RGB Channel Alignment for an Image

In this assignment, 4 tasks were to be performed. They have been briefly described below:

Task 1:
Separate the RGB channels into three single channel images and merge them to form an RGB image. For this task alignment was not required. In order to achieve this an assumption that all 3 channels are of the same size was made. With this assumption, the image was divided into 3 equal index intervals. Also, since the order in which the image channels were arranged had been given, the channels were arranged accordingly in the final image. Since no alignment was performed the images are highly blurred.

Task 2:
For this task, the alignment was to be done using a vanilla SSD-based implementation. For this task, we consider red as the base image and align green and blue over it. We first start with a window that is just a shift by n pixels in both x and y directions. This window is then passed over the base channel to identify the region which has the least SSD. The minimum distance is updated every time a better alignment is identified. The image is then shifted by the resultant offset to achieve a finer image.

Task 3:
Using the structure of this approach, we similarly design our normalized correlation-based alignment. The difference here is that instead of distance, we get a similarity measure(i.e. Higher the value more similar the pixels are). We compute the normalized correlation coefficient for this purpose. Over a given window, we first normalize the pixel values with respect to their mean. Then the resultant channel window matrix is divided by the standard deviation. The resultant value of these channel windows is then multiplied to the base channel. The result of this operation is then averaged to obtain the normalized correlation coefficient. From this task onwards, blue has been used as the base and align other channels over it.

Initially, however, a different approach was attempted. Instead of using the coefficient and comparing it against a max value, an attempt was made to identify the pixel position of the maximum value within the window and directly use that to compute the offset. However, after multiple attempts, I could not get satisfactory results and therefore decided to take this alternative approach.

Task 4:
We divide this task into 2 sub-tasks:

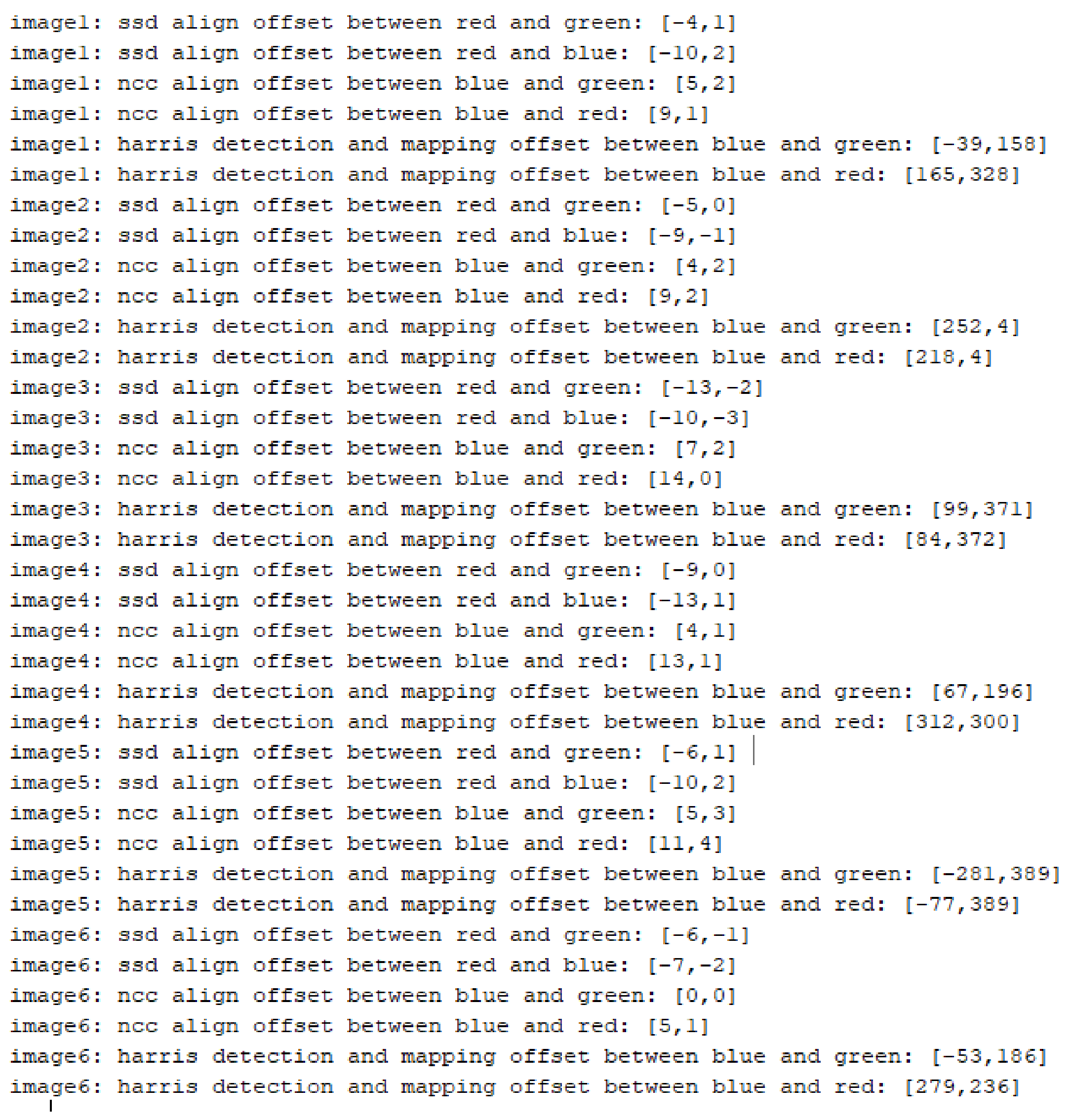
1. Detecting Features using harris corner detection.
2. Performing RANSAC on feature points to identify best matching points and computing the offset from their positions.

