Signal and Image Processing
Assignment 6

Nikolaj Dybdahl Rathcke (rfq695)

March 21, 2016

1 Experiments on translations

1.1

The image $\widetilde{I}(i,j)$ which is obtained by shifting the image I one pixel to the right is called a translation of I and can mathematically be expressed as $\widetilde{I}(i,j) = I(i-1,j)$. The filter h for the operation can be expressed as:

$$h = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

which, looking at one pixel, would simply only include the intesity of the pixel to the left.

1.2

The code found in `src/q1_2.m` translates an image I of size $(77, 77)$ with a white square in the middle of size 3 by 3. It is translated with the vector $d = \begin{bmatrix} 30 \\ 30 \end{bmatrix}$ using Matlab's built-in function `imtranslate`, but the method `shiftImg` takes any parameters for x and y. Figure 1 show the original image and the result after translation with d.

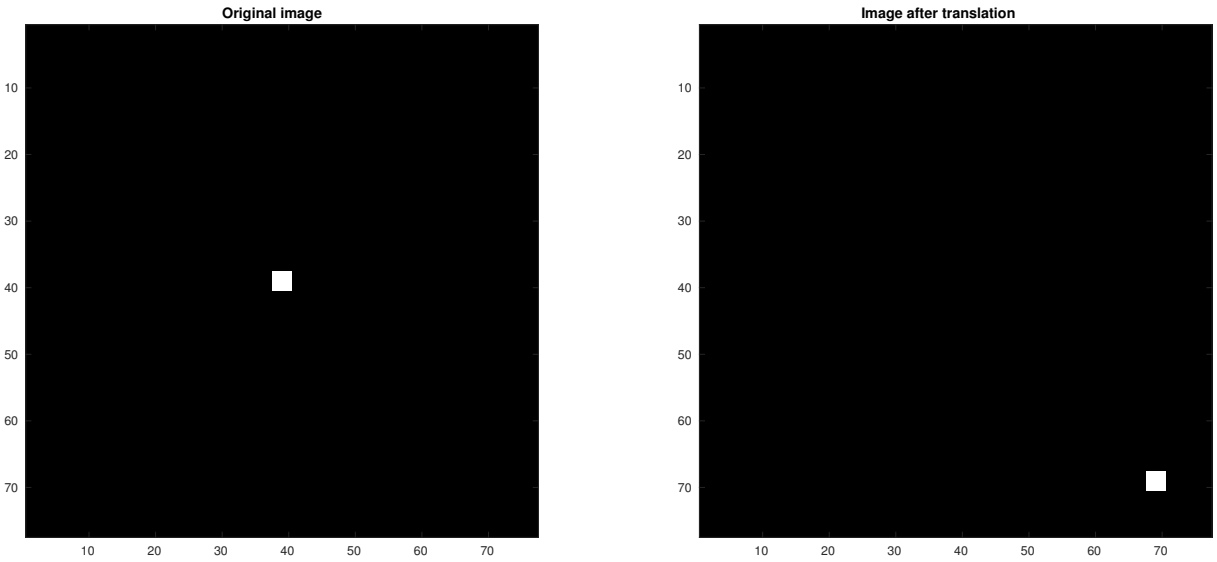

Figure 1: An image I and the result after translating it with vector d.

Where the image is shifted 30 pixel along both axes. However, we need to consider that it might try to access an index which is out of bounds for I. To overcome this, we could extend the image with rows and columns corresponding to the given x and y, with all values set to some constant (0 in this image). If the filter is constructed in the same way as (1.1), we can expect it to have size $x \cdot y$, so extending the image would ensure that the translation is possible. We would have to remove the rows and columns we added afterwards (meaning we could move the white square out of the image, so we are left with a completely black image).

1.3

A property of the Fourier transform is that a shift in the time domain does not make a difference for the frequencies, but it does change the phase of the frequencies. The shift theorem states that a translation in one of the domains corresponds to a multiplication by some complex exponential in the other domain. This thought is implemented in `src/q1_3.m` and produces Figure 2.

