Assignment 6 - Signal and Image Processing 2014

Martin Jørgensen, tzk173

October 14, 2014

Contents

1 Task 1: Experiments on translations

Part 1

The general recipe for creating a translation transformation matrix is

$$M_{\text{trans}} = \begin{bmatrix} 1 & 0 & x_d \\ 0 & 1 & y_d \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

$$\begin{bmatrix} g_x \\ g_y \\ \_ \end{bmatrix} = M_{\text{trans}} \times \begin{bmatrix} f_x \\ f_y \\ 1 \end{bmatrix} \tag{2}$$

where x_d and y_d are the offsets we wish to impose on the x and y-axis respectively and f_x and f_y are coordinates in the original image.

Using Equation 1 the matrix for translating 1 pixel to the right is

$$M = \begin{bmatrix} 1 & 0 & x_d \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

If we wish to create a kernel/filter for translating $1px$ towards the right, we use the following matrix

$$M_{filter} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

The code for creating this matrix is found in the appendix.

When convolved with an image this filter will make all pixels assume the value of their left neighbour, effectively translation the entire image $1px$ to the right.

Part 2

Using the method described in Equation 1 and 2 I created the function `MyTranslate` which takes an image and two integers as input, the integers are the offset to translate with. The code for the function and for creating Figure 1 as well as the code for the function can be found in the appendix.

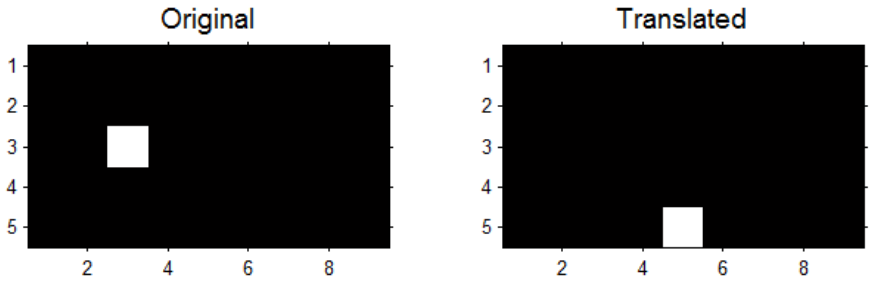

Figure 1: The image before and after translation.

The handling of boundary conditions in my implementation is simple, all of the output image is initialized to 0. This means that pixels that don't get assigned a value will just stay zero, coordinates from the old image that translates outside the bounds of the output (which have the same size as the input), are simply ignored.

Part 3

Translation in the Frequency domain is given by

$$f(x - \Delta x, y - \Delta y) = F(u, v)e^{-i2\pi(\frac{u\Delta x}{N} + \frac{v\Delta y}{M})} \tag{3}$$

Where M is the number of rows in the image, and N is the number of columns.

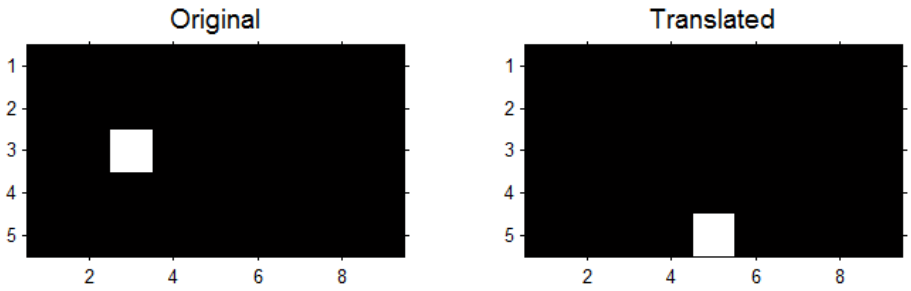

Figure 2: The image before and after translation.

The code for producing Figure 2 can be found in the appendix along with the code for my translation function called `MyFTranslate`.

Part 4

The code for producing Figure 3 can be found in the appendix.

