Image Distances and Point Transformations

Paul Vu
301169550
Feb 3rd 2017

SFU

Project Overview

In part 1 of the project:

- I took 10 mugshot images in Gray scale and calculated the Euclidean distance between the images, my results are illustrated.
- Created an MxN image where each pixel value held the mean between the 10 mugshots, and displayed the mean
- Created an MxN image where each pixel value held the Standard-Deviation between the 10 mugshots, and displayed the standard-deviation image
- Calculated Penrose distance between each mugshot with respect to the mean and displayed it
- Calculated Penrose distance between each mugshot with respect to each other and displayed it
- I then took 3 images, a laptop, backpack and classroom and calculated the Penrose distance to the mean of my mugshot images

In part 2 of the project:

- I examined the FFT2 Function in matlab

10 Mugshot Photos in GrayScale

These 10 Mugshot photos were used throughout the assignment.

Figure1: Mugshots 1 through 5 from top left to top right, Mugshots 6 through 10 from bottom left to bottom right

Mugshot 1 Euclidean Distance:

Mugshot 1 is my reference image, and it’s Euclidean distance was measured against the other 10 mugshots. As expected, the Euclidean distance with respect to itself is zero since the difference between these images is zero for all pixels. The largest distance was found when measured against mugshot 9 which could be the case since in mugshot 9 I stretch my face more then any other mugshot, so there is a big difference compared to the reference image, resulting in a larger Euclidean distance. My smallest Euclidean distance is with respect to mugshot 2, here I change my facial expression lightly (since I didn’t make much effort to change my face).

eucArrMug1

1x10 double

	1	2	3	4	5	6	7	8	9	10
1	0	7.2373	10.2720	10.9750	9.3858	10.1800	9.1429	11.1154	11.1287	10.5040

Figure 2: Mugshot 1 Euclidean Distance

Mugshot 2 Euclidean Distance:

Closest Euclidean Distance: Mugshot 1
Furthest Euclidean Distance: Mugshot 8

Note* As expected the Euclidean distance from mugshot 1 w.r.t mugshot 2 is the same distance as mugshot 2 w.r.t mugshot 1

eucArrMug1 eucArrMug3 eucArrMug2

1x10 double

	1	2	3	4	5	6	7	8	9	10
1	7.2373	0	8.8994	10.3278	8.5291	8.9826	8.2513	10.2497	10.1522	9.6074

Figure 3: Mugshot 2 Euclidean Distance

Euclidean Distance of all mugshots:

The results are shown in the figure below for all Euclidean distances of all 10 mugshots. We can see that indeed Euclidean Distance of (A,B) is indeed the same as Euclidean Distance of (B,A) since we can see the matrix is symmetric on the diagonal

eucArrMug3 | row | row1

10x10 double

	1	2	3	4	5	6	7	8	9	10
1	0	7.2373	10.2720	10.9750	9.3858	10.1800	9.1429	11.1154	11.1287	10.5040
2	7.2373	0	8.8994	10.3278	8.5291	8.9826	8.2513	10.2497	10.1522	9.6074
3	10.2720	8.8994	0	9.1382	8.0405	8.5401	9.4106	8.3158	8.2838	8.7279
4	10.9750	10.3278	9.1382	0	8.9115	10.1536	10.3372	9.0436	10.2299	10.2395
5	9.3858	8.5291	8.0405	8.9115	0	7.8532	7.7689	8.0355	8.2461	8.0186
6	10.1800	8.9826	8.5401	10.1536	7.8532	0	8.1592	8.5870	7.9478	8.4884
7	9.1429	8.2513	9.4106	10.3372	7.7689	8.1592	0	8.9427	8.8288	7.5999
8	11.1154	10.2497	8.3158	9.0436	8.0355	8.5870	8.9427	0	6.9021	8.0317
9	11.1287	10.1522	8.2838	10.2299	8.2461	7.9478	8.8288	6.9021	0	7.5371
10	10.5040	9.6074	8.7279	10.2395	8.0186	8.4884	7.5999	8.0317	7.5371	0

Figure 4: Euclidian distance of 10 mugshots

Mean of 10 mugshots

The mean of the 10 mugshots is depicted below.