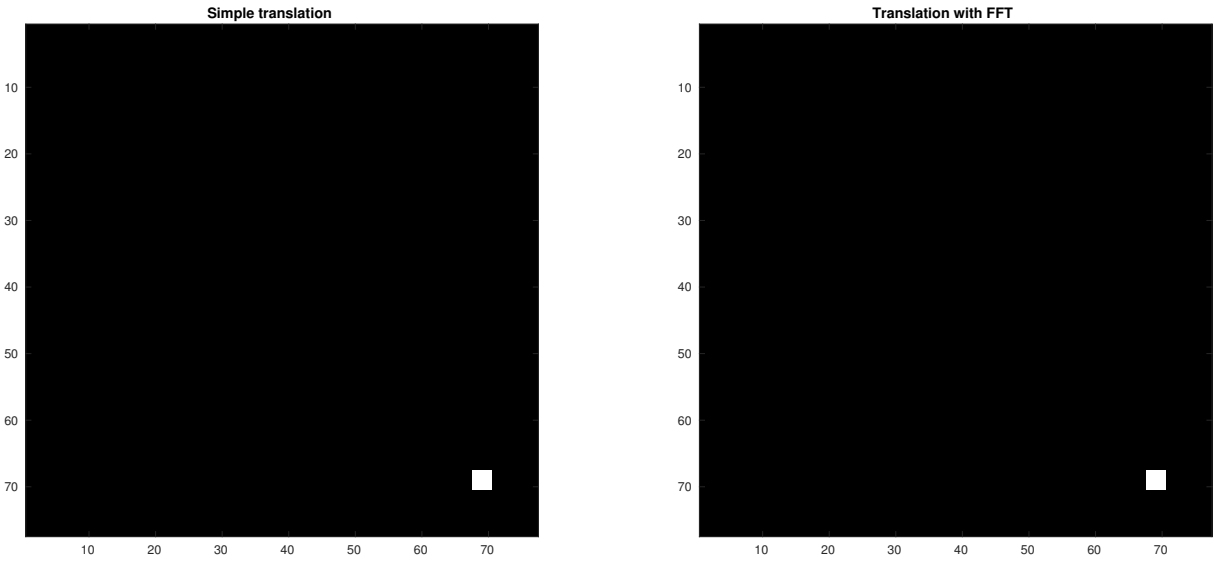

Figure 2: Comparison between using a simple translation and translation with FFT.

As we can see, the result from using translation with FFT is exactly the same as we found in (1.2).

1.4

This is possible as the translation in the frequency domain is performed by shifting the phase and this is possible for any real number. However, if we shift it with a non-integer, the phase will affect more than one pixel. Figure 3 show the result of this.

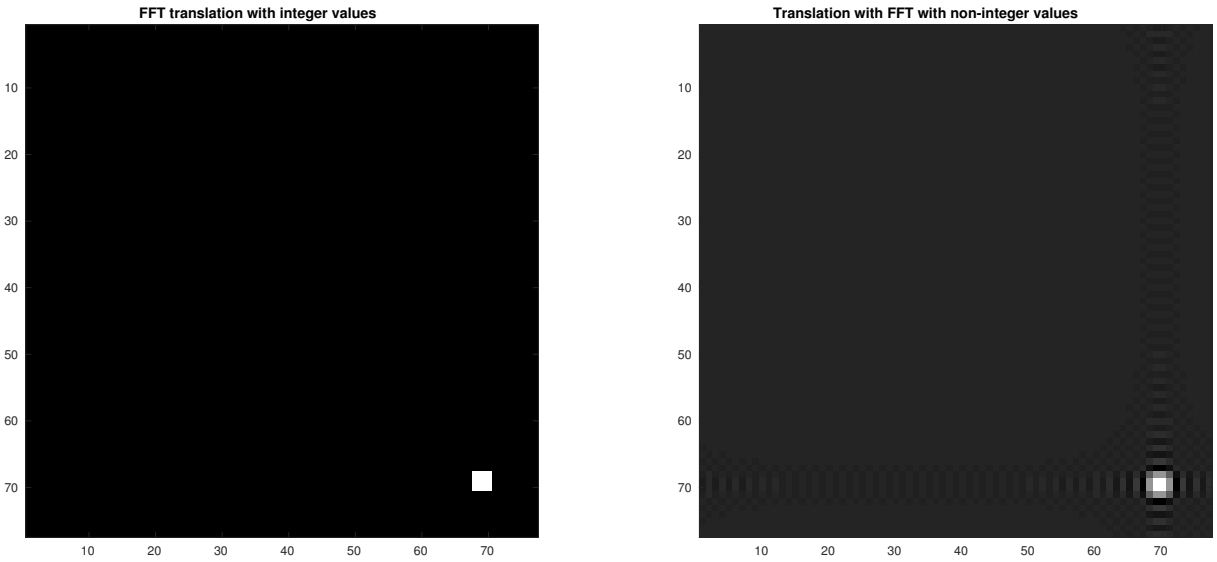

Figure 3: Comparison between integer and non-integer values when doing translations with FFT.

