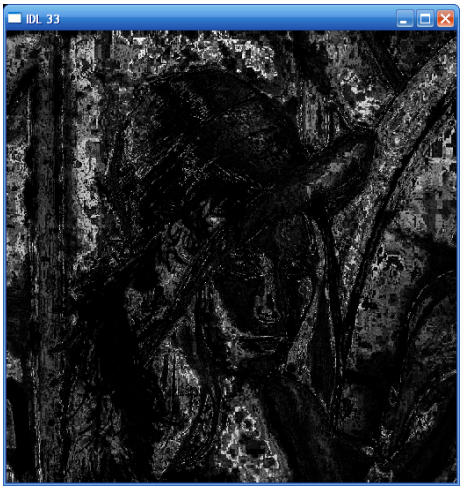


Figure 4.4-1

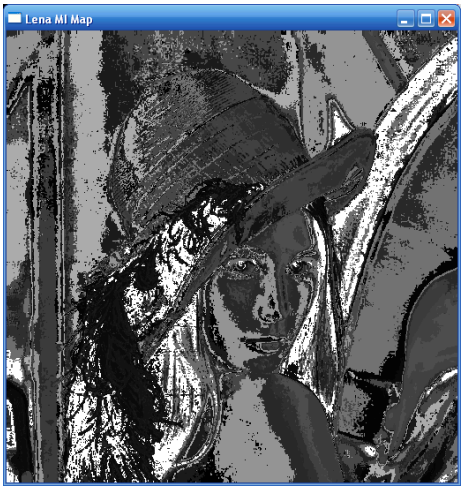


Figure 4.4-2

Figure 4.4-3

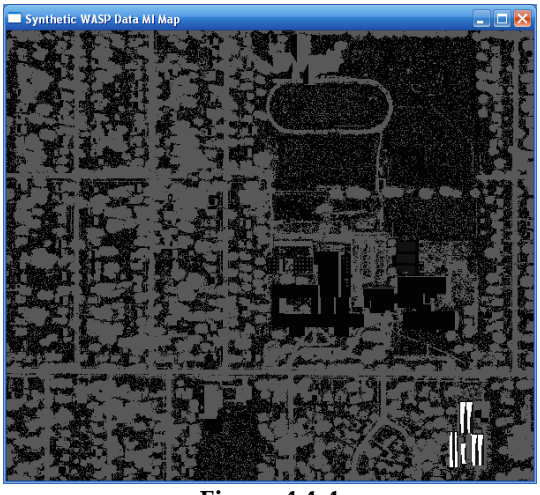


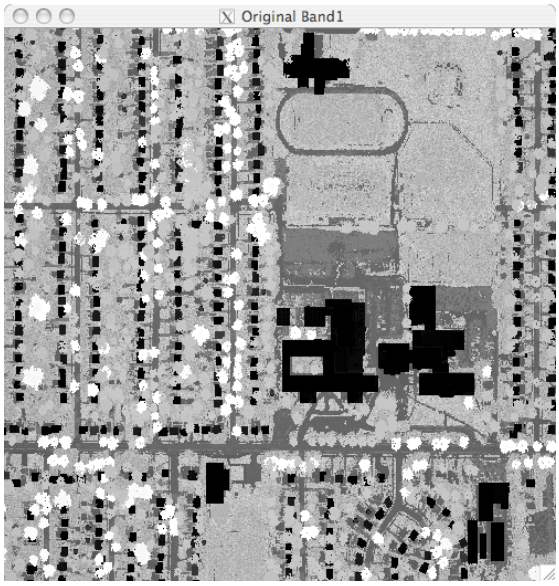
Figure 4.4-4

4.5 Particle Swarm Optimization

The particle swarm optimization (PSO) routine was first tested using controlled parameters before integration into the entire software package. Using synthetic imagery bands, manual translations, scales, and rotations were applied to a band of the synthetic imagery. Based on this manual geometric transform, a rough guess matrix was created that offered a solution close to the correct transform. Employing the generated guess matrix and the PSO parameters, 9 solutions were initialized using varying boundaries and allowed to run for 30 generations. The boundary limits of each of the parameters were varied to determine the effect the size of the solution space had on the algorithm's ability to converge to the correct answer. The input conditions for test 1 are shown below in table 4.5-1.