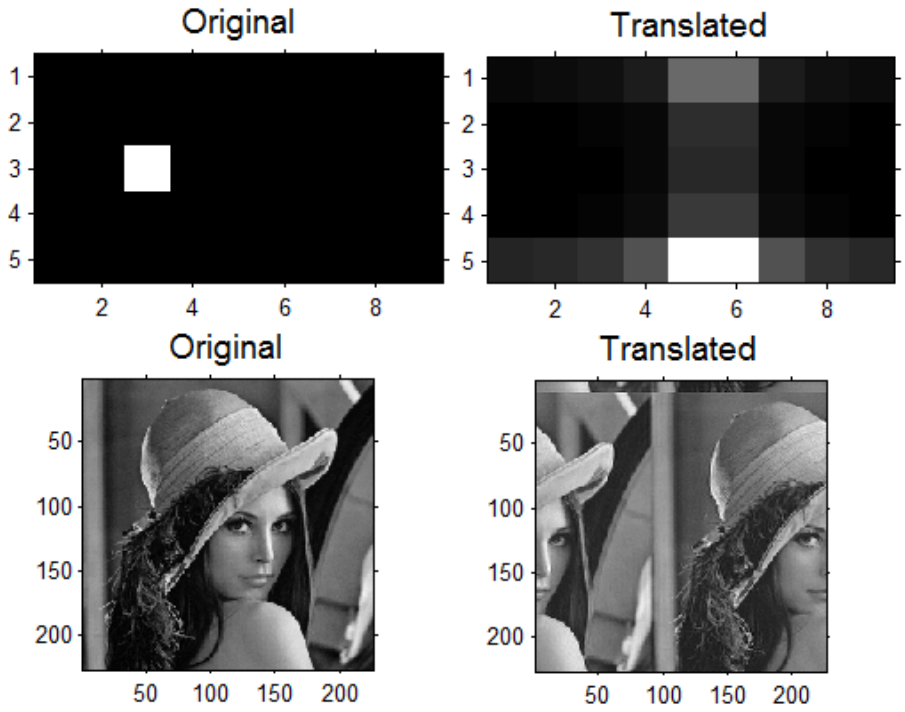

Figure 3: The image before and after translation.

We can see that the translation does not go smoothly, in the lena image we see that the image gets significantly brighter when it gets translated, and in the point image we can see why, the bright spots gets "smeared" out in the directions the image gets translated.

2 Task 2: Procrustes transformations

Part 1

For translation only, the minimum N is 1, since we can do translation from one point to another, by subtracting them. The same goes for scaling, except it is simply multiplying the single point until the MSE stops falling.

Part 2

The code for producing the values in Equation 4 and Figure 4 can be found in the appendix.

Filling the values printed from `p22.m` into Equation (7.13) from p. 176 into the matrices we get the result show in Equation 4.

$$\mathbf{X} = \begin{bmatrix} 1 & 0 & -80.4378 \\ 0 & 1 & 6.2076 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{R} = \begin{bmatrix} 0.9864 & -0.1647 & 0 \\ 0.1647 & 0.9863 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{S} = \begin{bmatrix} 1.0893 & 0 & 0 \\ 0 & 1.0893 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4)$$

Figure 4: The two images and the result of performing a procrustes transformation and overlaying the result with the original.

In Figure 4 we see that the features of the images overlay very well meaning the MRE for the alignment will be very small.

3 Task 4: Affine and projective alignments

Part 1

The main appeal is the ability to use one single matrix multiplication to do all the transformations needed. It also allows us to append transformations as a chain of matrix multiplications. A single transformation would be

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = M \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (5)$$

Since the result of the multiplication in Equation 5 is also a 3-element vector, we can simply chain multiplications on like so

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = M_n, M_{n-1}, ..., M_0 \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (6)$$

Part 2

The code for producing Figure 5 can be found in the appendix.