As we can see, the image with translation with FFT and non-integer values is distorted as we expected.

2 Procrustes transformations

2.1

The $2D$ Procrustes transformation has four free variables. This is because the Procrustes transformation only performs a translation X, a rotation R and a scaling S. Thus, the transformation is given by $T = SRX$. The scaling has one free variable, the rotation has one free variable and the translation has two free variables, hence the four free variables. The last two free variables come from the shear transformation in affine transformations.
We know that to find a single solution to n free variables, we need at least n (different) equations. As we have points (that have two unknowns), we will need at least two points, so our minimal N should be 2.

2.2

This was implemented `src/q2_2.m`. The two images loaded in and we use Matlab's `getpts` to get the coordinates of three points for both images. The points that were chosen are located near the three big intersections. We then use `cp2tform` with the coordinates for both images and the parameter 'nonreflective similarity' as the documentation states that this option is used when the image is only distorted by translation, rotation or scaling (Procrustes). Finally, we apply the transformation using Matlab's `imtransform`. In Figure 4, we see the result from attempting to transform the image `westconcordaerial.png` to the image `westconcordorthophoto.png`.

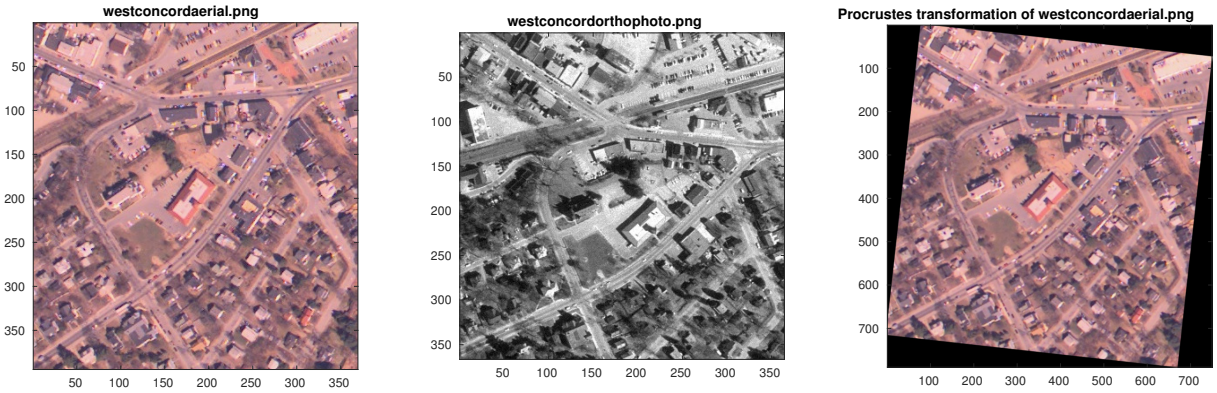

Figure 4: Two images of the same location and the Procrustes transformation of the first image to make it look like the second.

We can clearly see the image has been scaled and rotated slightly to the right to match the target image (the middle one). The translation is a bit harder to see, but the three intersections should be located in the same place as in the target. This might not be completely accurate as I think Matlab's `getpts` did not work very well with my window manager, so these could be more precise.

3 Bonus: Optimal rotation for Procrustes alignment

Not completed.

4 Affine and projective alignments

4.1

The main advantage of using homogeneous coordinates is that we not need to perform a vector addition if we want to translate the image. This means that any transformation can be written as matrix multiplication which is practical as it is faster and easier. It is also more practical in the sense that if we want to perform a sequence of transformations (scaling, rotation, translation, shearing), we would like a single matrix (precomputed) that we can apply to all input. If we have translation that requires a vector addition, we are prevented from doing this.