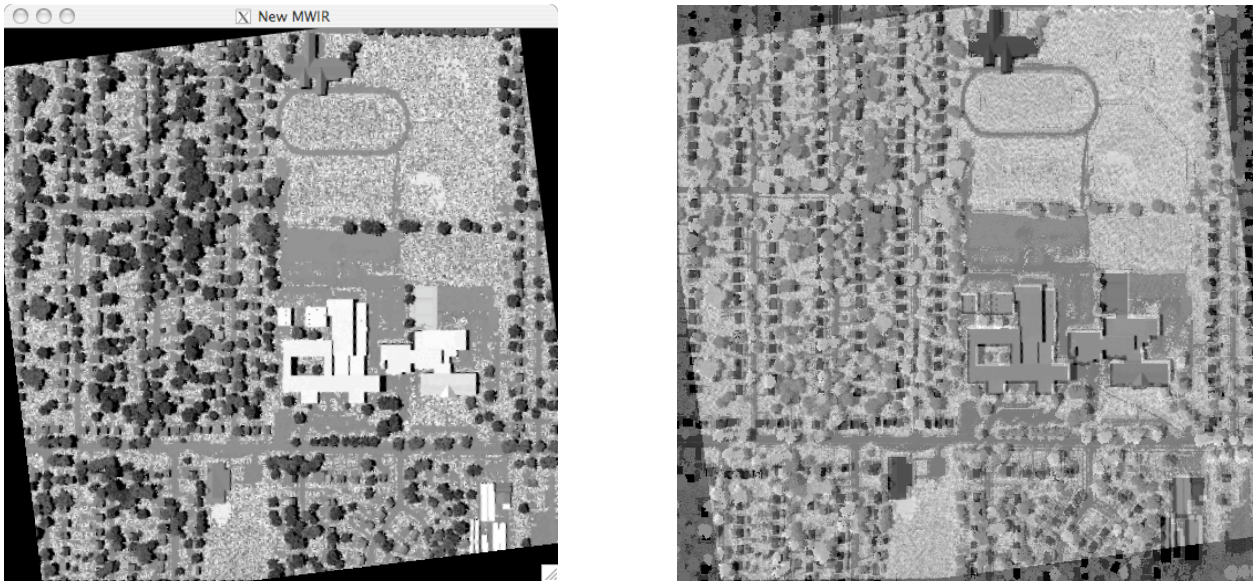
	Actual	Guess	Minimum	Maximum
Tx	5.0	-2.0	-1.0	-8.0
Ty	5.0	-3.0	-1.0	-8.0
Sx	1.0	1.0	1.0	1.0
Sy	1.0	1.0	1.0	1.0
θ	7.0	5.5	0.0	-9.0

Table 4.5-1: Parameters for PSO test 1



Figures 4.5-2 and 4.5-3: Original band 1 (on left) and original transformed band 2 (on right)

Based on the actual manual transform applied to band 2 prior to optimization, PSO should calculated the affine parameters or the restoration matrix to be the following: t_x = -5.0, t_y = -5.0, s_x = 1.0, s_y = 1.0, and θ = -7.0. The optimization found the affine parameters for the given two bands to be the following: t_x = -4.667, t_y = -4.667, s_x = 1.0, s_y= 1.0, and θ = -7.000007. Below in figures 4.5-4 and 4.5-5 are the resulting band 2 after the applied affine transform and the overlay of the resulting band 2 on the original band 1.



Figures 4.5-4 and 4.5-5: Resulting band 2 with found transform matrix applied (on left) and resulting band 2 overlaid on original band 1

From the matrix parameters and visual inspection PSO performed perfectly. The solution was found in 13.64 minutes. Although this run time seems short when compared to other more traditional optimization routines, it could have been shorter. As displayed by the chart in figure 4.5-6, the global best solution's mutual information had reached a plateau after approximately seven generations. Due to inadequate stopping conditions, the solution search was not abandoned after it had stopped changing.

