**Report for Assignment 3 – Part II: Effect of Image Match on Registration**

**Ravikiran Janardhana**

**Medical Image Analysis – COMP 775**

**Fall 2011**

**Tasks completed**

1. Implemented a matlab program to perform intensity transformation of the given fixed image and computed the moving image by applying an ‘x translation (Δx)’ on the result.
2. By varying the ‘x translation from -128 to +128’, the different image mismatch values namely “Sum of squared intensity differences”, “Normalized Cross-Correlation”, “Euclidean differences in quantile functions” and” Mutual information” were computed for each translation.
3. Plotted the image mismatch value on y axis against the x translation (Δx) on x-axis for all possible translations (-128 to +128).
4. Wrote a registration program for each of the above image mismatch functions to compute the most optimal ‘x translation’ which yields the least image mismatch value (can be local minima) using gradient descent method.

**Result Images**

The following ***20 images (graphs 1-20)*** correspond to the below listed set of intensity transformations and image mismatch functions which will be further be referred to as ANNEXURE-A and ANNEXURE-B.

**ANNEXURE-A – INTENSITY TRANSFORMATIONS**

Transformed Intensity(x) = alpha(x) \* intensity(x) + beta + zero mean gaussian distributed pixel

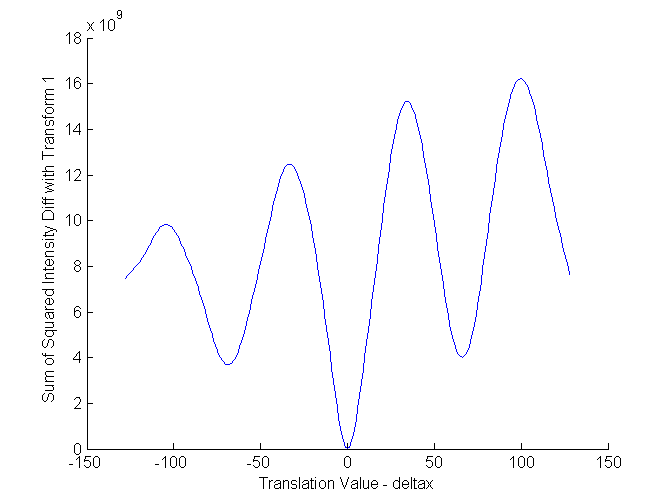
intensity noise with standard deviation 10

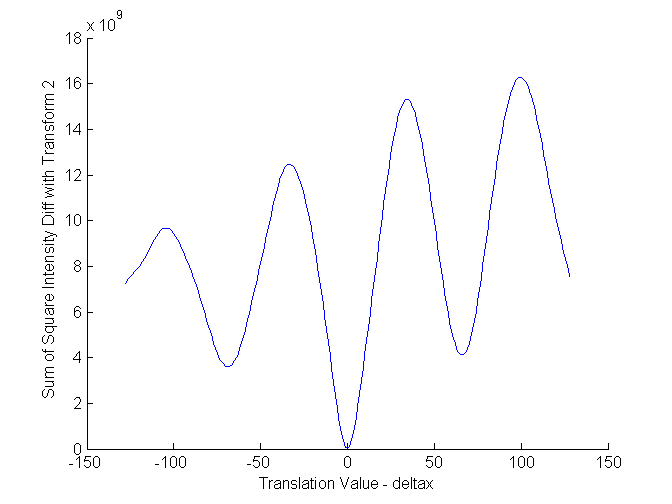
1. alpha=1, beta = 0
2. alpha=1, beta = 20
3. alpha=-1, beta = 1100
4. alpha(x) = 2D Gaussian with mean at the center of the image and standard deviation 300, i.e. roughly 60% of the linear width of the image, beta=0.

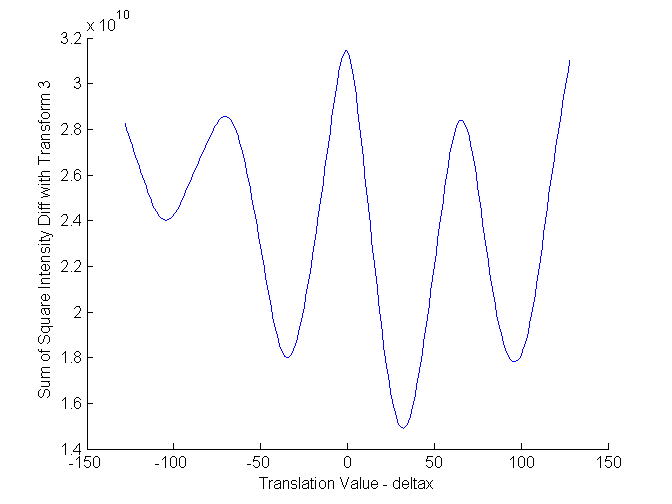
**ANNEXURE-B – IMAGE MISMATCH FUNCTIONS**

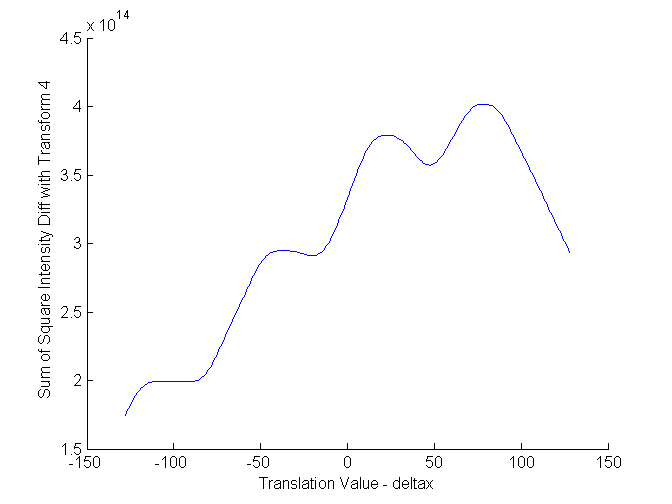
1. Sum of squared intensity differences
2. Sum over 64× 64 squares, into which you divide your image, of the square of the normalized cross correlation over the squares. In the base image comparison region there will be 16 such squares.
3. Sum over the same squares of the Euclidean differences in quantile function on intensity, as modified appropriately to handle both positive and reverse contrast polarity (negative alpha values; see below) relations between the intensities in the fixed and moving images. Explain your modification. Use deciles from 0.0 to 1.0 for quantile estimation.
4. For the same above quantiles, normalize the values using “(q-q(0.5))/(q(0.7)-q(0.3))” equation and use this as image mismatch.
5. Mutual information, obtained by forming the 2D (base intensity, moving image intensity) histogram and computing mutual information from it.

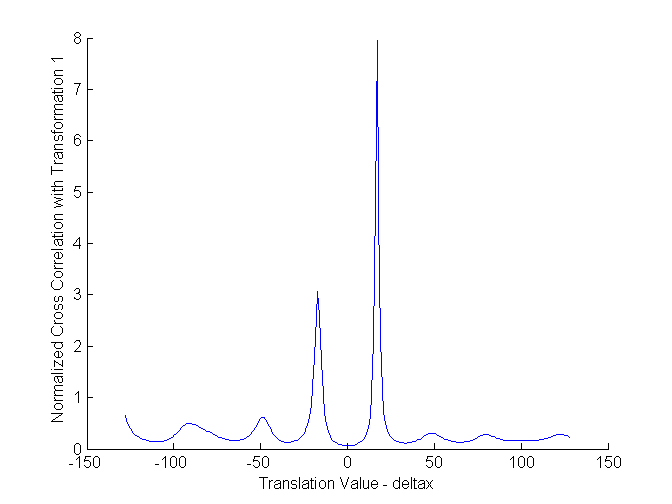
**Note:** Since **Mutual Information** and **Normalized Cross Correlation** indicate how well the two images match or how much of mutual overlap exists or the dependence of one image on the other, the **inverse of these values** are taken to obtain an image mismatch value.