4.2

The plot of the two quadrilaterals given by X and Y can be seen in Figure 5 which is produced by running the file `src/q4_2.m`.

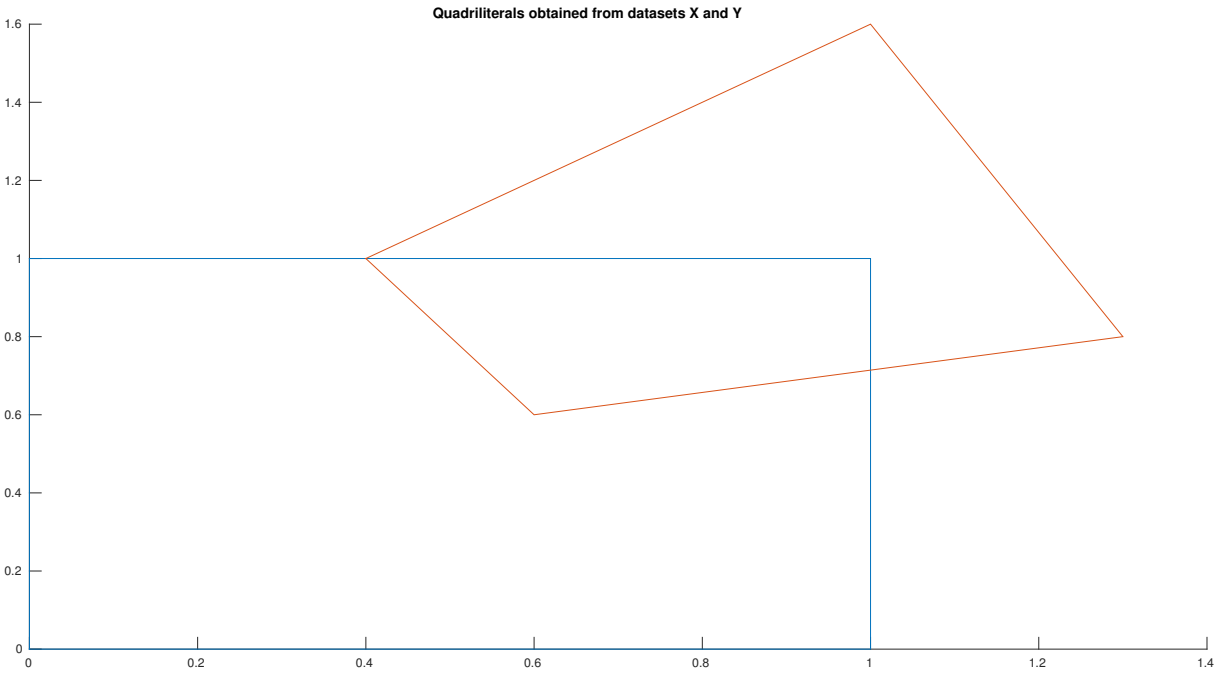

Figure 5: Two quadrilaterals given by the dataset X and Y

There cannot be a Procrustes transformation from one quadrilateral to the other. A Procrustes transformation retains its shape as we only perform scalings, rotations and translation. If we wanted to turn one of the into the other, we would need to use shear transformations as well.
With the Procrustes alignment, implemented in Matlab as `procrustes`, we can try to match one quadrilateral with the other as best as we can (the one with the smallest squared error between points). Running `src/q4_2b`, we try to scale, rotate and translate the dataset X. This results in Figure 6