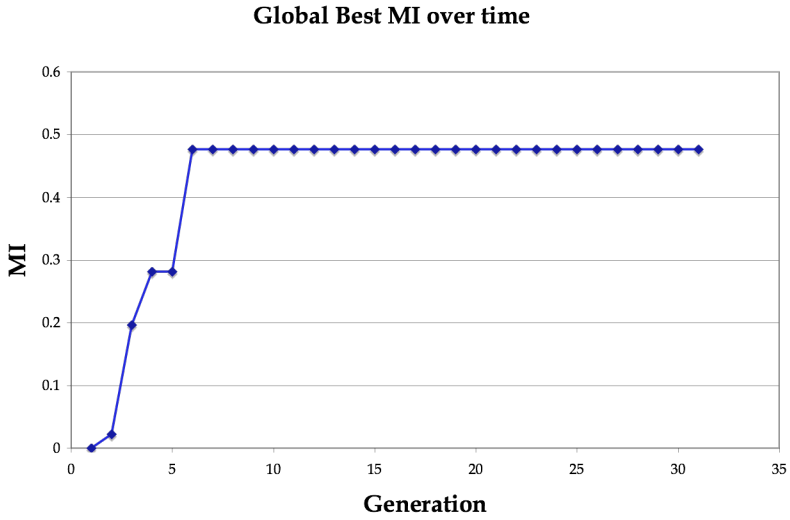


Figure 4.5-6: Global best solution's mutual information for test 1, tracked over time

However, this test was very controlled and had a limited solution space. A second and third test was performed where the solutions space was both increased and decreased. The supplied parameters for both these tests are shown below in table 2.

	Test 2				Test 3			
	Actual	**Guess**	**Minimum**	**Maximum**	**Actual**	**Guess**	**Minimum**	**Maximum**
Tx	5.0	-2.0	0.0	-14.0	5.0	-2.0	-4.0	-6.0
Ty	5.0	-2.0	0.0	-14.0	5.0	-2.0	-4.0	-6.0
Sx	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Sy	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
θ	7.0	5.5	0.0	-20.0	7.0	5.5	0.0	-6.0

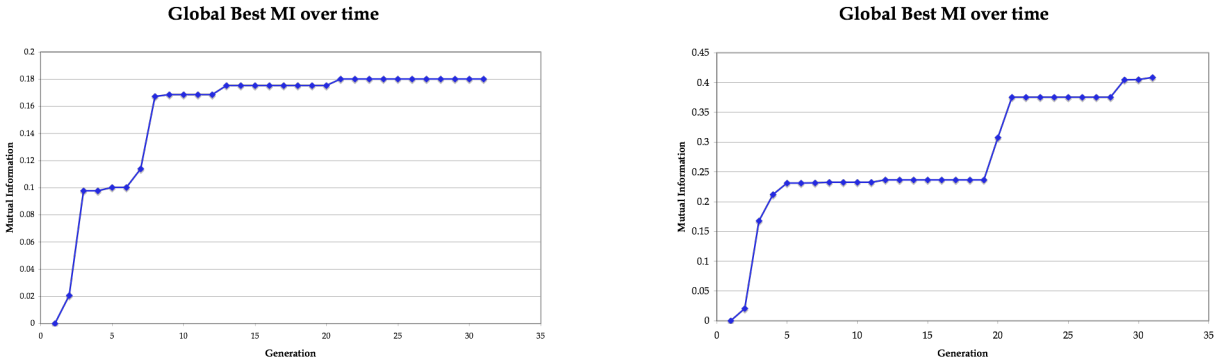
Table 4.5-2: Parameters for PSO test 2 and 3

The results from both these tests are shown below in table 4.5-3.

	Perfect	**Test 2**	**Test 3**
Tx	-5.0	-6.8833	-4.9778
Ty	-5.0	-6.8833	-5.5111
Sx	1.0	1.0	1.0
Sy	1.0	1.0	1.0
θ	-7.0	-7.4999	-6.9335

Table 4.5-3: Results from PSO test 2 and 3

Furthermore, the plots tracking the global best solution's mutual information for both test 2 and test 3 are shown below in figures 4.5-7 and 4.5-8.



Figures 4.5-7 and 4.5-8: Global best MI over time for test 2 (on left) and test 3 (on right)

From both the results in table 4.5-3 and the two graphs shown above it can be seen that the third test did produce better results than second test, but was not as accurate as the initial test. This result is an interesting outcome due to the higher level of constraint placed on the third test when compared to the first. One would have expected the smaller search space to produce a more accurate answer, quicker. On the contrary, the third test produced less accurate tests and required more generations to achieve those results. However, from inspection of both figures 4.5-7 and 4.5-8, it can be seen that the global best solution's mutual information tends to plateau for several generations and then increase. It is possible, that if the optimization was initialized with more solutions or allowed to exist for more generations, the second test could have produced results that are more accurate than the first test. However, that does not provide an obvious explanation for the speed at which the smaller solution space of the first test was able to converge on the correct answer.