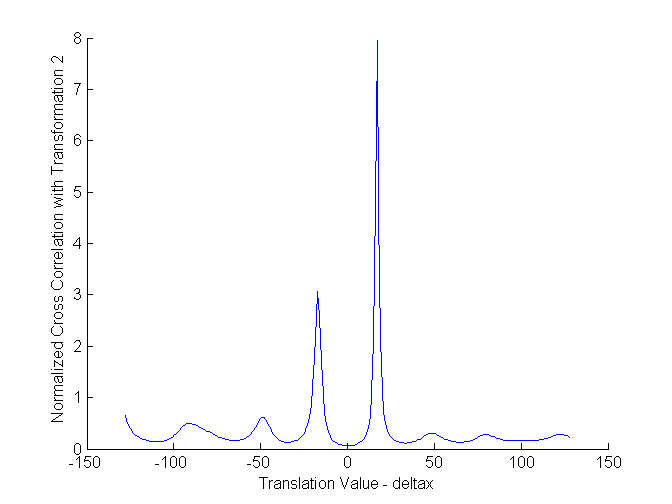


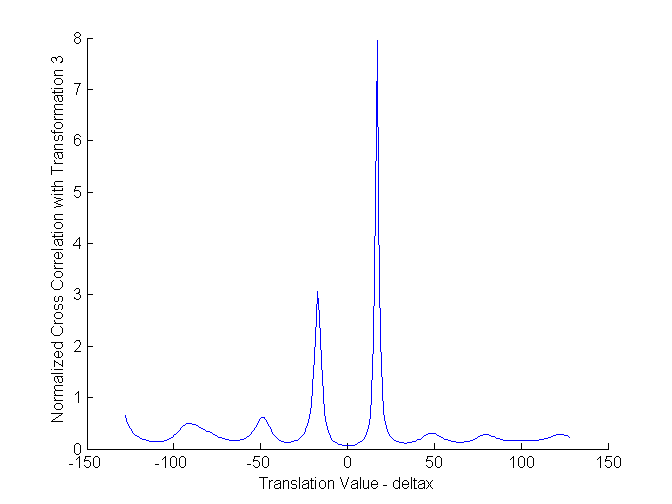


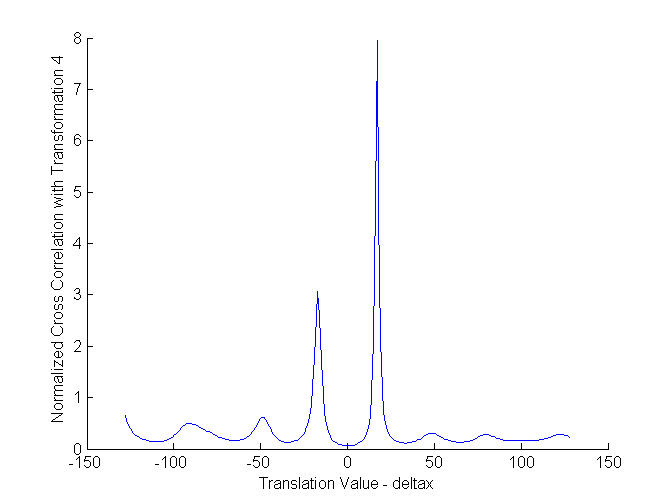


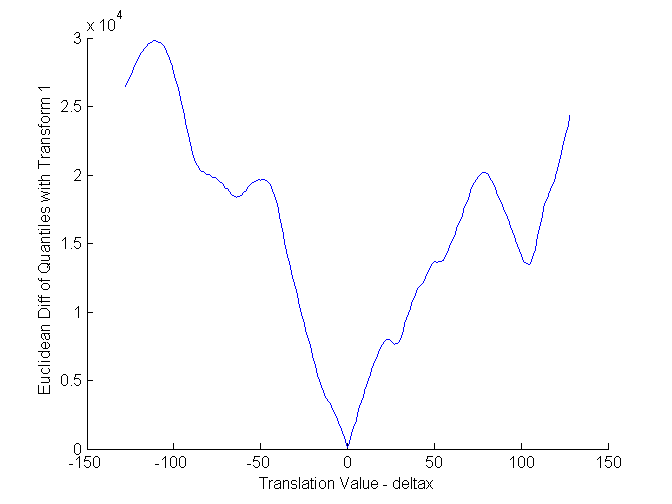


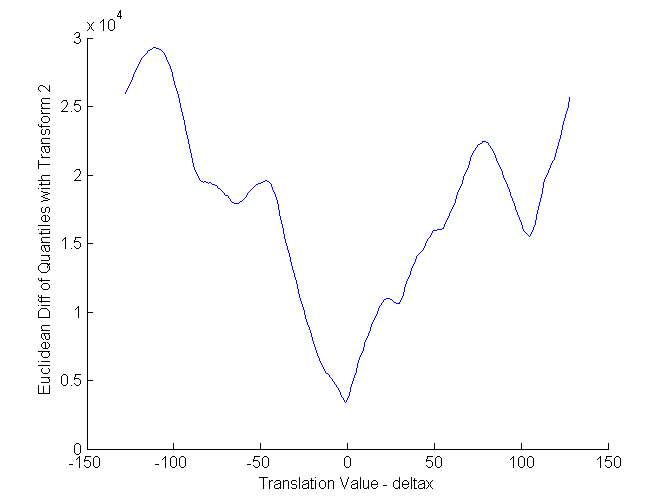


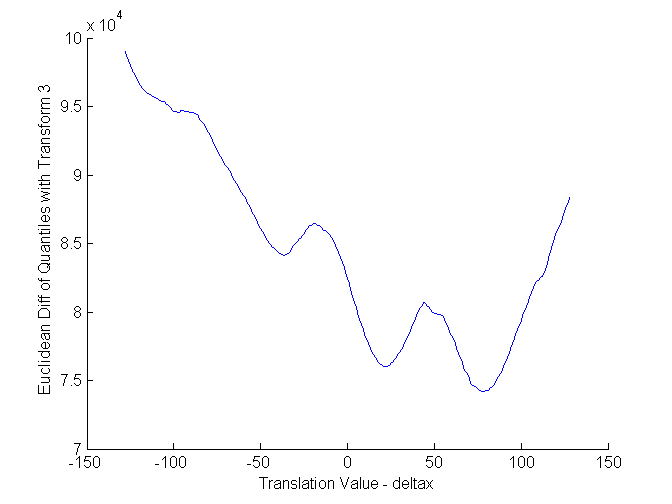


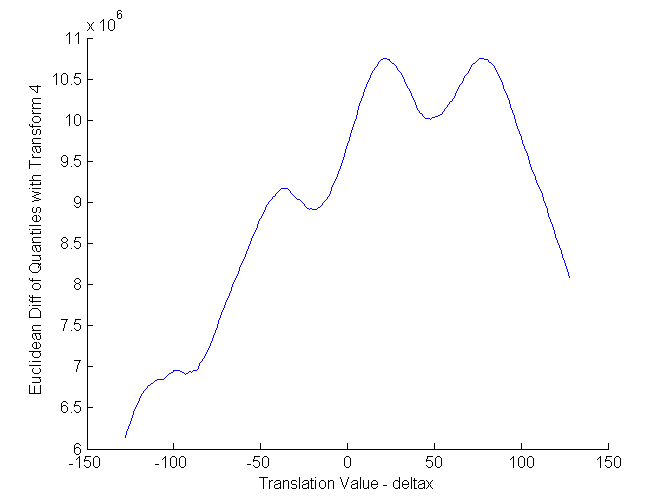


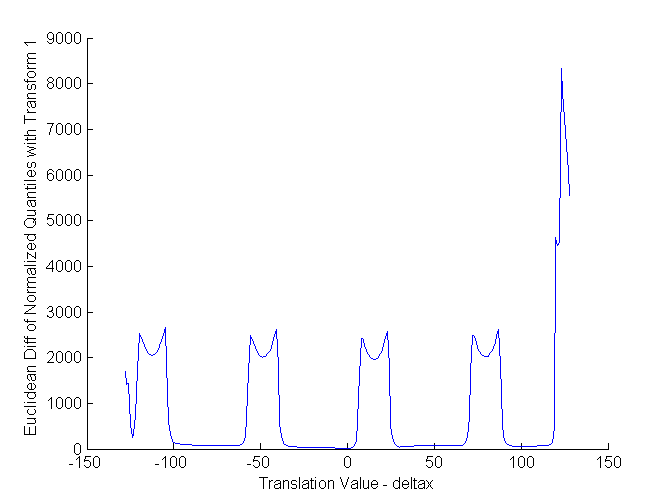


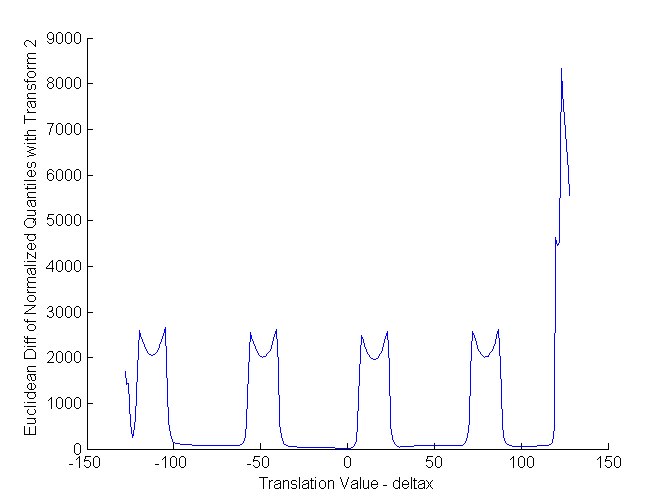


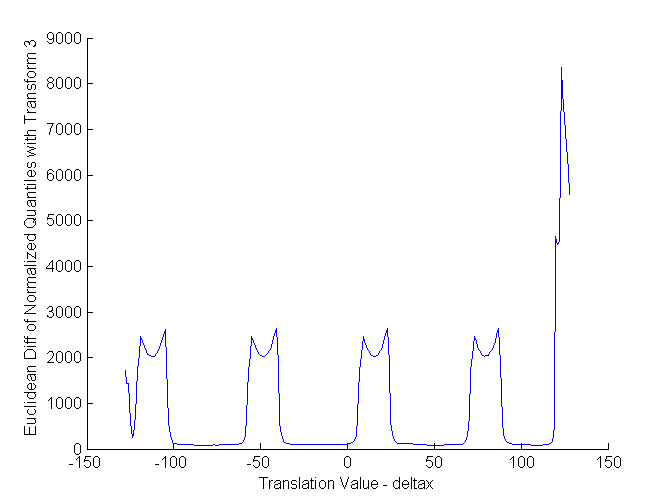


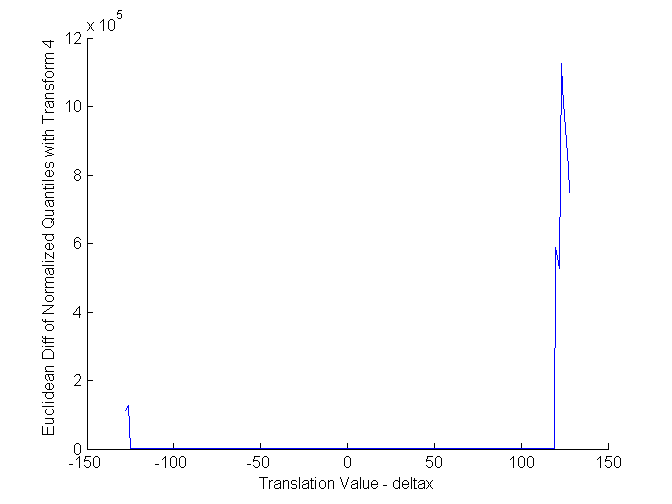


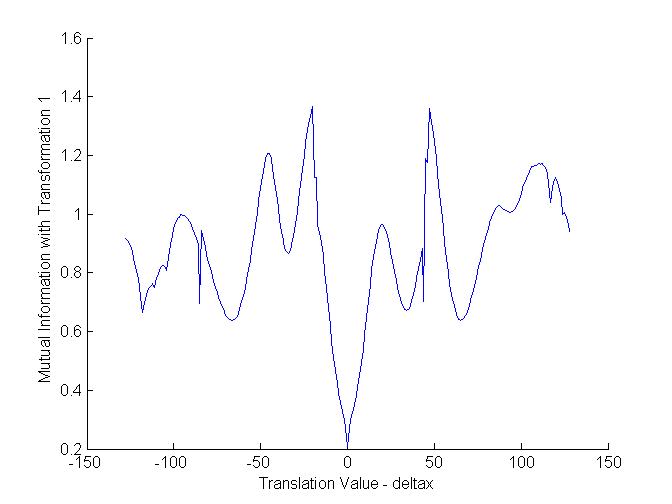


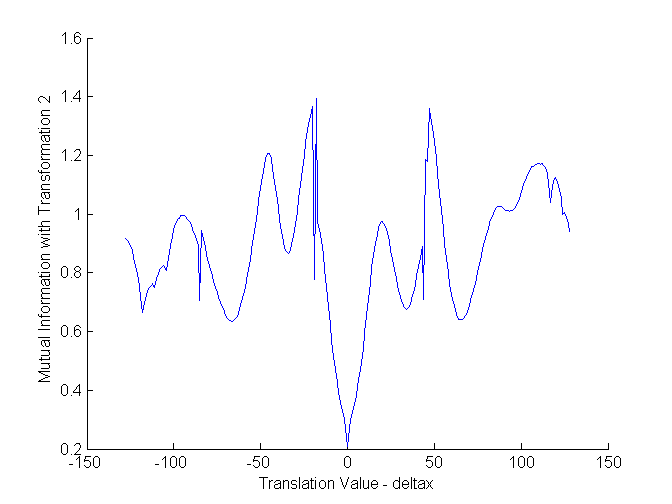


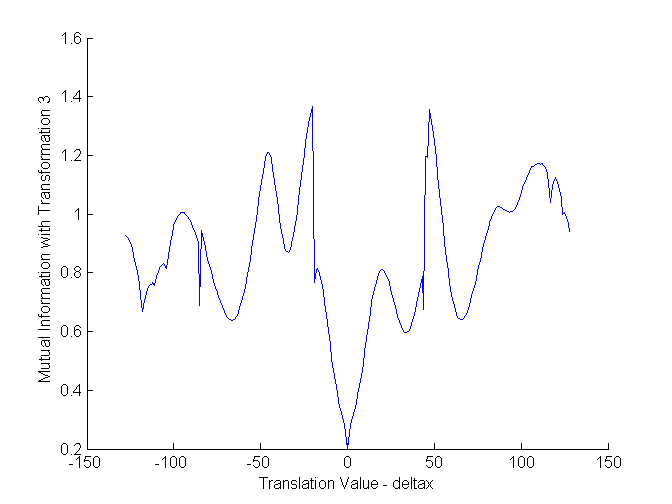