- Here we can see that because my facial expressions were the only things moving from image to image, and that my head and body were in relatively the same position in each image, we expect to see that my body and head's mean would be consistent since the mean of these pixel values are relatively unchanged.
- On the other hand, my facial expressions changed in every photo (particularly my mouth), and the values at each pixel is constantly changing from image to image. Therefore we expect that the mean of all these values would distort my face (specifically my mouth) and cause a blur in my face since these values are dynamically changing

Figure5: Mean image of 10 mugshots

Standard Deviation of 10 mugshots

The standard deviation of the 10 mugshots is depicted below.

- Here we can see that because I chose a gray scale image, and that my facial expressions changed very minimally between each image, we expect that the deviation of each pixel from the mean to be zero, or near zero. We can see that this is the case, and as a result plotting these values results in a black image. These values are also shown as near zero for confirmation.

	212	213	214	215
147	2.1940e-...	1.9907e-...	2.2555e-...	2.1206e-...
148	1.3260e-...	1.8523e-...	1.8608e-...	2.3666e-...
149	1.3482e-...	1.5806e-...	1.8950e-...	2.3940e-...
150	1.4781e-...	1.5054e-...	1.6216e-...	1.5738e-...
151	1.6079e-...	1.6216e-...	1.4627e-...	1.1090e-...
152	1.8950e-...	1.7019e-...	1.4781e-...	1.0868e-...
153	1.9839e-...	5.4150e-...	3.2466e-...	2.3581e-...
154	6.8418e-...	1.4053e-...	6.9785e-...	2.3324e-...
155	6.3975e-...	4.1420e-...	5.3928e-...	2.9886e-...
156	3.2543e-...	4.2734e-...	3.4271e-...	9.5075e-...
157	0.0040	0.0046	0.0036	8.9805e-...
158	0.0061	0.0062	0.0062	0.0058
159	0.0024	0.0027	0.0040	0.0045
160	1.6207e-...	6.6703e-...	0.0021	0.0031

Figure6: Standard Deviation of 10 images, with sample vlaues

Penrose Distance Results

From class, we know that the Penrose distance is the difference between image intensities at each pixel, squared divided by the standard deviation and square rooted. There is a flaw in this approach, as we can see a case where standard deviation is 0, this would throw our Penrose distance to infinity. To address this issue, I chose to divide by a threshold value rather than a standard deviation. This threshold was 0.001 above my standard deviation value. I scaled all my values up by 0.001 such that I was only interested in variances above this value. The results are shown on the next slides…

$$(Penrose\ Distance\ Theory)^2 = \sum_{i=0}^{MxN} \frac{(xi - ji)^2}{\sigma}$$

$$(Penrose\ Distance\ Practical)^2 = \sum_{i=0}^{MxN} \frac{(xi - ji)^2}{\sigma + \mathrm{T}}$$

Where T = threshold = 0.01
σ = standard deviation
xi = Pixel intensity value in image 1
xj = Pixel intensity value in image 2

Eq. 1: Penrose theoretical and practical equations

Penrose Distance Results

The Penrose Distance from each image to the mean and w.r.t each other is displayed below. Interestingly the Penrose distance is smaller between each image w.r.t the mean in comparison to the mugshot w.r.t each other. For example the distances measured for mugshot 1 w.r.t all the images says mugshot 2 is most similar since it's value is the smallest, 35.39. However, the distance to the mean is 33.903 meaning it's more similar to the mean Img. This is true for all the images.

stdImg2 | penArr | eucArr | penArrMugMean

1x10 double

	1	2	3	4	5	6	7	8	9	10
1	33.9039	28.4111	27.0796	36.2059	23.9541	30.2771	26.0562	29.2699	29.9764	29.0960