A fourth test was designed and implemented to test PSO performance when given two bands that are already perfectly registered. Two bands were passed into the PSO algorithm without any prior translation or rotation manipulation. The same rough guess matrix used for the three previous tests was also supplied as well as the solution space boundaries of the third test. The resulting affine transform

parameters from the fourth test are shown below in table 4.5-4.

	Perfect	Test 4
Tx	0.0	-2.0
Ty	0.0	-3.0
Sx	1.0	1.0
Sy	1.0	1.0
θ	0.0	-0.08

Table 4.5-4: PSO results for test 4

PSO clearly failed when supplied with two perfectly registered bands. This failure is emphasized by figure 4.5-9, which displays the global best solution's mutual information over time in test 4.

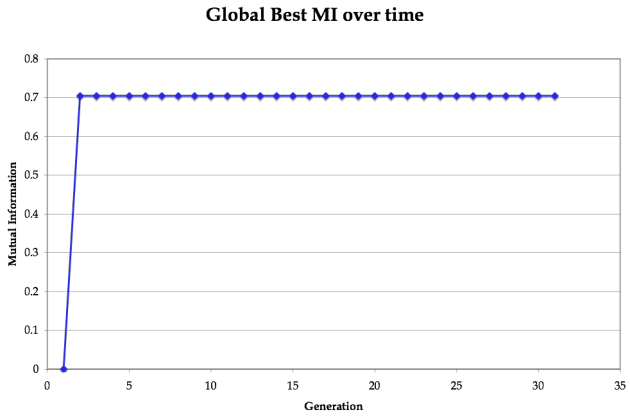


Figure 4.5-9: Global best MI over time for test 4

It can be seen that after the first iteration PSO never calculated a MI value higher than the one achieved by the rough guess matrix solution. This result does not intuitively make sense. One would assume that two bands that are considered perfectly registered without any geometrical transformation would produce a higher mutual information value when overlaid perfectly without any shifts in either band, versus a two and three pixel shift in one of the bands. It is unclear at this point why this error occurred and is an area of continued investigation.

4.6 Levenberg-Marquardt

Images of the before and after Levenberg-Marquardt optimization are shown in figure 4.6-1 with the synthetic imagery, figure 4.6-2 with the real WASP imagery, and figure 4.6-3 with the Lena red and blue bands. Each of these started with a rotation of ten degrees and translational offsets of 10 pixels along both the x- and y-axis.

Figure 4.6-1

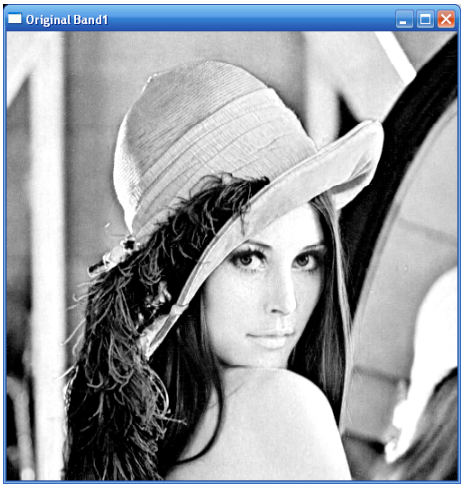
Figure 4.6-2

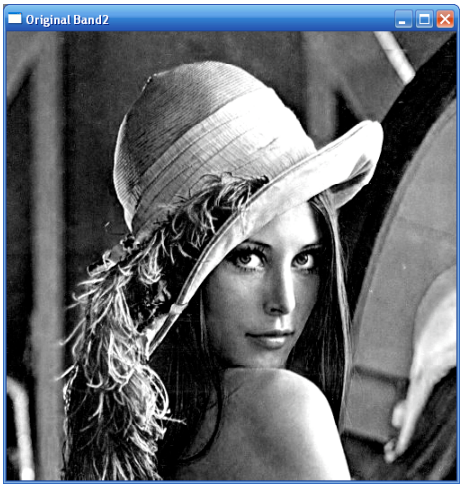
Figure 4.6-3

Each of these examples show excellent registration after LM optimization from fairly poor initial conditions. This stand alone testing verified the concept that mutual information does correspond to the quality of registration and also the Levenberg-Marquardt optimization implementation correctly follows the pathway to the optimal solution.