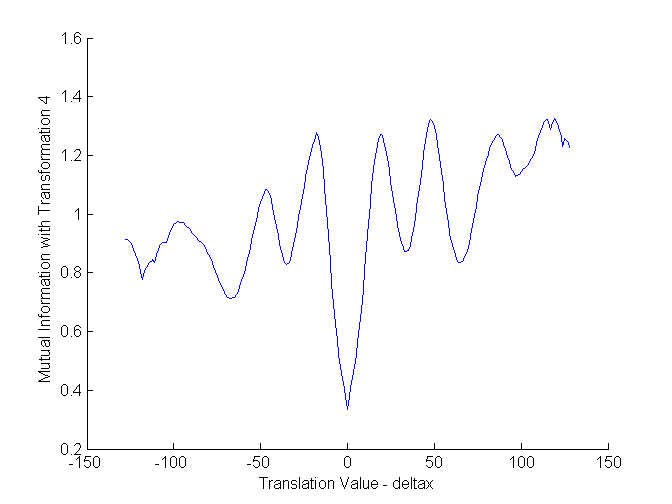












**Analysis**

**1) Sum of squared Intensity Differences:** The image mismatch values plotted against the “x-translation” are shown above (graphs 1-4). The global minimum was located at ‘x-translation’ 0 for intensity transforms 1 and 2, 32 for transform 3 and -128 for transform 4. In the registration program, the global minimum was reached comfortably with Intensity transformations 1 and 2 (as shown in the result tables that follow), however when the Intensity changed drastically in transformations 3 and 4, the program got stuck at local minima.