Figure 7: Penrose distance of each mugshots (1-10) w.r.t the mean

10x10 double

	1	2	3	4	5	6	7	8	9	10
1	0	35.3989	46.3171	52.6698	43.2811	48.7885	43.9132	51.6947	52.1263	50.2295
2	35.3989	0	39.8685	49.8482	39.2727	44.0116	39.2357	47.3239	46.9803	45.2891
3	46.3171	39.8685	0	45.8900	37.6309	43.1060	42.5201	41.5982	41.2254	41.8622
4	52.6698	49.8482	45.8900	0	42.9074	46.7056	49.3885	48.7124	53.7208	52.8721
5	43.2811	39.2727	37.6309	42.9074	0	38.3437	36.5613	40.2216	42.1227	40.4763
6	48.7885	44.0116	43.1060	46.7056	38.3437	0	40.4789	46.5020	46.0850	46.7678
7	43.9132	39.2357	42.5201	49.3885	36.5613	40.4789	0	42.0049	41.7881	36.8844
8	51.6947	47.3239	41.5982	48.7124	40.2216	46.5020	42.0049	0	35.0253	39.4056
9	52.1263	46.9803	41.2254	53.7208	42.1227	46.0850	41.7881	35.0253	0	37.1089
10	50.2295	45.2891	41.8622	52.8721	40.4763	46.7678	36.8844	39.4056	37.1089	0

Figure 8: Penrose distance of each mugshots w.r.t each other

Penrose Distance w.r.t random objects

The penrose distance is now calculated between the mean of my mugshots to a classroom(1), notebook(2) and bag(3). We can see these values are huge, about 5-7 times larger than my penrose distances of my 10 mugshots to the mean. This shows that the distance calculated with reference can identify that these objects are very different from my mugshots.

stdImg2 | penArr | eucArr | penArrMugMean

1x10 double

	1	2	3	4	5	6	7	8	9	10
1	33.9039	28.4111	27.0796	36.2059	23.9541	30.2771	26.0562	29.2699	29.9764	29.0960

Figure 9: Penrose distance of each mugshots (1-10) w.r.t the mean

penMeanToObj

1x3 double

	1	2	3
1	158.2139	165.0826	170.2354

Figure 10: Penrose w.r.t objects

FFT2 Matlab: Results

For part 2 of the assignment, I created a 100x100 Matrix and set all the pixels to black (0). I then set a 50x20 matrix in the center of this matrix to be 1 making it a white rect-like function with amplitude 50. I then compute the FFT of this matrix using matlab's FFT2 function. I then display the image before and after log transformation. My results are shown below: (note this code is adapted from Lecture 8)

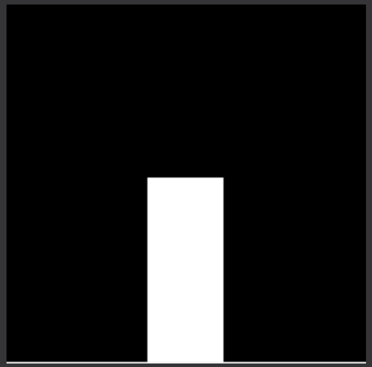
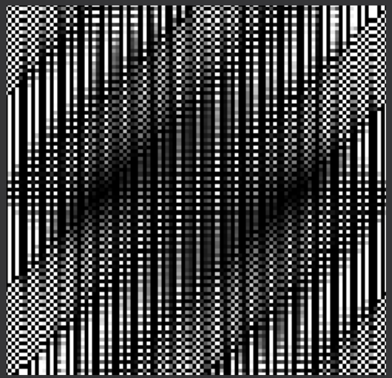
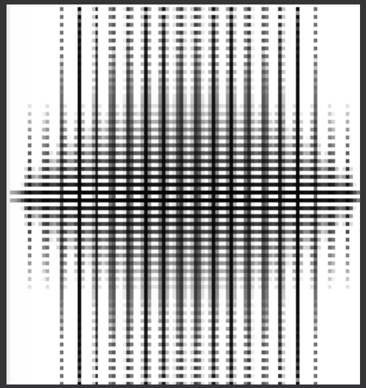


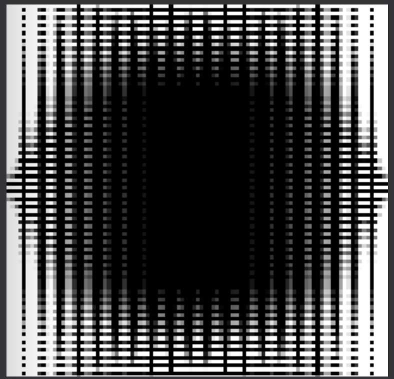
Image : I



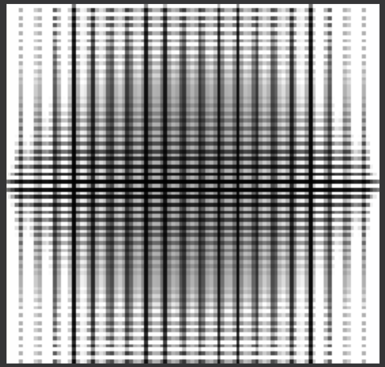
FFT of I



Log(abs(I))



Log(FFT(I))



Log(abs(I)+1)

Figure 11: Illustrates the FFT and log transform of a rect

Expected Results…

Below is the expected result of a Fourier transform on a Rect function.