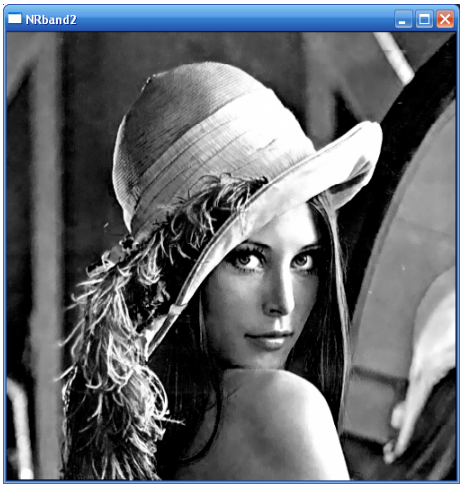
4.7 Whole process

After all initial testing was performed on each individual module all of the processes run in sequence as a single program. The same image was run through the process twice using the same pre-processing settings. The first run through the program used the Particle Swarm Optimization to produce a final affine transform matrix. The second run utilized the LM optimizer. For both situations, the algorithm was handed two perfectly registered images. The results from each step in the algorithm are shown below in figures 4.7-1 through 4.7-9.

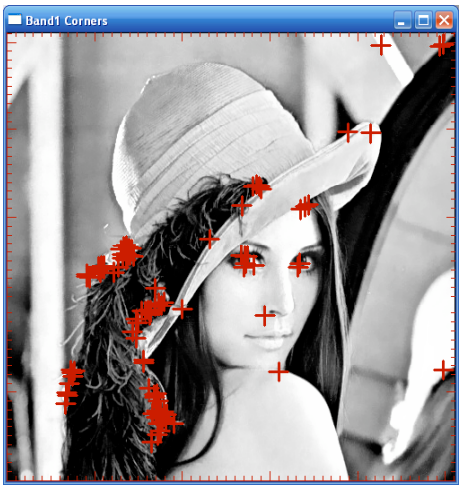


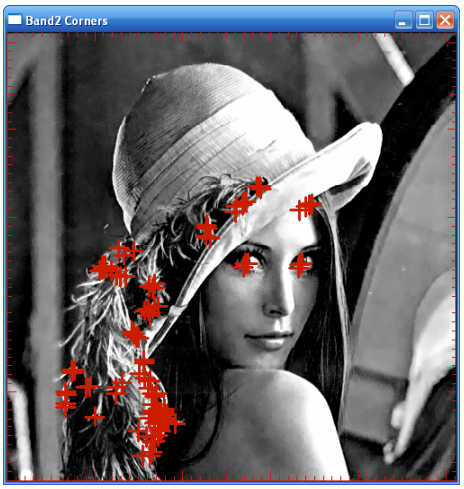


Figures 4.7-1 and 4.7-2: The two original bands



Figures 4.7-3 and 4.7-4: The two bands after noise-reducing pre-processing





Figures 4.7-4 and 4.7-5: Overlay of corners detected in each band using Harris corner detection