**2) Normalized Cross-Correlation:** The image mismatch values plotted against the “x-translation” are shown above (graphs 5-8). The global minimum was located at ‘x-translation’ 0 for intensity transforms 1, 2, 3 and 4. The registration program was frequently found to be stuck at local minima (as shown in the result table that follows), but the convergence to the global minimum was more in case of intensity transforms 1 and 2 compared to 3 and 4. It is to be noted that since normalized cross-correlation describes the correlation/dependence between two images, its inverse was taken to indicate the image mismatch.

3) **Quantile Functions:** The image mismatch values plotted against the “x-translation” are shown above (graphs 9-12). The global minimum was located at ‘x-translation’ 0 for intensity transforms 1 and 2, 78 for transform 3 and -128 for transform 4. The registration program converged to the global minimum more frequently when intensity transforms 1 and 2 were applied and rarely converged for transforms 3 and 4. I have used the Matlab ‘quantile’ function to compute the image mismatch and I did not make any separate modification to handle the negative alpha value as the function returned expected results without modficiations.

4) **Normalized Quantile Functions:** The image mismatch values plotted against the “x-translation” are shown above (graphs 13-16). The global minimum was located at ‘x-translation’ 0 for intensity transforms 1, 2 and 4 and -10 for transform 3. The registration program converged to the global minimum more frequently when intensity transforms 1, 2 compared to transform 3 and 4 where the intensity changed drastically.

5) **Mutual Information:** The image mismatch values plotted against the “x-translation” are shown above (graphs 17-20). The global minimum was located at ‘x-translation’ 0 for intensity transforms 1, 2, 3 and 4. The registration program was frequently found to be stuck at local minima (as shown in the result table that follows), but the convergence to the global minimum was more in case of intensity transforms 1 and 2 compared to 3 and 4. It is to be noted that since mutual information describes the commonality between two images, its inverse was taken to indicate the image mismatch.

**It is to be noted for all the above methods, the initialization was crucial as the global minimum was reachable only if the starting ‘x-translation’ was close enough or was not far enough from the global minimum else they all got stuck in local minima.**

(Note: See Annexure-A and Annexure-B for description of image intensity transformations.)

Below follows the result tables in which the experiments were performed with an image mismatch function and corresponding image intensity transform. **5 trials** were conducted where a **random ‘x-translation’** is initially chosen to be **startx** and **endx** is the final translation value reached via image mismatch minimization program. The image mismatch value is noted and compared with global minimum to find out if the minimum reached was a local/global minimum.

In the case of Normalized cross-correlation and Mutual information, inverse values are computed to get image mismatch value and we search the minima of these inverse values.

**Results**

1. Image Mismatch Method = Sum of squared intensity differences

Image Transformation = 1

Global Minimum found by plotting all possible values = 3.274867e+004 at 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **startx** | **endx** | **Image Mismatch Value** | **Is it Local Minima?** |
| 1 | 26 | 0 | 3.274867e+004 | No |
| 2 | -29 | 0 | 3.274867e+004 | No |
| 3 | 33 | 0 | 3.274867e+004 | No |
| 4 | -47 | -69 | 3.690990e+009 | Yes |
| 5 | -50 | -69 | 3.690990e+009 | Yes |

2. Image Mismatch Method = Sum of squared intensity differences

Image Transformation = 2

Global Minimum found by plotting all possible values = 2.758693e+007 at 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **startx** | **endx** | **Image Mismatch Value** | **Is it Local Minima?** |
| 1 | 64 | 66 | 4.119767e+009 | Yes |
| 2 | 32 | 0 | 2.758693e+007 | No |
| 3 | 51 | 66 | 4.119767e+009 | Yes |
| 4 | -43 | -69 | 3.600633e+009 | Yes |
| 5 | -84 | -69 | 3.600633e+009 | Yes |

3. Image Mismatch Method = Sum of squared intensity differences

Image Transformation = 3

Global Minimum found by plotting all possible values = 1.491196e+010 at 32

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **startx** | **endx** | **Image Mismatch Value** | **Is it Local Minima?** |
| 1 | -18 | -35 | 1.801391e+010 | Yes |
| 2 | 67 | 96 | 1.781612e+010 | Yes |
| 3 | 13 | 32 | 1.491196e+010 | No |
| 4 | -22 | -35 | 1.801397e+010 | Yes |
| 5 | 77 | 96 | 1.781612e+010 | Yes |

4. Image Mismatch Method = Sum of squared intensity differences

Image Transformation = 4

Global Minimum found by plotting all possible values = 1.747618e+014 at -128

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **startx** | **endx** | **Image Mismatch Value** | **Is it Local Minima?** |
| 1 | -58 | -99 | 1.990057e+014 | Yes |
| 2 | 22 | -20 | 2.909207e+014 | Yes |
| 3 | 27 | 47 | 3.571666e+014 | Yes |
| 4 | 100 | 146 | 2.695352e+014 | Yes |
| 5 | 69 | 47 | 3.571666e+014 | Yes |

5. Image Mismatch Method = Normalized Cross Correlation (Invert to get mismatch)

Image Transformation = 1

Global Minimum found by plotting all possible values = 6.250025e-002 at 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **startx** | **endx** | **Image Mismatch Value** | **Is it Local Minima?** |
| 1 | -57 | -65 | 1.646666e-001 | Yes |
| 2 | 52 | 65 | 1.328494e-001 | Yes |
| 3 | -24 | -34 | 1.293653e-001 | Yes |
| 4 | 18 | 33 | 1.163895e-001 | Yes |
| 5 | 41 | 33 | 1.163895e-001 | Yes |

6. Image Mismatch Method = Normalized Cross Correlation (Invert to get mismatch)

Image Transformation = 2

Global Minimum found by plotting all possible values = 6.250025e-002 at 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **startx** | **endx** | **Image Mismatch Value** | **Is it Local Minima?** |
| 1 | 9 | 0 | 6.250025e-002 | No |
| 2 | -36 | -34 | 1.293663e-001 | Yes |
| 3 | 48 | 33 | 1.163891e-001 | Yes |
| 4 | -96 | -110 | 1.400706e-001 | Yes |
| 5 | -2 | 0 | 6.250025e-002 | Yes |

7. Image Mismatch Method = Normalized Cross Correlation (Invert to get mismatch)

Image Transformation = 3

Global Minimum found by plotting all possible values = 6.250025e-002 at 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **startx** | **endx** | **Image Mismatch Value** | **Is it Local Minima?** |
| 1 | -80 | -65 | 1.646817e-001 | Yes |
| 2 | 61 | 65 | 1.328465e-001 | Yes |
| 3 | 23 | 33 | 1.163912e-001 | Yes |
| 4 | 9 | 0 | 6.250025e-002 | No |
| 5 | 99 | 96 | 1.603889e-001 | Yes |

8. Image Mismatch Method = Normalized Cross Correlation (Invert to get mismatch)

Image Transformation = 4

Global Minimum found by plotting all possible values = 6.277239e-002 at 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **startx** | **endx** | **Image Mismatch Value** | **Is it Local Minima?** |
| 1 | 77 | 65 | 1.337506e-001 | Yes |
| 2 | -50 | -65 | 1.576975e-001 | Yes |
| 3 | 97 | 96 | 1.635126e-001 | Yes |
| 4 | 43 | 33 | 1.250343e-001 | Yes |
| 5 | -98 | -109 | 1.430642e-001 | Yes |