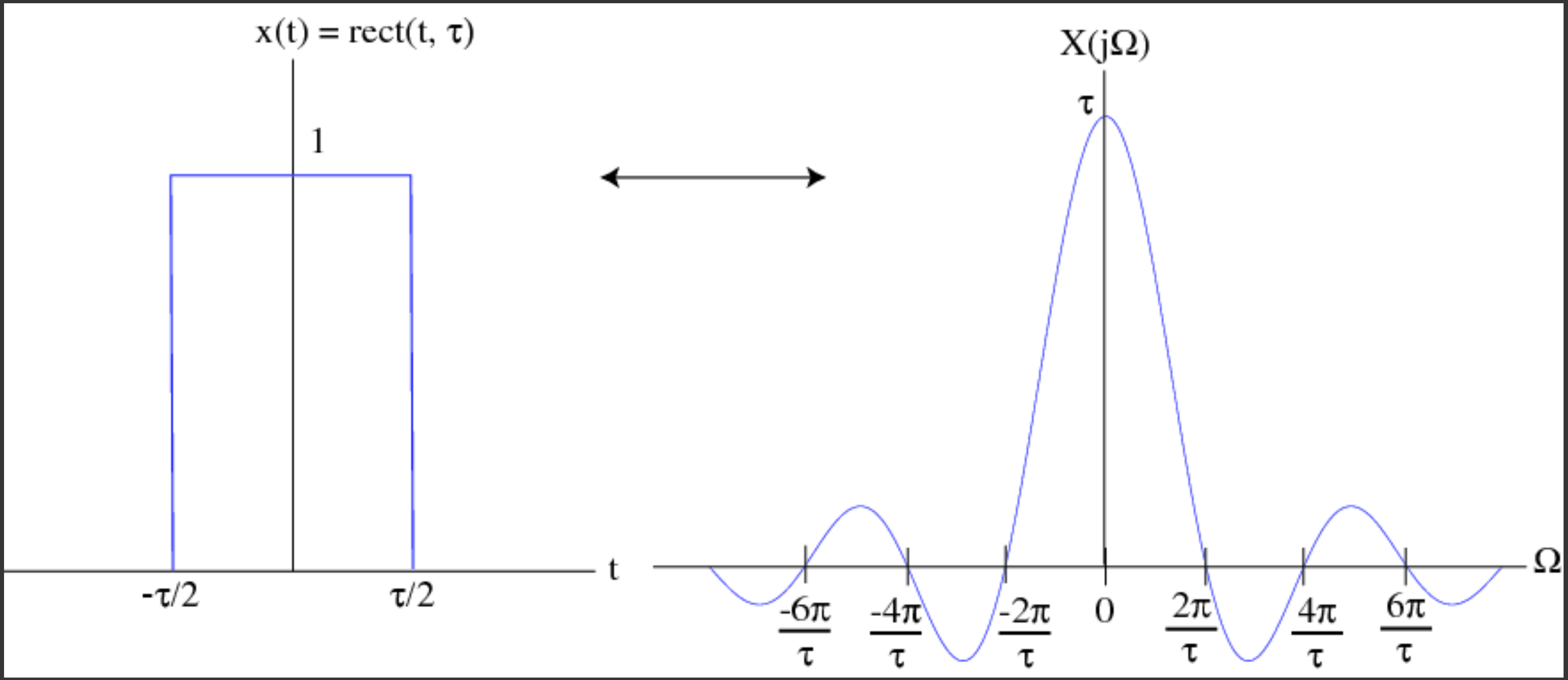


Figure 12: Expected results is a sinc function after a fourier transform of a rect

Source: http://archive.cnx.org/resources/48cee6149366a369f569290dcf9e10aee8d02b64/sinc.png

FFT2 Matlab: Analysis

Ideally, the fourier transform of a rect function should look like a sinc(f), where a X(f) is at it’s peak where the rect was centered. However this was not observe. This is because the image is actually broken into 4 corner pieces by matlab which makes the sinc graph look ovular. By applying the fftshift function, we get the expected result. The low frequency components are clearly on the edges of the image, while the high frequency values are shown by the dark black-grey components in the center. The DC value of the image is at the corners of the image.

