**Problem 1.1**  
Proving existence of H in two image points of 2 cameras whose projection matrices are P1 and P2. X1 and X2 are points in homogenous coordinates.

X1 ≡ H\*X2

Let’s define a point in the homogenous coordinate,

O (Xi, Yi, Zi,1)

Now,

The image of O in P1 and P2 will be

X1 = P1\*O and X2 = P2\*O

∴ (P1)-1\*X1 =(P2)-1\*X2

So, X1 = X2\*P1\*(P2)-1

Now,

P1\*(P2)-1 = H

∴ X ≡ H\*X2

∴ We can say that the equation is correct to a scaling factor -> X1 ≡ H\*X2

**Problem 1.2.1**

Total degrees of freedom are the number of elements in the matrix, from which one unit is deducted to account for scaling factor.

∴ Degrees of freedom of **h = 8**

**Problem 1.2.2**  
To solve h, we need a total of 8 points,

∴ **4 point pairs** will be required.

**Problem 1.2.3**  
Deriving Ai

Given X1i = H\*X2i -------------------> ①

Say, X1 = and X2 =

Now, H =

­

Using Scale factor λ, we can rewrite ① as,

= \*

Let’s, put λ = 1 and divide 1st and 2nd row with the 3rd.

We get,

-h11\*C-h12\*d-h13+(h31\*c+h32\*d+h33) \*a = 0

and

-h21\*C-h22\*d-h23+(h31\*c+h32\*d+h33) \*b = 0

The above 2 equations can be now expressed in the matrix form as

Ai\*h=0

Where,

A =

And, h =

**Problem 1.2.4**

Trivial Solution for h will be

H=

Here, size of h is 9\*1.

A is not a full rack matrix. Since only 4 points are required to calculate h, A is an 8\*9 Matrix.

Since the 8 columns are linearly independent, 8 out of the 9 vectors will be linearly dependent as well, so one of the eigen values will be zero.

The eigen vector corresponding to this 0 eigen value will map to A\*h = 0

**Problem 1.4.1**

We have

For 1st camera: X1 = k1X

For 2nd camera: X1 = k2X

Here,

I: Identity Matrix

R: Rotation Matrix

X is a point in the 3D Space

X =

So,

X1 = k1X

X1 = k1 = k1\*

∴ X =

Similarly, on substituting tis value of X in the equation for 2nd Camera,

X1 = k2X

X1 = k2 = k2

∴ X2 = k1\*R\*K1-1\*X1

∴ on comparing with X1 = H\*X2

We get,

H=K2\*R\*k1-1

On H=K1\*R-1 \*K2-1

Therefore, there exists a homography H such that it satisfies

X1 ≡ H\*X2

**Problem 1.4.2**

We know,

H=K\*R\*k-1

And R =

So, R-1 =

Now, H2 = K\*R()\*k-1 \*K\*R()\*k-1

H2 = K\*

H2 = K\*

We Know,

And

H2 = k \*

∴H2 = K\*R()\*k-1

H2 is the homography corresponding to a rotation of .

**Problem 1.4.3**

Planar homography is not sufficient since its repeated pattern handling is inefficient. It works well only between

arbitrary image and the viewpoint. If the scene/image is planar, which is not how it is in actuality.

Also, between subregion of 2 images, there exists different homographies, corresponding to the viewpoint on subregion of same planar.

**Problem 1.4.4**  
Consider a 3D line with Coordinates as , and

Now,

Perception matrix P =

On multiplying P with line,

X =

X =

From the x value, we can see that line is projected to xy plane and is still preserved.

**Problem 2.1.1**

**FAST detector** and Harris corner detector are used to detect corners in a scene based on the change in intensity value. Fast detector samples a pixel and considers a 16 pixel circle around it, a threshold is defined on which change in intensity based on which pixel intensities out of four pixels on the axis are checked. The decision of the corner is made based on if the intensity of chosen pixel is above or below the threshold.

On the other hand, the **Harris Corner Detector** is a corner detection operator that uses a window around each pixel and uses sum of squared difference of the pixel values when the window is shifted by a small amount in any direction. It takes differential of the corner score into account with reference to direction directly and hence is more accurate in its detection.

**Problem 2.1.2**

The filter banks seen in the lectures requires a lot of computations to find binary strings whereas by utilizing less memory, faster matching and higher recognition rate, BRIEF Descriptor is an easy way to get binary descriptors. BRIEF is very fast both to build and to match. It does that by comparing intensities of the selected location pairs from the part of image with smooth patch.