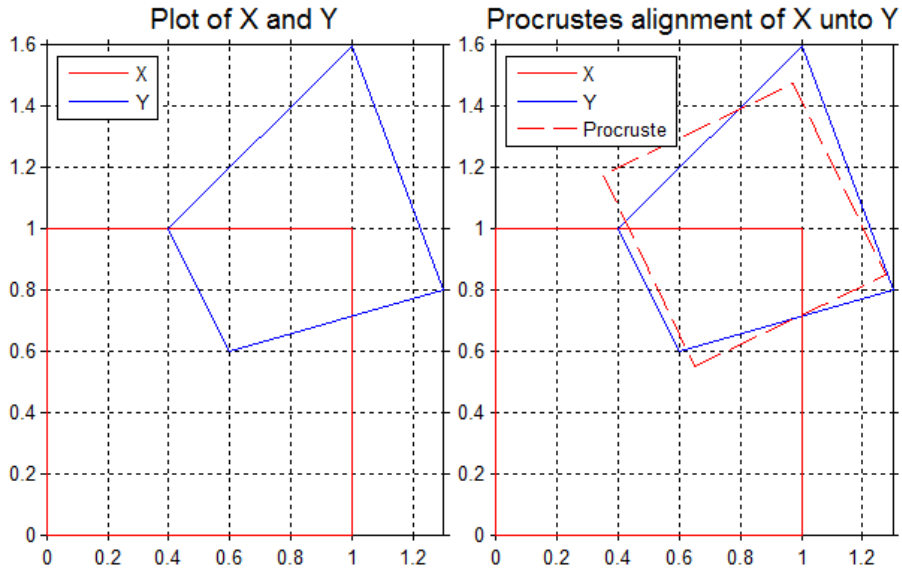

Figure 5: Left: X and Y drawn in the same plot. Right: Same plot, but with a Procrustes aligned X overlaid.

It will not be possible to get an exact match for X mapped to Y since the edges in X are pairwise parallel, whereas only 2 edges in Y are parallel. Neither translation, rotation or scaling, the angles between edges in the figure. Figure 5 shows the closest mapping Procrustes analysis can create.

Part 3

The code for producing Figure 6 can be found in the appendix.

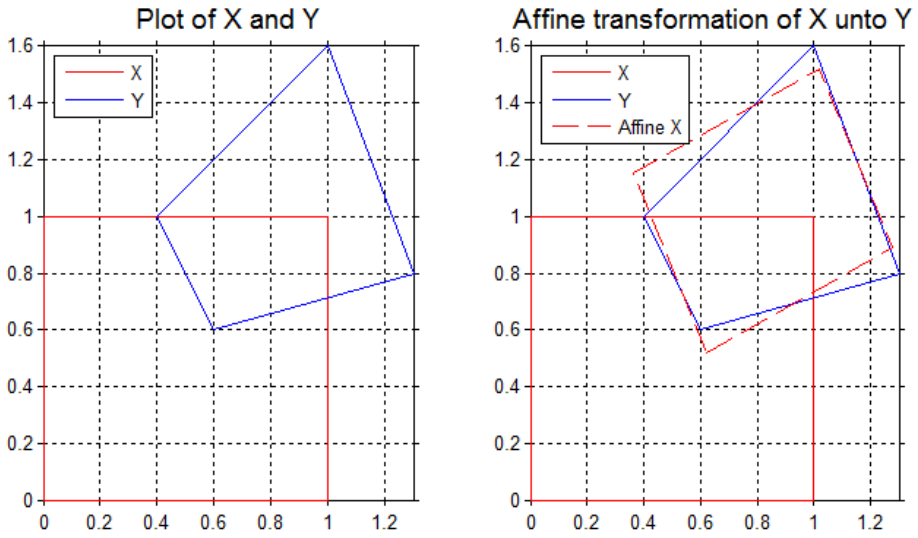

Figure 6: Left: X and Y drawn in the same plot. Right: Same plot, but with a Affine transformed X overlaid.