9. Image Mismatch Method = Quantile Functions

Image Transformation = 1

Global Minimum found by plotting all possible values = 9.080816e+001 at 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **startx** | **endx** | **Image Mismatch Value** | **Is it Local Minima?** |
| 1 | 9 | 0 | 9.080816e+001 | No |
| 2 | -7 | 0 | 9.080816e+001 | No |
| 3 | -91 | -64 | 1.838335e+004 | Yes |
| 4 | -33 | 0 | 9.080816e+001 | No |
| 5 | 54 | 51 | 1.359458e+004 | Yes |

10. Image Mismatch Method = Quantile Functions

Image Transformation = 2

Global Minimum found by plotting all possible values = 3.418886e+003 at 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **startx** | **endx** | **Image Mismatch Value** | **Is it Local Minima?** |
| 1 | 72 | 51 | 1.594811e+004 | Yes |
| 2 | 11 | 0 | 3.418886e+003 | No |
| 3 | -81 | -80 | 1.941308e+004 | Yes |
| 4 | 68 | 51 | 1.594851e+004 | Yes |
| 5 | -66 | -65 | 1.790544e+004 | Yes |

11. Image Mismatch Method = Quantile Functions

Image Transformation = 3

Global Minimum found by plotting all possible values = 7.420422e+004 at 78

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **startx** | **endx** | **Image Mismatch Value** | **Is it Local Minima?** |
| 1 | 34 | 22 | 7.602265e+004 | Yes |
| 2 | -46 | -37 | 8.412073e+004 | Yes |
| 3 | 64 | 78 | 7.420422e+004 | No |
| 4 | 16 | 22 | 7.602265e+004 | Yes |
| 5 | 56 | 78 | 7.420422e+004 | No |

12. Image Mismatch Method = Quantile Functions

Image Transformation = 4

Global Minimum found by plotting all possible values = 6.143643e+006 at -128

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **startx** | **endx** | **Image Mismatch Value** | **Is it Local Minima?** |
| 1 | 18 | -19 | 8.907576e+006 | Yes |
| 2 | 90 | 128 | 8.083583e+006 | Yes |
| 3 | 50 | 48 | 1.001421e+007 | Yes |
| 4 | -75 | -90 | 6.935927e+006 | Yes |
| 5 | -46 | -90 | 6.935927e+006 | Yes |

13. Image Mismatch Method = Normalized Quantile Functions

Image Transformation = 1

Global Minimum found by plotting all possible values = 2.892949e-001 at 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **startx** | **endx** | **Image Mismatch Value** | **Is it Local Minima?** |
| 1 | 44 | 55 | 6.622079e+001 | Yes |
| 2 | -14 | 0 | 2.892949e-001 | No |
| 3 | -92 | -85 | 7.395698e+001 | Yes |
| 4 | -79 | -85 | 7.395698e+001 | Yes |
| 5 | -44 | -49 | 2.009845e+003 | Yes |

14. Image Mismatch Method = Normalized Quantile Functions

Image Transformation = 2

Global Minimum found by plotting all possible values = 2.883188e-001 at 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **startx** | **endx** | **Image Mismatch Value** | **Is it Local Minima?** |
| 1 | -9 | 0 | 2.883188e-001 at 0 | No |
| 2 | -63 | -65 | 7.769623e+001 | Yes |
| 3 | 80 | 79 | 2.014122e+003 | Yes |
| 4 | 31 | 30 | 4.351601e+001 | Yes |
| 5 | -72 | -75 | 8.116120e+001 | Yes |

15. Image Mismatch Method = Normalized Quantile Functions

Image Transformation = 3

Global Minimum found by plotting all possible values = 8.319905e+001 at -10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **startx** | **endx** | **Image Mismatch Value** | **Is it Local Minima?** |
| 1 | 2 | -10 | 8.319905e+001 | No |
| 2 | -68 | -75 | 8.595024e+001 | Yes |
| 3 | 51 | 54 | 8.570028e+001 | Yes |
| 4 | -62 | -75 | 8.595024e+001 | Yes |
| 5 | -51 | -49 | 2.029650e+003 | Yes |

16. Image Mismatch Method = Normalized Quantile Functions

Image Transformation = 4

Global Minimum found by plotting all possible values = 6.983253e+000 at 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **startx** | **endx** | **Image Mismatch Value** | **Is it Local Minima?** |
| 1 | 100 | 104 | 4.947212e+001 | Yes |
| 2 | 12 | 15 | 2.088804e+002 | Yes |
| 3 | 74 | 80 | 2.536278e+002 | Yes |
| 4 | -58 | -65 | 7.738811e+001 | Yes |
| 5 | 48 | 55 | 6.294758e+001 | Yes |

17. Image Mismatch Method = Mutual Information (Invert to get mismatch)

Image Transformation = 1

Global Minimum found by plotting all possible values = 2.095722e-001 at 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **startx** | **endx** | **Image Mismatch Value** | **Is it Local Minima?** |
| 1 | -17 | 0 | 2.095722e-001 | No |
| 2 | -75 | -66 | 6.348768e-001 | Yes |
| 3 | -26 | -34 | 8.662688e-001 | Yes |
| 4 | 88 | 93 | 1.006705e+000 | Yes |
| 5 | 84 | 65 | 6.389567e-001 | Yes |

18. Image Mismatch Method = Mutual Information (Invert to get mismatch)

Image Transformation = 2

Global Minimum found by plotting all possible values = 2.100643e-001 at 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **startx** | **endx** | **Image Mismatch Value** | **Is it Local Minima?** |
| 1 | -100 | -104 | 8.070211e-001 | Yes |
| 2 | -18 | -19 | 7.711670e-001 | Yes |
| 3 | 80 | 65 | 6.387632e-001 | Yes |
| 4 | 7 | 0 | 2.100643e-001 | No |
| 5 | -53 | -66 | 6.360859e-001 | Yes |

19. Image Mismatch Method = Mutual Information (Invert to get mismatch)

Image Transformation = 3

Global Minimum found by plotting all possible values = 2.016125e-001 at 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **startx** | **endx** | **Image Mismatch Value** | **Is it Local Minima?** |
| 1 | 71 | 65 | 6.401470e-001 | Yes |
| 2 | 29 | 34 | 5.979192e-001 | Yes |
| 3 | 3 | 0 | 2.016125e-001 | No |
| 4 | 55 | 65 | 6.401470e-001 | Yes |
| 5 | -37 | -34 | 8.676084e-001 | Yes |

20. Image Mismatch Method = Mutual Information (Invert to get mismatch)

Image Transformation = 4

Global Minimum found by plotting all possible values = 3.341187e-001 at 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **startx** | **endx** | **Image Mismatch Value** | **Is it Local Minima?** |
| 1 | 75 | 64 | 8.335159e-001 | Yes |
| 2 | 41 | 34 | 8.714125e-001 | Yes |
| 3 | 87 | 97 | 1.129029e+000 | Yes |
| 4 | 11 | 0 | 3.341187e-001 | No |
| 5 | 2 | 0 | 3.341187e-001 | No |

**Code**

The following is the code for image mismatch functions, the registration program and the intensity transformation function.