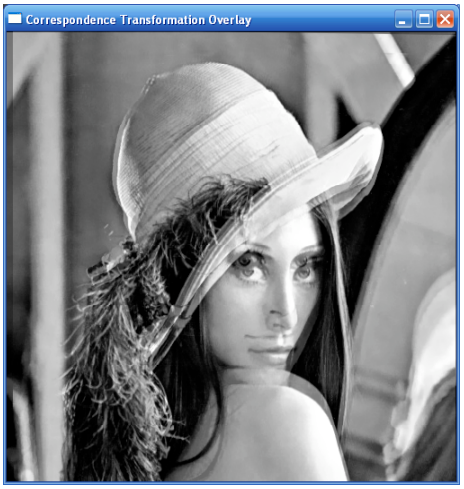
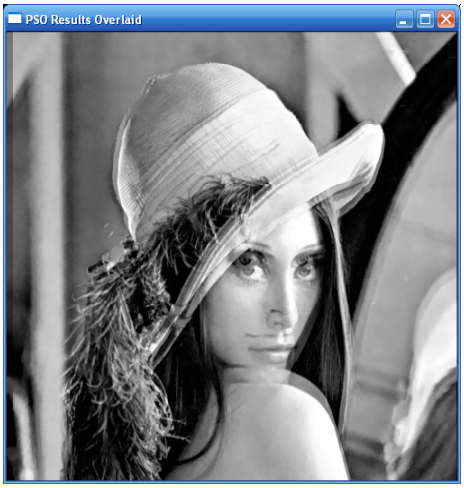
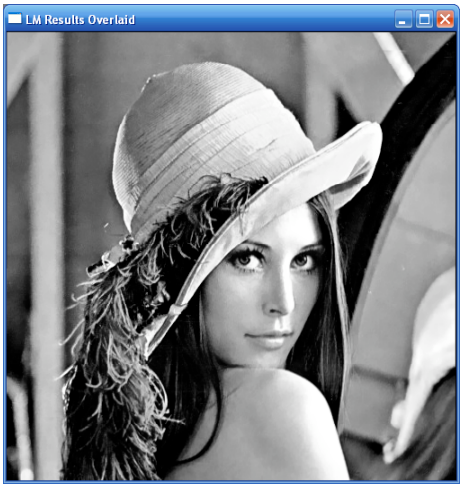


Figure 4.7-6: Using the corners found in the corner detection process, the feature correspondence algorithm calculated a rough guess affine transform matrix. The figure above shows the resulting registration using this matrix





Figures 4.7-7 and 4.7-8: Starting with the rough guess affine matrix, the set of roughly registered bands were sent through both optimization routines. PSO is on the left, LM is on the right.

From the results displayed above, it can be seen that LM outperformed PSO in this particular application. Below, in table 4.7-1 are some quantitative results from both of these tests.

	Mutual Information	Run Time
Feature Correspondence	0.2490	4.5 mins
PSO	0.2565	6.2 mins
LM	0.5248	57.7 mins

Table 4.7-1: Quantitative results from algorithm tests

Even though LM did surpass PSO in the final result, it was significantly slower. The failure of PSO for this particular test most likely is routed in its failure during stand-alone testing to register perfectly registered images. Based on the other tests ran with PSO using controlled parameters it is strongly believed that PSO is capable of performing as well as LM in significantly less time once improvements have been made.

5.0 Conclusions and Improvements

Based on all of the testing there are several areas of improvement in the proposed registration technique. While bilateral filtering can certainly be considered an appropriate and successful pre-processing technique for this application, the effect of varying the domain and range filter sigmas could be investigated.

There are several areas of improvement inside the PSO algorithm. Based on testing it can be seen that accuracy of the PSO results is heavily correlated to the input parameters. When integrated into this particular registration process, PSO is fed initial parameters based on the feature correspondence supplied affine matrix. The search space of the PSO parameters is based on this first rough guess matrix. It would be worthwhile to investigate the best method for initializing the particle parameters and defining the solution space as a whole, given this initial matrix. In addition, the velocity vector, which controls the movement of each of the particles, has several variables affecting its value that have little to no physical meaning. It would be advisable to determine a more realistic method of determining these variables that would tie their calculated value into the actual function attempting to be solved. Finally, all of the tests discussed were completed using the same number of solutions and generations. It is highly likely that given a larger swarm of particles and more generations PSO would have converged on the correct answer. It would be interesting to explore the concept of dynamically determining the swarm size and number of generations based on the particular solution be optimized.

References:

[1] C. Tomasi, R. Manduchi, "Bilateral Filtering for Gray and Color Images," *iccv*, p. 839, Sixth International Conference on Computer Vision (ICCV'98), 1998

[2] Eberhart, R. C. and Kennedy, J. "A new optimizer using particle swarm theory", Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan. pp. 39-43, 1995

[3] Qi Li and Hongbing Ji, "Medical image registration based on maximization of mutual information and particle swarm optimization", Proc. SPIE Int. Soc. Opt. Eng. **6534**, 65342M (2007)

[4] Xuan Yang *et al.*, "Maximization of Feature Potential Mutual Information in Multimodality Image Registration Using Particle Swarm Optimization", Proc. SPIE Int. Soc. Opt. Eng. **5747**, 1300 (2005) *(10 pages)*

[5] Goshtasby, A. Ardeshir, *2-D and 3-D Image Registration*. John Wiley & Sons, Inc., Hoboken, New Jersey 2005