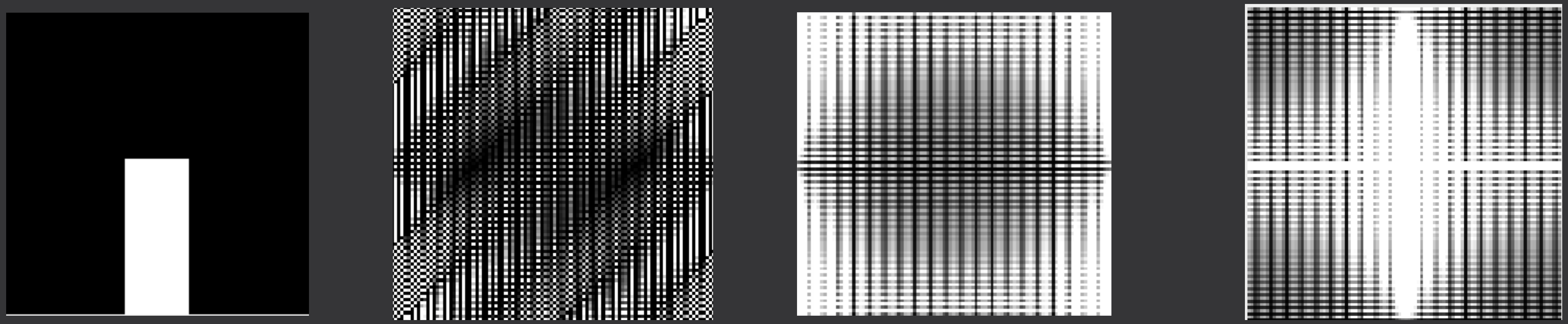


Image : I

F = FFT of I

Log(abs(F)+1)

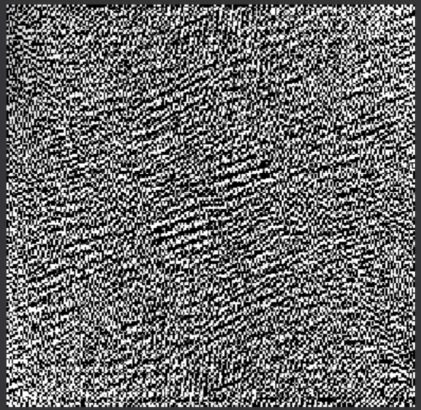
Log(abs(fftshift(F))+1)

Figure 13: Illustrates the FFT and log transform of Image I

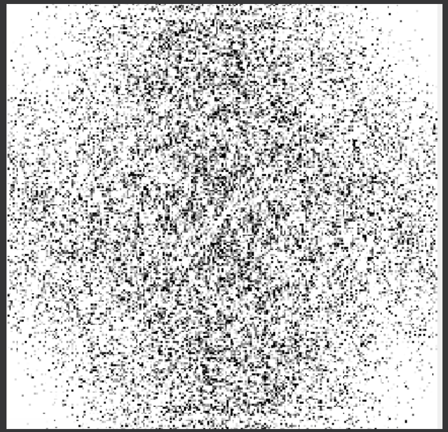
Mugshot 1 FFT2 Transformation Spectrum after Log Transform

I applied the FFT transform to my own mugshot 1 and the results are shown below. Very interesting.

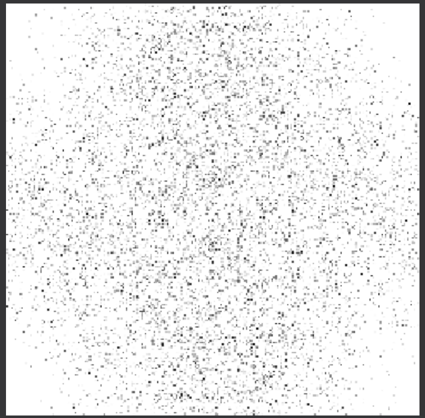
Image : I



FFT of I



Log(abs(FFT(I)))



Log(abs(FFT)+1)

Figure 14: Illustrates the FFT and log transform of Mugshot 1