1) intensity\_transformation.m

function output = intensity\_transformation(img, alpha, beta, sigma)

%intensity\_transformation - produces the intensity transformation image

if nargin < 4

disp('Please pass all the required parameters to intensity\_transformation function, Exiting');

quit;

end

length = size(img, 1);

width = size(img, 2);

gauss\_noise = imnoise(img, 'gaussian', 0, sigma\*sigma);

output = zeros(length, width);

alphaType = class(alpha);

if ~strcmp(alphaType, 'char')

for i = 1 : length

for j = 1 : width

output(j,i) = alpha \* img(j,i) + beta + gauss\_noise(j,i);

end

end

else

g = fspecial('gaussian', [512 512], 300);

img\_gauss = imfilter(img, g, 'replicate');

for i = 1 : length

for j = 1 : width

output(j,i) = img\_gauss(j,i) \* img(j,i) + beta + gauss\_noise(j,i);

end

end

end

end

2) sum\_of\_squared\_intensity\_differences.m

function sum = sum\_of\_squared\_intensity\_differences( img1, img2 , startx, endx, starty, endy)

%sum\_of\_intensity\_differences - Computes the sum of squared intensity

%differences from (startx, starty) to (endx, endy)

if nargin < 6

disp('Please pass all the required parameters to sum\_of\_squared\_intensity\_differences, Exiting');

quit;

end

sum = 0.0;

t1 = 0.0;

t2 = 0.0;

term = 0.0;

maxterm = 0.0;

for i = starty : endy

for j = startx : endx

t1 = img1(j,i);

t2 = img2(j,i);

term = t1 - t2;

term = term \* term;

if maxterm < term

maxterm = term;

end

sum = sum + term;

end

end

%disp(sum);

end

3) registration\_squarediff.m

% Image mismatch program

% Read the base image

baseimg = double(imread('../FixedImage.png'));

maxx = size(baseimg,1);

maxy = size(baseimg,2);

% Generate the intensity transformation

transform\_choice = 1;

gauss\_sigma = 10;

movimg1 = intensity\_transformation(baseimg, 1, 0, gauss\_sigma);

movimg2 = intensity\_transformation(baseimg, 1, 20, gauss\_sigma);

movimg3 = intensity\_transformation(baseimg, -1, 1100, gauss\_sigma);

movimg4 = intensity\_transformation(baseimg, 'gauss', 0, gauss\_sigma);

switch transform\_choice

case 1

movimg = movimg1;

case 2

movimg = movimg2;

case 3

movimg = movimg3;

case 4

movimg = movimg4;

end

% Compute the moving image which is intensity transformed and translated

tempimg = zeros(maxx,maxy);

initval = 128;

startx = 128;

starty = 128;

deltax = randi([-100 100],1);

deltaxArr = zeros(1,initval\*2+1);

mismatchArr = zeros(1,initval\*2+1);

fprintf('Chosen deltax = %d\n',deltax);

%Compute mismatch at deltax

T = [1 0 0; 0 1 0; deltax 0 1];

tform = maketform('affine', T);

[tempimg, xdata, ydata] = imtransform(movimg, tform, ...

'XData', [1 size(movimg,2)], 'YData', [1 size(movimg,1)]);

mismatch\_val2 = sum\_of\_squared\_intensity\_differences( baseimg, ...

tempimg, startx+1, startx+256, starty+1, starty+256);

deltaxArr(1) = deltax;

mismatchArr(1) = mismatch\_val2;

%Compute mismatch at deltax-1

T = [1 0 0; 0 1 0; deltax-1 0 1];

tform = maketform('affine', T);

[tempimg, xdata, ydata] = imtransform(movimg, tform, ...

'XData', [1 size(movimg,2)], 'YData', [1 size(movimg,1)]);

mismatch\_val1 = sum\_of\_squared\_intensity\_differences( baseimg, ...

tempimg, startx+1, startx+256, starty+1, starty+256);

%Compute mismatch at deltax+1

T = [1 0 0; 0 1 0; deltax+1 0 1];

tform = maketform('affine', T);

[tempimg, xdata, ydata] = imtransform(movimg, tform, ...

'XData', [1 size(movimg,2)], 'YData', [1 size(movimg,1)]);

mismatch\_val3 = sum\_of\_squared\_intensity\_differences( baseimg, ...

tempimg, startx+1, startx+256, starty+1, starty+256);

skipFlag = false;

if mismatch\_val3 > mismatch\_val2 && mismatch\_val2 > mismatch\_val1

step = -1;

curr\_val = mismatch\_val1;

deltax = deltax+step;

elseif mismatch\_val3 < mismatch\_val2 && mismatch\_val2 < mismatch\_val1

step = 1;

curr\_val = mismatch\_val3;

deltax = deltax+step;

else

skipFlag = true;

step = 1;

curr\_val = mismatch\_val2;

end

if ~skipFlag

deltaxArr(2) = deltax;

mismatchArr(2) = curr\_val;

end

deltax = deltax + step;

count = 3;

while true

T = [1 0 0; 0 1 0; deltax 0 1];

tform = maketform('affine', T);

[tempimg, xdata, ydata] = imtransform(movimg, tform, ...

'XData', [1 size(movimg,2)], 'YData', [1 size(movimg,1)]);

next\_val = sum\_of\_squared\_intensity\_differences( baseimg, ...

tempimg, startx+1, startx+256, starty+1, starty+256);

if next\_val > curr\_val || deltax < -128 || deltax > 128

%fprintf('deltax = %d, val = %e\n',deltax,next\_val);

deltax = deltax - step;

count = count - 1;

break

end

deltaxArr(count) = deltax;

mismatchArr(count) = next\_val;

%fprintf('deltax = %d, val = %e\n',deltax,next\_val);

curr\_val = next\_val;

deltax = deltax+step;

count = count + 1;

end

x = zeros(1,count);

y = zeros(1,count);

for i = 1 : count

x(i) = deltaxArr(i);

y(i) = mismatchArr(i);

end

fprintf('startx = %d\n', deltaxArr(1));

fprintf('endx = %d\n', deltaxArr(count));

fprintf('mismatch = %e\n', mismatchArr(count));

figure;

axis on;

xlabel('Translation Value - deltax');

yl = char(strcat('Sum of Square Intensity Diff with Transform ',int2str(transform\_choice)));

ylabel(yl);

hold on;

plot(x,y);

4) normalized\_cross\_correlation.m

function totalSum = normalized\_cross\_correlation( img1, img2 , startx, endx, starty, endy)

%normalized\_cross\_correlation - computes normalized cross-correlation

if nargin < 6

disp('Please pass all the required parameters to normalized cross-correlation, Exiting');

quit;

end

totalSum = 0.0;

x1 = startx;

y1 = starty;

x2 = x1 + 64 - 1;

y2 = y1 + 64 - 1;

while true

if x2 > endx

x1 = startx;

x2 = x1 + 64 - 1;

y1 = y2 + 1;

y2 = y1 + 64 - 1;

end

if y2 > endy

break;

end

tempimg1 = img1(x1:x2, y1:y2);

tempimg2 = img2(x1:x2, y1:y2);

m1 = mean2(tempimg1);

m2 = mean2(tempimg2);

s1 = std2(tempimg1);

s2 = std2(tempimg2);

s = 0.0;

count = 0;

for i = 1 : size(tempimg1,2)

for j = 1 : size(tempimg1,1)

s = s + ( ((tempimg1(j,i) - m1) \* (tempimg2(j,i) - m2)) / (s1 \* s2));

count = count + 1;

end

end

s = s / (count - 1);

totalSum = totalSum + s\*s;

x1 = x2 + 1;

x2 = x1 + 64 - 1;

end

totalSum = 1 / totalSum;

%disp(totalSum);

end

5) registration\_norm\_cross\_corr.m

% Image mismatch program

% Read the base image

baseimg = double(imread('../FixedImage.png'));

maxx = size(baseimg,1);

maxy = size(baseimg,2);

% Generate the intensity transformation

transform\_choice = 1;

gauss\_sigma = 10;

movimg1 = intensity\_transformation(baseimg, 1, 0, gauss\_sigma);

movimg2 = intensity\_transformation(baseimg, 1, 20, gauss\_sigma);

movimg3 = intensity\_transformation(baseimg, -1, 1100, gauss\_sigma);

movimg4 = intensity\_transformation(baseimg, 'gauss', 0, gauss\_sigma);

switch transform\_choice

case 1

movimg = movimg1;

case 2

movimg = movimg2;

case 3

movimg = movimg3;

case 4

movimg = movimg4;

end

% Compute the moving image which is intensity transformed and translated

tempimg = zeros(maxx,maxy);

initval = 128;

startx = 128;

starty = 128;

deltax = randi([-100 100],1);

deltaxArr = zeros(1,initval\*2+1);

mismatchArr = zeros(1,initval\*2+1);

fprintf('Chosen deltax = %d\n',deltax);

%Compute mismatch at deltax

T = [1 0 0; 0 1 0; deltax 0 1];

tform = maketform('affine', T);

[tempimg, xdata, ydata] = imtransform(movimg, tform, ...