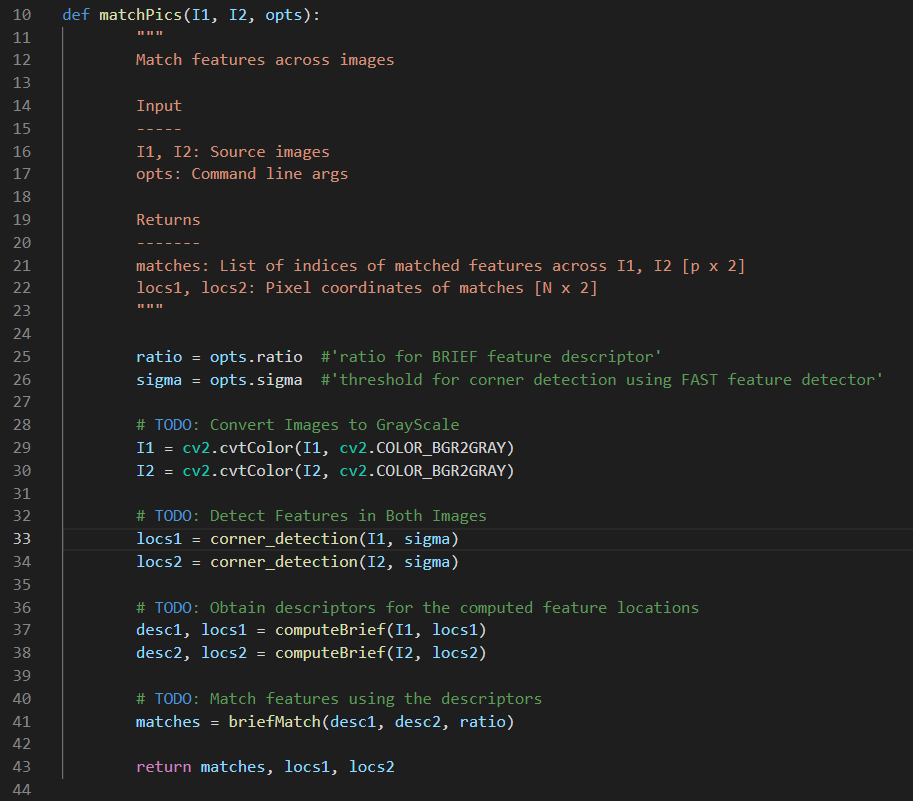
**Problem 2.1.3**

Binary strings in BRIEF Descriptor that are used to match features can use Hamming distance as a metric for computing the match.

In Nearest Neighbor, make two sets. From the first image, pick N interest points and put them in first set. Then from ground truth data, deduce the corresponding points in the other, and put it in 2nd set. After computing the 2N associated descriptors, for each point in first set, use Nearest Neighbor to find the second one and call it a match.

Hamming distance, as compared to Euclidean distance provides better speed-up in measuring distance because finding hamming distance is just applying XOR and bit count, which are very fast in modern CPUs with SSE instructions.

