

Question 5

1. Code

```
1 function [x,y, impulse, visibility] = getTranslation(I1,I2)
2     [r1,c1] = size(I1);
3     [r2,c2] = size(I2);
4     assert(r1 == r2);
5     assert(c1 == c2);
6
7     % calculate FFTs
8     F1 = fft2(I1);
9     F2 = fft2(I2);
10
11     % compute the cross-power spectrum, and the corresponding impulse
12     Prod = (F1.*conj(F2))./(abs(F1.*F2) + 1e-15);
13     prod = abs(ifft2(Prod));
14     impulse = prod/max(prod(:));
15
16     % interpret the translation from the obtained impulse
17     [~, linearIndexesOfMaxes] = max(prod(:));
18     y = rem(linearIndexesOfMaxes-1,300);
19     x = (linearIndexesOfMaxes-y-1)/300;
20     startx = x;
21     starty = y;
22     if (x > idivide(int32(c1),2))
23         x = x - c1;
24     end
25     if (y > idivide(int32(r1),2))
26         y = y - r1;
27     end
28
29     % create a yellow circle around the impulse for better visibility
30     visibility = cat(3,impulse,impulse,impulse);
31     visibility = visibility/max(visibility(:));
32     circle = zeros(20,20,3);
33     for i = 1:20
34         for j = 1:20
35             if (abs((i-10)^2 + (j-10)^2 - 81) < 20)
36                 circle(i,j,:) = [1,1,0];
37             end
38         end
39     end
40     visibility(starty-9:starty+10, startx-9:startx+10,:) = visibility(
41         starty-9:starty+10, startx-9:startx+10,:) + circle;
```

2. Image Registration without noise

In the noiseless case, the translation was obtained at $t_x = -30$ and $t_y = 70$.

Below are the corresponding plots. The 4th plot is just the 3rd plot with a marking circle added for better visibility. (Kindly zoom-in the pdf for better clarity)