'XData', [1 size(movimg,2)], 'YData', [1 size(movimg,1)]);

mismatch\_val2 = normalized\_cross\_correlation( baseimg, ...

tempimg, startx+1, startx+256, starty+1, starty+256);

deltaxArr(1) = deltax;

mismatchArr(1) = mismatch\_val2;

%Compute mismatch at deltax-1

T = [1 0 0; 0 1 0; deltax-1 0 1];

tform = maketform('affine', T);

[tempimg, xdata, ydata] = imtransform(movimg, tform, ...

'XData', [1 size(movimg,2)], 'YData', [1 size(movimg,1)]);

mismatch\_val1 = normalized\_cross\_correlation( baseimg, ...

tempimg, startx+1, startx+256, starty+1, starty+256);

%Compute mismatch at deltax+1

T = [1 0 0; 0 1 0; deltax+1 0 1];

tform = maketform('affine', T);

[tempimg, xdata, ydata] = imtransform(movimg, tform, ...

'XData', [1 size(movimg,2)], 'YData', [1 size(movimg,1)]);

mismatch\_val3 = normalized\_cross\_correlation( baseimg, ...

tempimg, startx+1, startx+256, starty+1, starty+256);

skipFlag = false;

if mismatch\_val3 > mismatch\_val2 && mismatch\_val2 > mismatch\_val1

step = -1;

curr\_val = mismatch\_val1;

deltax = deltax+step;

elseif mismatch\_val3 < mismatch\_val2 && mismatch\_val2 < mismatch\_val1

step = 1;

curr\_val = mismatch\_val3;

deltax = deltax+step;

else

skipFlag = true;

step = 1;

curr\_val = mismatch\_val2;

end

if ~skipFlag

deltaxArr(2) = deltax;

mismatchArr(2) = curr\_val;

end

deltax = deltax + step;

count = 3;

while true

T = [1 0 0; 0 1 0; deltax 0 1];

tform = maketform('affine', T);

[tempimg, xdata, ydata] = imtransform(movimg, tform, ...

'XData', [1 size(movimg,2)], 'YData', [1 size(movimg,1)]);

next\_val = normalized\_cross\_correlation( baseimg, ...

tempimg, startx+1, startx+256, starty+1, starty+256);

if next\_val > curr\_val || deltax < -128 || deltax > 128

%fprintf('deltax = %d, val = %f\n',deltax,next\_val);

deltax = deltax - step;

count = count - 1;

break

end

deltaxArr(count) = deltax;

mismatchArr(count) = next\_val;

%fprintf('deltax = %d, val = %f\n',deltax,next\_val);

curr\_val = next\_val;

deltax = deltax+step;

count = count + 1;

end

x = zeros(1,count);

y = zeros(1,count);

for i = 1 : count

x(i) = deltaxArr(i);

y(i) = mismatchArr(i);

end

fprintf('startx = %d\n', deltaxArr(1));

fprintf('endx = %d\n', deltaxArr(count));

fprintf('mismatch = %e\n', mismatchArr(count));

figure;

axis on;

xlabel('Translation Value - deltax');

yl = char(strcat('Sum of Square Intensity Diff with Transform ',int2str(transform\_choice)));

ylabel(yl);

hold on;

plot(x,y);

6) quantile\_function\_diff.m

function totalSum = quantile\_function\_diff( img1, img2 , startx, endx, starty, endy, normalized)

%normalized\_cross\_correlation - computes normalized cross-correlation

if nargin < 6

disp('Please pass all the required parameters to normalized cross-correlation, Exiting');

quit;

end

if nargin < 7

normalized = 0;

end

quantDist = [0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0];

totalSum = 0.0;

x1 = startx;

y1 = starty;

x2 = x1 + 64 - 1;

y2 = y1 + 64 - 1;

while true

if x2 > endx

x1 = startx;

x2 = x1 + 64 - 1;

y1 = y2 + 1;

y2 = y1 + 64 - 1;

end

if y2 > endy

break;

end

% take transpose and reshape into a one dimension array

tempimg1 = reshape(img1(x1:x2, y1:y2)', 1, 64\*64);

tempimg2 = reshape(img2(x1:x2, y1:y2)', 1, 64\*64);

quant1 = quantile(tempimg1, quantDist);

quant2 = quantile(tempimg2, quantDist);

if normalized

% Normalize by (q - q(0.5)) / (q(0.7) - q(0.3))

quant1 = (quant1 - quant1(6)) / (quant1(8) - quant1(4));

quant2 = (quant2 - quant2(6)) / (quant2(8) - quant2(4));

end

term = sum(abs(quant1-quant2));

totalSum = totalSum + term;

x1 = x2 + 1;

x2 = x1 + 64 - 1;

end

end

7) registration\_quantilediff.m

% Image mismatch program

% Read the base image

baseimg = double(imread('../FixedImage.png'));

maxx = size(baseimg,1);

maxy = size(baseimg,2);

% Generate the intensity transformation

transform\_choice = 1;

normalized\_quantile = 0;

gauss\_sigma = 10;

movimg1 = intensity\_transformation(baseimg, 1, 0, gauss\_sigma);

movimg2 = intensity\_transformation(baseimg, 1, 20, gauss\_sigma);

movimg3 = intensity\_transformation(baseimg, -1, 1100, gauss\_sigma);

movimg4 = intensity\_transformation(baseimg, 'gauss', 0, gauss\_sigma);

switch transform\_choice

case 1

movimg = movimg1;

case 2

movimg = movimg2;

case 3

movimg = movimg3;

case 4

movimg = movimg4;

end

% Compute the moving image which is intensity transformed and translated

tempimg = zeros(maxx,maxy);

initval = 128;

startx = 128;

starty = 128;

deltax = randi([-100 100],1);

deltaxArr = zeros(1,initval\*2+1);

mismatchArr = zeros(1,initval\*2+1);

fprintf('Chosen deltax = %d\n',deltax);

%Compute mismatch at deltax

T = [1 0 0; 0 1 0; deltax 0 1];

tform = maketform('affine', T);

[tempimg, xdata, ydata] = imtransform(movimg, tform, ...

'XData', [1 size(movimg,2)], 'YData', [1 size(movimg,1)]);

mismatch\_val2 = quantile\_function\_diff( baseimg, ...

tempimg, startx+1, startx+256, starty+1, starty+256, normalized\_quantile);

deltaxArr(1) = deltax;

mismatchArr(1) = mismatch\_val2;

%Compute mismatch at deltax-1

T = [1 0 0; 0 1 0; deltax-1 0 1];

tform = maketform('affine', T);

[tempimg, xdata, ydata] = imtransform(movimg, tform, ...

'XData', [1 size(movimg,2)], 'YData', [1 size(movimg,1)]);

mismatch\_val1 = quantile\_function\_diff( baseimg, ...

tempimg, startx+1, startx+256, starty+1, starty+256, normalized\_quantile);

%Compute mismatch at deltax+1

T = [1 0 0; 0 1 0; deltax+1 0 1];

tform = maketform('affine', T);

[tempimg, xdata, ydata] = imtransform(movimg, tform, ...

'XData', [1 size(movimg,2)], 'YData', [1 size(movimg,1)]);

mismatch\_val3 = quantile\_function\_diff( baseimg, ...

tempimg, startx+1, startx+256, starty+1, starty+256, normalized\_quantile);

skipFlag = false;

if mismatch\_val3 > mismatch\_val2 && mismatch\_val2 > mismatch\_val1

step = -1;

curr\_val = mismatch\_val1;

deltax = deltax+step;

elseif mismatch\_val3 < mismatch\_val2 && mismatch\_val2 < mismatch\_val1

step = 1;

curr\_val = mismatch\_val3;

deltax = deltax+step;

else

skipFlag = true;

step = 1;

curr\_val = mismatch\_val2;

end

if ~skipFlag

deltaxArr(2) = deltax;

mismatchArr(2) = curr\_val;

end

deltax = deltax + step;

count = 3;

while true

T = [1 0 0; 0 1 0; deltax 0 1];

tform = maketform('affine', T);

[tempimg, xdata, ydata] = imtransform(movimg, tform, ...