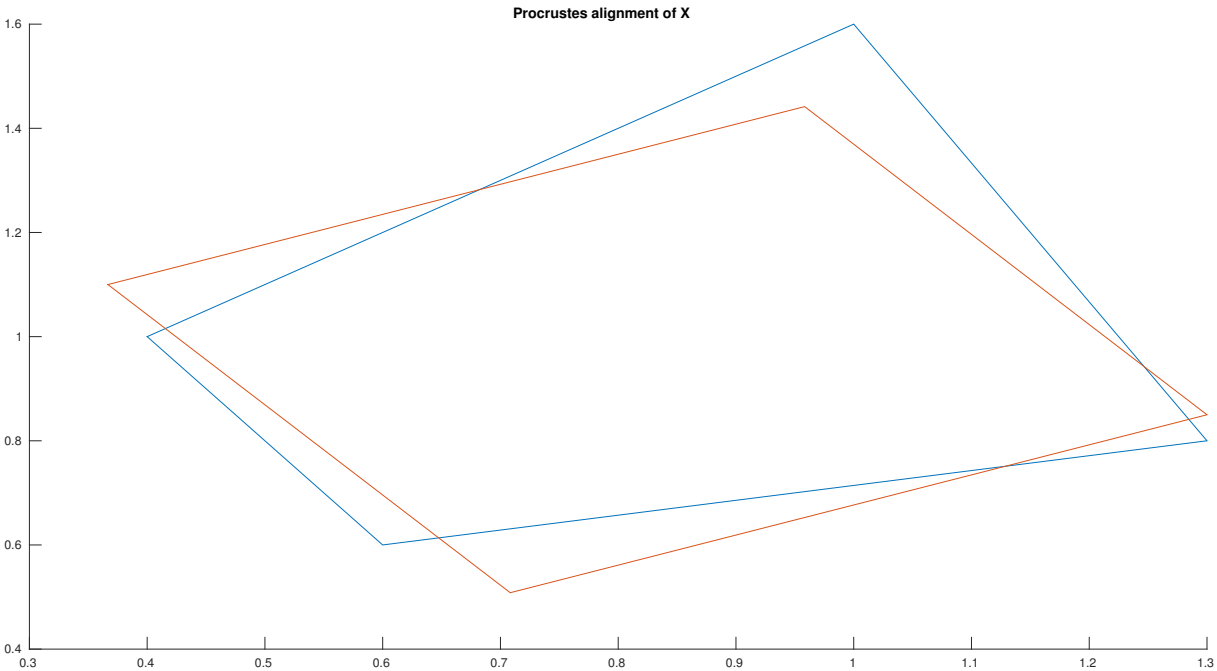

Figure 6: Procrustes alignment of dataset X with respect to Y.

This is the best transformation we can perform on Y to make it look like X. We can see that all three types of transformation have been used.

4.3

It is not possible to make an affine transformation between the two quadrilateral in Figure 5 even though we are also allowed to use shear transformations. This is due parallel lines remains parallel after an affine transformation. We can see the quadrilateral given by dataset Y has two diagonal non-parallel lines, while all diagonal lines in the quadrilateral given by X are parallel.
Figure 7, obtained by running `q4_3.m` shows the result of trying to make an affine transformation from X to Y.