For the first sub-task, a simplistic approach is used. First, a Sobel filter is used to identify the derivative of the image in x-direction using convolution. Similarly, we compute the derivative of the image in the y-direction using the rotated filter. The partial derivative along xy is given by the product of the 2 derivatives. We then compute the second-order derivatives using a gaussian filter. These values then form the H i.e. gradient matrix of the image which is then used to compute the difference between the Determinant and trace of the matrix squared times an experimental coefficient k at each pixel position. The positions which have values greater than a pre-defined threshold are then used to identify the top feature points. For verification purposes only, I created a test function to compare feature points through visual output for my implementation and for the in-built implementation for the corner library function defined in the image processing toolbox.

For the second sub-task, we implement RANSAC to identify the best match. For this assignment, a very simple approach is being used. Fork iterations, I select a point at random from my base image. I then compare the corresponding value against each point from the channel to be aligned. If the distance between them is less than a threshold, then the point is assumed to be a good fit.

At this moment, the results of this approach are not satisfactory. I believe that doing this over a moving window and updating the best match should rectify this issue.

The below image shows offsets obtained for each of the 3 alignments align with which channels were aligned:



```
image1: ssd align offset between red and green: [-4,1]
image1: ssd align offset between red and blue: [-10,2]
image1: ncc align offset between blue and green: [5,2]
image1: ncc align offset between blue and red: [9,1]
image1: harris detection and mapping offset between blue and green: [-39,158]
image1: harris detection and mapping offset between blue and red: [165,328]
image2: ssd align offset between red and green: [-5,0]
image2: ssd align offset between red and blue: [-9,-1]
image2: ncc align offset between blue and green: [4,2]
image2: ncc align offset between blue and red: [9,2]
image2: harris detection and mapping offset between blue and green: [252,4]
image2: harris detection and mapping offset between blue and red: [218,4]
image3: ssd align offset between red and green: [-13,-2]
image3: ssd align offset between red and blue: [-10,-3]
image3: ncc align offset between blue and green: [7,2]
image3: ncc align offset between blue and red: [14,0]
image3: harris detection and mapping offset between blue and green: [99,371]
image3: harris detection and mapping offset between blue and red: [84,372]
image4: ssd align offset between red and green: [-9,0]
image4: ssd align offset between red and blue: [-13,1]
image4: ncc align offset between blue and green: [4,1]
image4: ncc align offset between blue and red: [13,1]
image4: harris detection and mapping offset between blue and green: [67,196]
image4: harris detection and mapping offset between blue and red: [312,300]
image5: ssd align offset between red and green: [-6,1]
image5: ssd align offset between red and blue: [-10,2]
image5: ncc align offset between blue and green: [5,3]
image5: ncc align offset between blue and red: [11,4]
image5: harris detection and mapping offset between blue and green: [-281,389]
image5: harris detection and mapping offset between blue and red: [-77,389]
image6: ssd align offset between red and green: [-6,-1]
image6: ssd align offset between red and blue: [-7,-2]
image6: ncc align offset between blue and green: [0,0]
image6: ncc align offset between blue and red: [5,1]
image6: harris detection and mapping offset between blue and green: [-53,186]
image6: harris detection and mapping offset between blue and red: [279,236]
```