'XData', [1 size(movimg,2)], 'YData', [1 size(movimg,1)]);

next\_val = quantile\_function\_diff( baseimg, ...

tempimg, startx+1, startx+256, starty+1, starty+256, normalized\_quantile);

if next\_val > curr\_val || deltax < -128 || deltax > 128

%fprintf('deltax = %d, val = %f\n',deltax,next\_val);

deltax = deltax - step;

count = count - 1;

break

end

deltaxArr(count) = deltax;

mismatchArr(count) = next\_val;

%fprintf('deltax = %d, val = %f\n',deltax,next\_val);

curr\_val = next\_val;

deltax = deltax+step;

count = count + 1;

end

x = zeros(1,count);

y = zeros(1,count);

for i = 1 : count

x(i) = deltaxArr(i);

y(i) = mismatchArr(i);

end

fprintf('startx = %d\n', deltaxArr(1));

fprintf('endx = %d\n', deltaxArr(count));

fprintf('mismatch = %e\n', mismatchArr(count));

if 0

figure;

axis on;

xlabel('Translation Value - deltax');

yl = char(strcat('Sum of Square Intensity Diff with Transform ',int2str(transform\_choice)));

ylabel(yl);

hold on;

plot(x,y);

end

8) mi.m

function I=mi(A,B,varargin)

%MI Determines the mutual information of two images or signals

%

% I=mi(A,B) Mutual information of A and B, using 256 bins for

% histograms

% I=mi(A,B,L) Mutual information of A and B, using L bins for histograms

%

% Assumption: 0\*log(0)=0

if nargin>=3

L=varargin{1};

else

L=256;

end

A=double(A);

B=double(B);

na = hist(A(:),L);

na = na/sum(na);

nb = hist(B(:),L);

nb = nb/sum(nb);

n2 = hist2(A,B,L);

n2 = n2/sum(n2(:));

I=sum(minf(n2,na'\*nb));

% return inverse for image\_mismatch

I = 1/I;

% -----------------------

function y=minf(pab,papb)

I=find(papb(:)>1e-12 & pab(:)>1e-12); % function support

y=pab(I).\*log2(pab(I)./papb(I));

9) hist2.m

function n=hist2(A,B,L)

%HIST2 Calculates the joint histogram of two images or signals

%

% n=hist2(A,B,L) is the joint histogram of matrices A and B, using L

% bins for each matrix.

ma=min(A(:));

MA=max(A(:));

mb=min(B(:));

MB=max(B(:));

% For sensorimotor variables, in [-pi,pi]

% ma=-pi;

% MA=pi;

% mb=-pi;

% MB=pi;

% Scale and round to fit in {0,...,L-1}

A=round((A-ma)\*(L-1)/(MA-ma+eps));

B=round((B-mb)\*(L-1)/(MB-mb+eps));

n=zeros(L);

x=0:L-1;

for i=0:L-1

n(i+1,:) = histc(B(A==i),x,1);

end

end

10) registration\_mi.m

% Image mismatch program

% Read the base image

baseimg = double(imread('../FixedImage.png'));

maxx = size(baseimg,1);

maxy = size(baseimg,2);

% Generate the intensity transformation

transform\_choice = 1;

gauss\_sigma = 10;

movimg1 = intensity\_transformation(baseimg, 1, 0, gauss\_sigma);

movimg2 = intensity\_transformation(baseimg, 1, 20, gauss\_sigma);

movimg3 = intensity\_transformation(baseimg, -1, 1100, gauss\_sigma);

movimg4 = intensity\_transformation(baseimg, 'gauss', 0, gauss\_sigma);

switch transform\_choice

case 1

movimg = movimg1;

case 2

movimg = movimg2;

case 3

movimg = movimg3;

case 4

movimg = movimg4;

end

% Compute the moving image which is intensity transformed and translated

tempimg = zeros(maxx,maxy);

initval = 128;

startx = 128;

starty = 128;

deltax = randi([-100 100],1);

deltaxArr = zeros(1,initval\*2+1);

mismatchArr = zeros(1,initval\*2+1);

fprintf('Chosen deltax = %d\n',deltax);

%Compute mismatch at deltax

T = [1 0 0; 0 1 0; deltax 0 1];

tform = maketform('affine', T);

[tempimg, xdata, ydata] = imtransform(movimg, tform, ...

'XData', [1 size(movimg,2)], 'YData', [1 size(movimg,1)]);

mismatch\_val2 = mi(baseimg(startx+1:startx+256,starty+1:starty+256), ...

tempimg(startx+1:startx+256,starty+1:starty+256));

deltaxArr(1) = deltax;

mismatchArr(1) = mismatch\_val2;

%Compute mismatch at deltax-1

T = [1 0 0; 0 1 0; deltax-1 0 1];

tform = maketform('affine', T);

[tempimg, xdata, ydata] = imtransform(movimg, tform, ...

'XData', [1 size(movimg,2)], 'YData', [1 size(movimg,1)]);

mismatch\_val1 = mi(baseimg(startx+1:startx+256,starty+1:starty+256), ...

tempimg(startx+1:startx+256,starty+1:starty+256));

%Compute mismatch at deltax+1

T = [1 0 0; 0 1 0; deltax+1 0 1];

tform = maketform('affine', T);

[tempimg, xdata, ydata] = imtransform(movimg, tform, ...

'XData', [1 size(movimg,2)], 'YData', [1 size(movimg,1)]);

mismatch\_val3 = mi(baseimg(startx+1:startx+256,starty+1:starty+256), ...

tempimg(startx+1:startx+256,starty+1:starty+256));

skipFlag = false;

if mismatch\_val3 > mismatch\_val2 && mismatch\_val2 > mismatch\_val1

step = -1;

curr\_val = mismatch\_val1;

deltax = deltax+step;

elseif mismatch\_val3 < mismatch\_val2 && mismatch\_val2 < mismatch\_val1

step = 1;

curr\_val = mismatch\_val3;

deltax = deltax+step;

else

skipFlag = true;

step = 1;

curr\_val = mismatch\_val2;

end

if ~skipFlag

deltaxArr(2) = deltax;

mismatchArr(2) = curr\_val;

end

deltax = deltax + step;

count = 3;

while true

T = [1 0 0; 0 1 0; deltax 0 1];

tform = maketform('affine', T);

[tempimg, xdata, ydata] = imtransform(movimg, tform, ...

'XData', [1 size(movimg,2)], 'YData', [1 size(movimg,1)]);

next\_val = mi(baseimg(startx+1:startx+256,starty+1:starty+256), ...

tempimg(startx+1:startx+256,starty+1:starty+256));

if next\_val > curr\_val || deltax < -128 || deltax > 128

%fprintf('deltax = %d, val = %f\n',deltax,next\_val);

deltax = deltax - step;

count = count - 1;

break

end

deltaxArr(count) = deltax;

mismatchArr(count) = next\_val;

%fprintf('deltax = %d, val = %f\n',deltax,next\_val);

curr\_val = next\_val;

deltax = deltax+step;

count = count + 1;

end

x = zeros(1,count);

y = zeros(1,count);

for i = 1 : count

x(i) = deltaxArr(i);

y(i) = mismatchArr(i);

end

fprintf('startx = %d\n', deltaxArr(1));

fprintf('endx = %d\n', deltaxArr(count));

fprintf('mismatch = %e\n', mismatchArr(count));

figure;

axis on;

xlabel('Translation Value - deltax');

yl = char(strcat('Sum of Square Intensity Diff with Transform ',int2str(transform\_choice)));

ylabel(yl);

hold on;

plot(x,y);