Figure 6 shows the result of the affine transformation, we can see that because we're also able to shear the figure, we get a slightly closer match. We are still not able to break the parallel relationship of edges though.

Part 4

The code for producing Figure 7 can be found in the appendix.

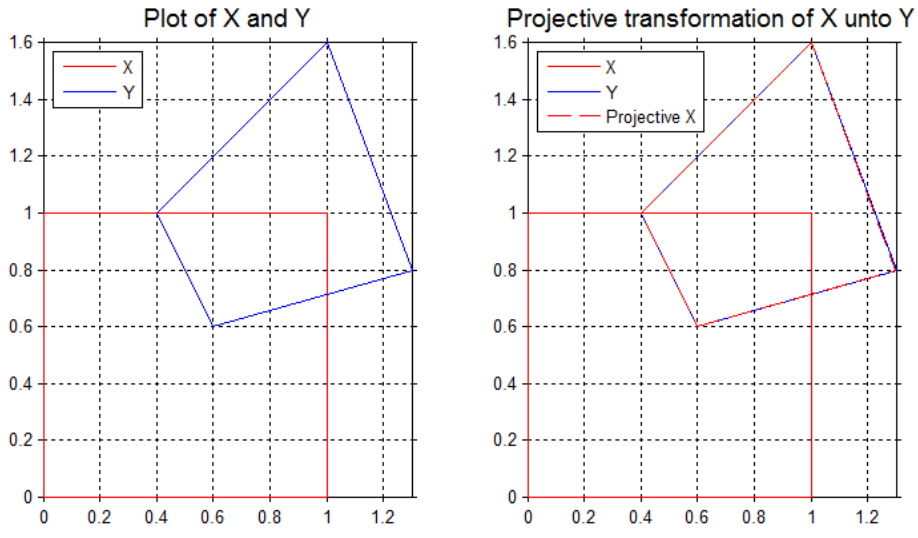

Figure 7: Left: X and Y drawn in the same plot. Right: Same plot, but with a Projective transformed X overlaid.

Figure 7 is the result of using Projective transformation on X. The ability to manipulate X in 3D space allows us to break the parallel relationship of edges and make a near perfect match.

According to Matlab there is some small differences, but they seem so small that it might be due to lost floating point precision in the processing. The transformed X and Y are to all intents and purposes matching.

A Appendix

Code for Task 1

```
% Solution for part 1.1 of Assignment 6.
% Written by: Martin Jrgensen, tzk173

clear all;

M = zeros(3);
M(2,1) = 1
```

Figure 8: Produce Matrix for Task 1.1 (../p11.m)

```
% Solution for part 1.2 of Assignment 6.
% Written by: Martin Jrgensen, tzk173

clear all;

% Create an image with a centered white box.
I1 = zeros(5,9);
I1(3,3) = 1;

% Translate down to lower right corner.
I2 = MyTranslate(I1, 2, 2);

% Show results.
h = figure(612); set(h,'Color','White'); colormap(gray);
subplot(1,2,1); imagesc(I1); axis image; set(gca,'TickDir','out');
title('Original','FontSize',14);
subplot(1,2,2); imagesc(I2); axis image; set(gca,'TickDir','out');
title('Translated','FontSize',14);
```

Figure 9: Produce Figure for Task 1.2 (../p12.m)

```
% Solution for part 1.3 of Assignment 6.
% Written by: Martin Jrgensen, tzk173

clear all;

% Create an image with a centered white box.
I1 = zeros(5,9);
I1(3,3) = 1;

% Translate down to lower right corner.
I2 = MyFTranslate(I1, 2, 2);

% Show results.
h = figure(613); set(h,'Color','White'); colormap(gray);
subplot(1,2,1); imagesc(I1); axis image; set(gca,'TickDir','out');
title('Original','FontSize',14);
subplot(1,2,2); imagesc(I2); axis image; set(gca,'TickDir','out');
title('Translated','FontSize',14);
```