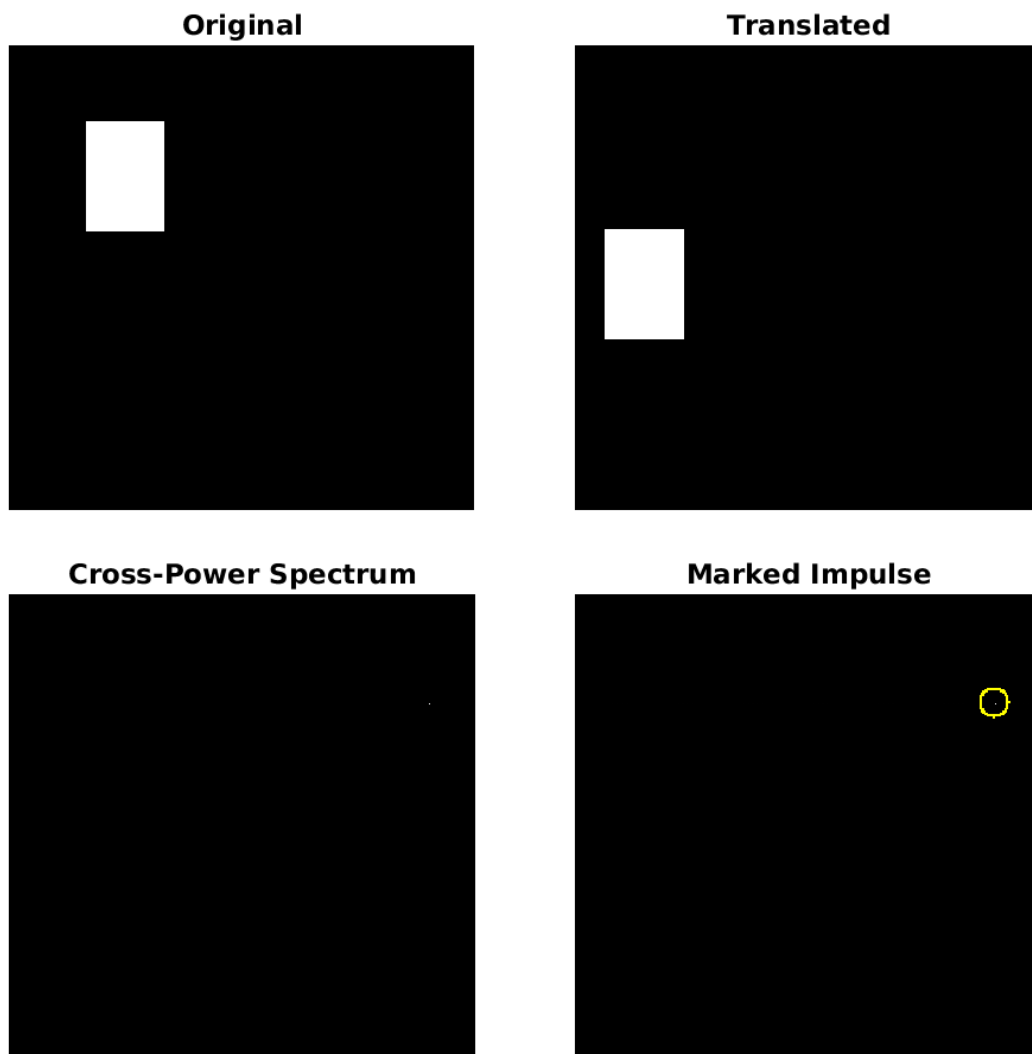


Figure 5.1: Image Registration

3. Image Registration with noise

In the noisy case, the translation was obtained at $t_x = -30$ and $t_y = 70$.

Below are the corresponding plots. The 4th plot is just the 3rd plot with a marking circle added for better visibility.