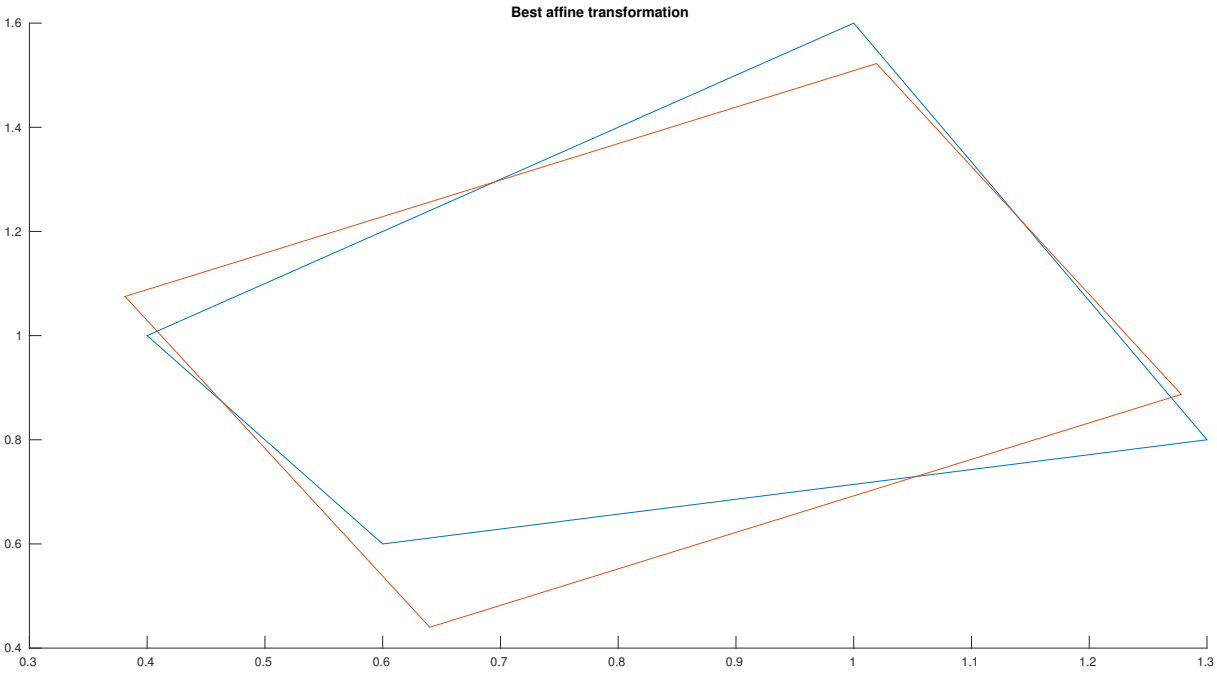

Figure 7: Affine alignment of dataset X with respect to Y.

As we can see, the transformation is not perfect, which we expected. However, it is closer to a perfect transformation than for Procrustes alignment as we are allowed to do shear transformations as well.

4.4

The projective transform has eight degrees of freedom. This means we have eight free variables and as we found in (2.1), that means we can make a perfect transformation for four points. Figure 8 shows the result of projective transform implemented in `src/q4_4.m`.

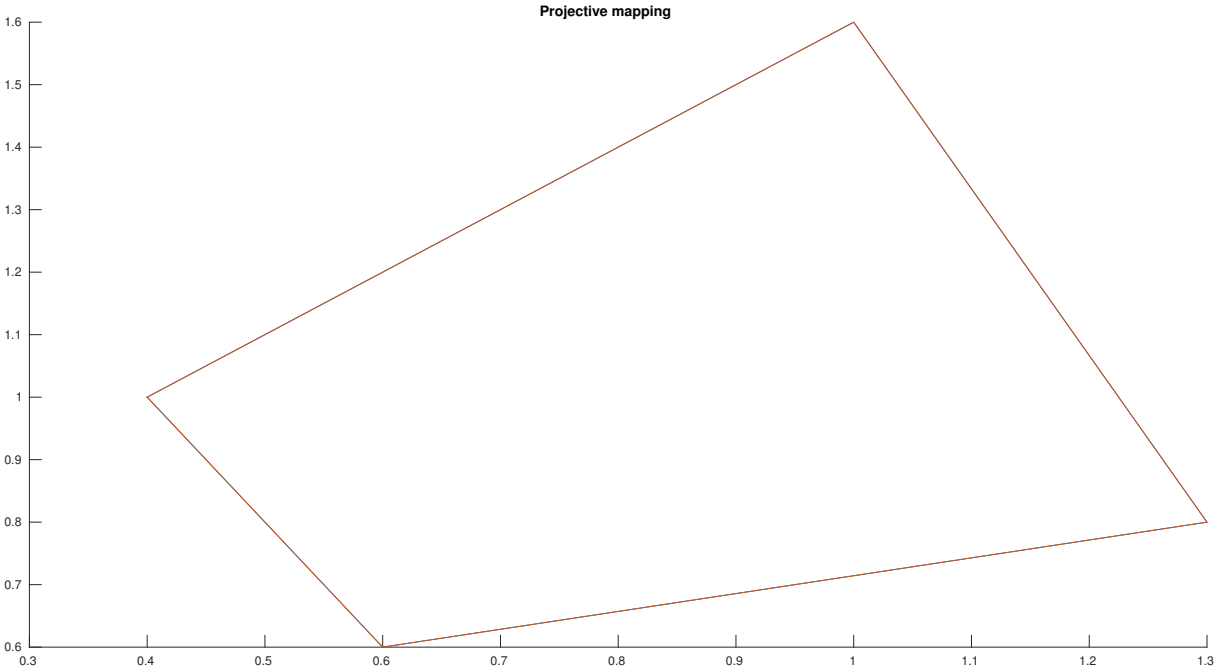

Figure 8: Affine alignment of dataset X with respect to Y.

We can only see the dataset Y, because the transformation is perfect (so there are two dataset with the same points in the figure).