Figure 10: Produce Figure for Task 1.3 (../p13.m)

```
% Solution for part 1.4 of Assignment 6.
% Written by: Martin Jrgensen, tzk173

clear all;

% Create an image with a centered white box.
I1 = zeros(5,9);
I1(3,3) = 1;
I2 = imread('lena.tif');


% Translate down to lower right corner.
I3 = MyFTranslate(I1, 2.5, 2.3);
I4 = MyFTranslate(I2, 90.5, 9.3);

% Show results.
h = figure(614); set(h,'Color','White'); colormap(gray);
subplot(2,2,1); imagesc(I1); axis image; set(gca,'TickDir','out');
title('Original','FontSize',14);
subplot(2,2,2); imagesc(I3); axis image; set(gca,'TickDir','out');
title('Translated','FontSize',14);
subplot(2,2,3); imagesc(I2); axis image; set(gca,'TickDir','out');
title('Original','FontSize',14);
subplot(2,2,4); imagesc(I4); axis image; set(gca,'TickDir','out');
title('Translated','FontSize',14);
```

Figure 11: Produce Figure for Task 1.4 (../p14.m)

```
function [ g ] = MyTranslate( f, dx, dy )
%MyTranslate Translate an image f with dx pixels along the x-axis and dy
%pixels along the y-axis.
    M = eye(3); M(1,3) = dx; M(2,3) = dy;

    [sx, sy] = size(f);
    g = zeros(sx,sy);
    for x=1:sx
        for y=1:sy
            np = M*[x y 1]';
            nx = np(1);
            ny = np(2);

            if (nx <= sx && ny <= sy)
                g(nx, ny) = f(x,y);
            end
        end
    end

end
```

Figure 12: Ordinary translation function using transformation matrix. (../MyTranslate.m)

```
function [ g ] = MyFTranslate( f, dx, dy )
%MyTranslate Translate an image f with dx pixels along the x-axis and dy
%pixels along the y-axis. Using FFT this time.

    F=fftshift(fft2(f));

    [sx, sy] = size(f);

    [u,v] = meshgrid(-(sy/2):((sy/2)-1), -(sx/2):((sx/2)-1));
    ex = exp(-i*2*pi*((u*dx)/sy + (v*dy)/sx));
    F=F.*ex;

    g = abs(real(ifft2(ifftshift(F))));
end
```

Figure 13: Translation function using Fast Fourier Transform. (../MyFTranslate.m)

Code for Task 2

```
% Solution for part 2.2 of Assignment 6.
% Written by: Martin Jrgensen, tzk173

clear all;

% Load Images
I1 = imread('westconcordaerial.png');
I2 = imread('westconcordorthophoto.png');

% Use Points that ship with matlab
load westconcordpoints;
% Or let the user pick
% figure, imshow(I1);
% [x,y] = ginput(4); fixedPoints = [x,y];
% figure, imshow(I2);
% [x,y] = ginput(4); movingPoints = [x,y];

% Calculate transformation and perform it.
TFORM = cp2tform(movingPoints,fixedPoints,'affine');
info = imfinfo('westconcordorthophoto.png');
I3 = imtransform(I1,TFORM,'XData',[1 info.Width], 'YData',[1 info.Height]);

% Overlay the two images.
I3 = imfuse(I2, I3,'blend','Scaling','joint');

% Display procrustes information
[D,Z,T] = procrustes(movingPoints,fixedPoints);
% Print Translation, Scale factor and Rotation.
T.c, T.b, T.T

h = figure(622); set(h,'Color','White'); colormap(gray);
subplot(1,3,1); imagesc(I1); axis image; set(gca,'TickDir','out');
title('westconcordaerial.png','FontSize',14);
subplot(1,3,2); imagesc(I2); axis image; set(gca,'TickDir','out');
title('westconcordorthophoto.png','FontSize',14);
subplot(1,3,3); imagesc(I3); axis image; set(gca,'TickDir','out');
title('Result','FontSize',14);
```