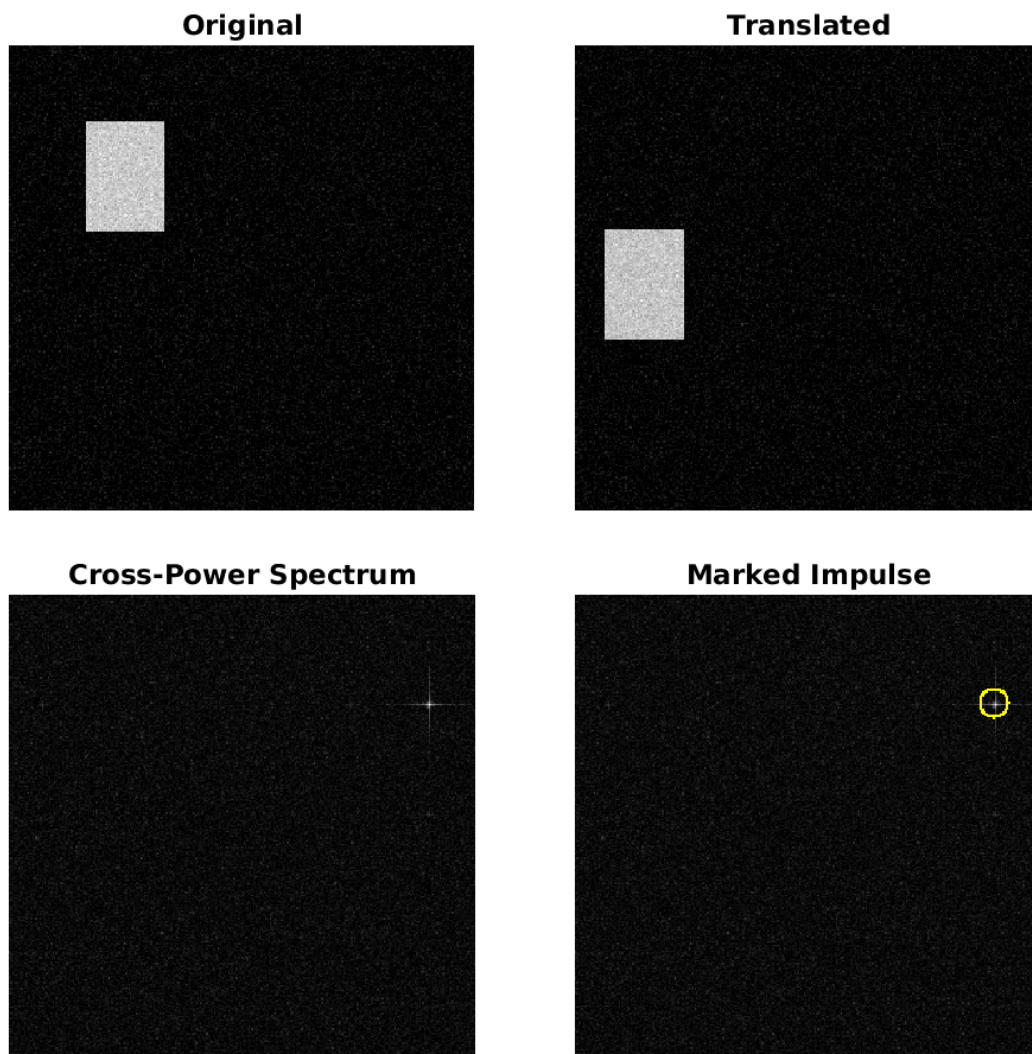


Figure 5.2: Image Registration with Noise

4. Time Complexity Analysis

For an image of size $N \times N$, the FFT algorithm takes $\mathcal{O}(n^2 \log n)$ (row-wise/column-wise FFT takes $\mathcal{O}(n \log n)$, and this is done for each of the n rows and columns). The next operation of element wise sum and division take $\mathcal{O}(n^2)$, and the inverse fourier transform again takes $\mathcal{O}(n^2 \log n)$.

Hence the overall time complexity is $\mathcal{O}(n^2 \log n) + \mathcal{O}(n^2) + \mathcal{O}(n^2 \log n) = \mathcal{O}(n^2 \log n)$

On the other hand, the pixel wise comparison will take $\mathcal{O}(n^4)$. Basically, we will compute the translated image for every possible translation combination. For a $N \times N$ image, there are a total of N^2 translation possibilities. For a given translation, the pixelwise similarity computation will again take $\mathcal{O}(n^2)$.

Hence the overall time complexity of the brute force algorithm is $\mathcal{O}(n^4)$

5. Correcting Rotation

The paper has broken this into two cases - rotation with and without scaling.

First we consider rotation without scaling. Let $f_2(x, y)$ be the translated and rotated replica of image $f_1(x, y)$, we have

$$f_2(x, y) = f_1(x \cos \theta_0 + y \sin \theta_0 - x_0, -x \sin \theta_0 + y \cos \theta_0 - y_0)$$

We know that a translation changes the phase but not the magnitude. Also, the magnitude is rotation invariant. As mentioned in the paper, using rotation property of the Fourier transform, we get,

$$F_2(\xi, \eta) = e^{-j2\pi(\xi x_0 + \eta y_0)} M_1(\xi \cos \theta_0 + \eta \sin \theta_0, -\xi \sin \theta_0 + \eta \cos \theta_0)$$

Hence the magnitudes M_1, M_2 are given as follows:

$$M_2(\xi, \eta) = M_1(\xi \cos \theta_0 + \eta \sin \theta_0, -\xi \sin \theta_0 + \eta \cos \theta_0)$$

Since we already know how to compute translation between images, we can convert the above form to polar coordinates, so that the angles are represented in the form of translation.

$$M_2(\rho, \theta) = M_1(\rho, \theta - \theta_0)$$

With the algorithm and code used in section 1, we can easily compute the rotation.

Now, if we want to consider scaling as well, we use logarithmic scale. Following a similar procedure, we get, (from eq 17 in the paper)

$$M_2(\rho, \theta) = M_1(\rho/a, \theta - \theta_0)$$

$$M_2(\log \rho, \theta) = M_1(\log \rho - \log a, \theta - \theta_0)$$

$$M_2(\xi, \theta) = M_1(\xi - d, \theta - \theta_0)$$

where,

$$\xi = \log \rho$$

$$d = \log a$$

This can be again solved using the formula and algorithm used in section I (for translation).

6. Usage of Code

- Simply run the **myMainScript.m**. This will produce and save all the required plots, and print out the translation values in both the pure and noisy case.