CS543 Assignment 2

**Your Name:** <name>Kaylin Chen

**Your NetId:** <netid>tc49

# Part 1 Fourier-based Alignment:

You will provide the following for each of the six low-resolution and three high-resolution images:

* Final aligned output image
* Displacements for color channels
* Inverse Fourier transform output visualization for ***both*** channel alignments ***without*** preprocessing
* Inverse Fourier transform output visualization for ***both*** channel alignments ***with*** any sharpening or filter-based preprocessing you applied to color channels

You will provide the following as further discussion overall:

* Discussion of any preprocessing you used on the color channels to improve alignment and how it changed the outputs
* Measurement of Fourier-based alignment runtime for high-resolution images (you can use the python time module again). How does the runtime of the Fourier-based alignment compare to the basic and multiscale alignment you used in Assignment 1?

## A: Channel Offsets

Replace <C1>, <C2>, <C3> appropriately with B, G, R depending on which you use as the base channel. Provide offsets in the **original image coordinates** (after the image has been divided into three equal parts corresponding to each channel) and be sure to account for any cropping or resizing you performed.

Low-resolution images (using channel <C1> as base channel):

|  |  |  |
| --- | --- | --- |
| Image | <C2> (h,w) offset | <C3> (h,w) offset |
| 00125v.jpg |  |  |
| 00149v.jpg |  |  |
| 00153v.jpg |  |  |
| 00351v.jpg |  |  |
| 00398v.jpg |  |  |
| 01112v.jpg |  |  |

High-resolution images (using channel <C1> as base channel):

|  |  |  |
| --- | --- | --- |
| Image | <C2> (h,w) offset | <C3> (h,w) offset |
| 01047u.tif |  |  |
| 01657u.tif |  |  |
| 01861a.tif |  |  |

## B: Output Visualizations

For each image, insert 5 outputs total (aligned image + 4 inverse Fourier transform visualizations) as described above. When you insert these outputs be sure to clearly label the inverse Fourier transform visualizations (e.g. “G to B alignment without preprocessing”).

### 00125v.jpg

### 00149v.jpg

### 00153v.jpg

### 00351v.jpg

### 00398v.jpg

### 01112v.jpg

### 01047u.tif

### 01657u.tif

### 01861a.tif

## C: Discussion and Runtime Comparison

# Part 2 Scale-Space Blob Detection:

You will provide the following for ***8 different examples***(4 provided, 4 of your own):

● original image

● output of your circle detector on the image

● running time for the "efficient" implementation on this image

● running time for the "inefficient" implementation on this image

You will provide the following as further discussion overall:

● Explanation of any "interesting" implementation choices that you made.

● Discussion of optimal parameter values or ones you have tried

### Example 1:

### Example 2:

### Example 3:

### Example 4:

### Example 5:

### Example 6:

### Example 7:

### Example 8:

## Discussion:

# Bonus:

## Blob-Detection Extra Credit

● Discussion and results of any extensions or bonus features you have implemented for Blob-Detection