Figure 14: Produce Figure for Task 2.2 (../p22.m)

Code for Task 4

```
% Solution for part 4.2 of Assignment 6.
% Written by: Martin Jrgensen, tzk173

clear all;

% Define the points
X = [[0,0]
     [1,0]
     [1,1]
     [0,1]
     [0,0]];
Y = [[0.6,0.6]
     [1.3,0.8]
     [1,1.6]
     [0.4,1]
     [0.6,0.6]];

 % Perform analysis & alignment.
[d,Z, TFORM] = procrustes(Y,X);

% Show results.
h = figure(642); set(h,'Color','White');

subplot(1,2,1);
plot(X(:,1), X(:,2),'r',Y(:,1), Y(:,2),'b');
legend('X','Y','Location','northwest'); grid on; axis image;
title('Plot of X and Y','FontSize',14); set(gca,'TickDir','out');

subplot(1,2,2);
plot(X(:,1), X(:,2),'r',Y(:,1), Y(:,2),'b',Z(:,1), Z(:,2),'--r');
legend('X','Y','Procruste','Location','northwest'); grid on; axis image;
title('Procrustes alignment of X unto Y','FontSize',14); set(gca,'TickDir','out')
    ;
```

Figure 15: Produce Figure for Task 4.2 (../p42.m)

```
% Solution for part 4.3 of Assignment 6.
% Written by: Martin Jrgensen, tzk173

clear all;

% Define the points
X = [[0,0]
     [1,0]
     [1,1]
     [0,1]
     [0,0]];
Y = [[0.6,0.6]
     [1.3,0.8]
     [1,1.6]
     [0.4,1]
     [0.6,0.6]];

 % Perform analysis & alignment.
 TFORM = cp2tform(X, Y, 'affine');
 Z = tformfwd(TFORM, X(:,1), X(:,2));

% Show results.
h1 = figure(643); set(h1,'Color','White');

subplot(1,2,1);
plot(X(:,1), X(:,2),'r',Y(:,1), Y(:,2),'b');
legend('X','Y','Location','northwest'); grid on; axis image;
title('Plot of X and Y','FontSize',14); set(gca,'TickDir','out');

subplot(1,2,2);
plot(X(:,1), X(:,2),'r',Y(:,1), Y(:,2),'b',Z(:,1), Z(:,2),'--r');
legend('X','Y','Affine X','Location','northwest'); grid on; axis image;
title('Affine transformation of X unto Y','FontSize',14); set(gca,'TickDir','out'
    );
```

Figure 16: Produce Figure for Task 4.3 (../p43.m)

```
% Solution for part 4.4 of Assignment 6.
% Written by: Martin Jrgensen, tzk173

clear all;

% Define the points
X = [[0,0]
     [1,0]
     [1,1]
     [0,1]
     [0,0]];
Y = [[0.6,0.6]
     [1.3,0.8]
     [1,1.6]
     [0.4,1]
     [0.6,0.6]];

% Perform analysis & alignment.
TFORM = cp2tform(X, Y, 'projective');
Z = tformfwd(TFORM, X(:,1), X(:,2));

% Show results.
h1 = figure(644); set(h1,'Color','White');

subplot(1,2,1);
plot(X(:,1), X(:,2),'r',Y(:,1), Y(:,2),'b');
legend('X','Y','Location','northwest'); grid on; axis image;
title('Plot of X and Y','FontSize',14); set(gca,'TickDir','out');

subplot(1,2,2);
plot(X(:,1), X(:,2),'r',Y(:,1), Y(:,2),'b',Z(:,1), Z(:,2),'--r');
legend('X','Y','Projective X','Location','northwest'); grid on; axis image;
title('Projective transformation of X unto Y','FontSize',14); set(gca,'TickDir','
    out');
```

Figure 17: Produce Figure for Task 4.4 (../p44.m)