**Problem 2.1.4**Code Snippet: matchPics



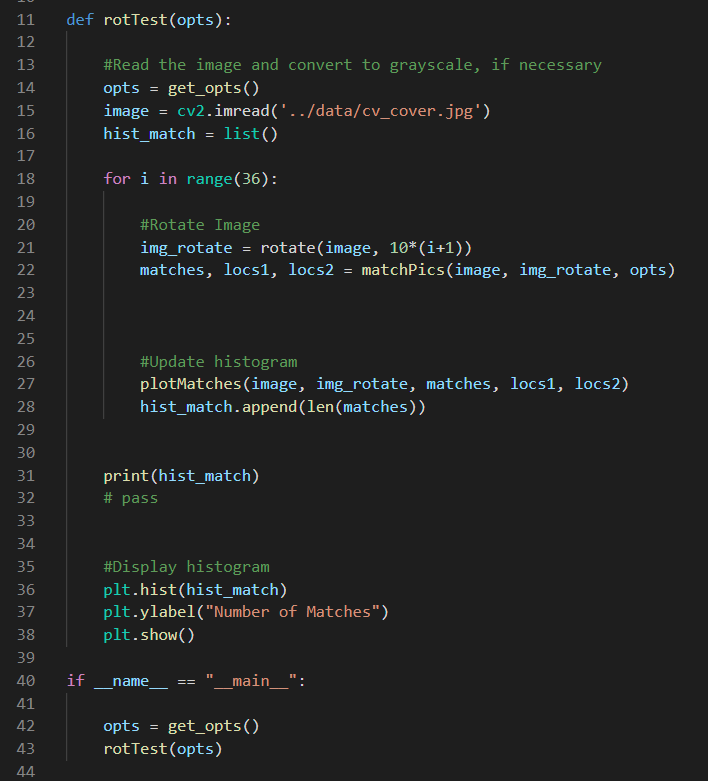
**Problem 2.1.5**

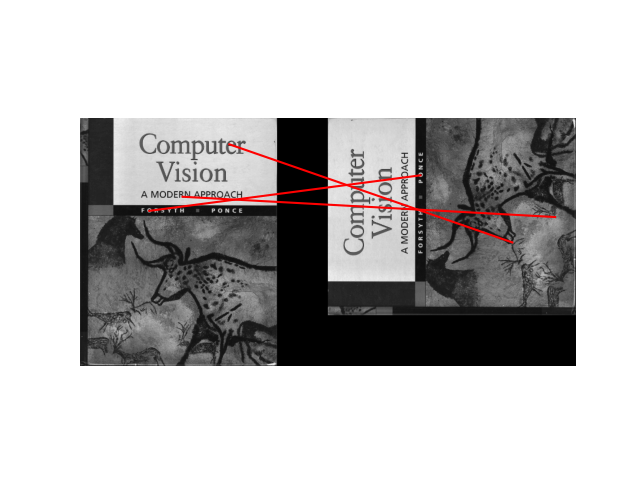
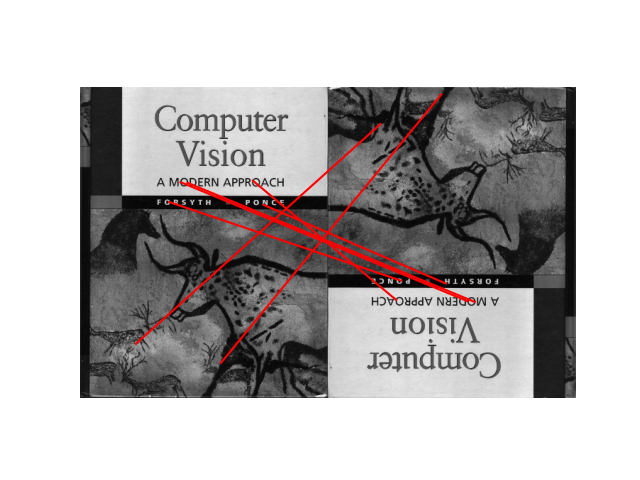
|  |  |
| --- | --- |
| **Initial Vanilla Parameters** | |
| **Sigma = 0.1 Ratio = 0.65** | **Sigma = 0.15 Ratio = 0.8** |
| **Sigma = 0.2 Ratio = 0.6** | **Sigma = 0.2 Ratio = 0.7** |
| **Sigma = 0.2 Ratio = 1** | **Sigma = 0.4 Ratio = 0.75** |

From the above figures it can be seen that at lower value of sigma there are more matches outside the book. Also, as the value ratio is lowered the number of matches reduces. And when the Ratio increases, even if sigma varies, that is increases or decreases, the no of matches between the images increases greatly. Another conclusion could be that ratio acts as a threshold of difference between the points, such that when ratio is high, points which are a mismatch are also considered as a match, but when the ratio is low, the only similar points are matched, which should be the case.

**Problem 2.1.6**

Code Snippet – briefRotTest

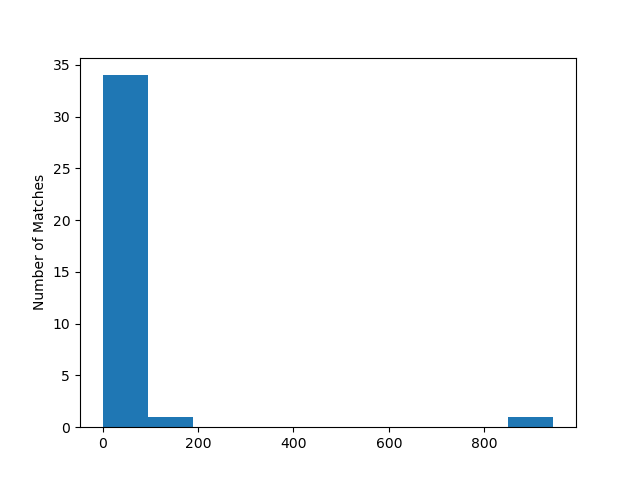


Orientation: 90° Orientation: 180°

Orientation: 220° Orientation: 360°

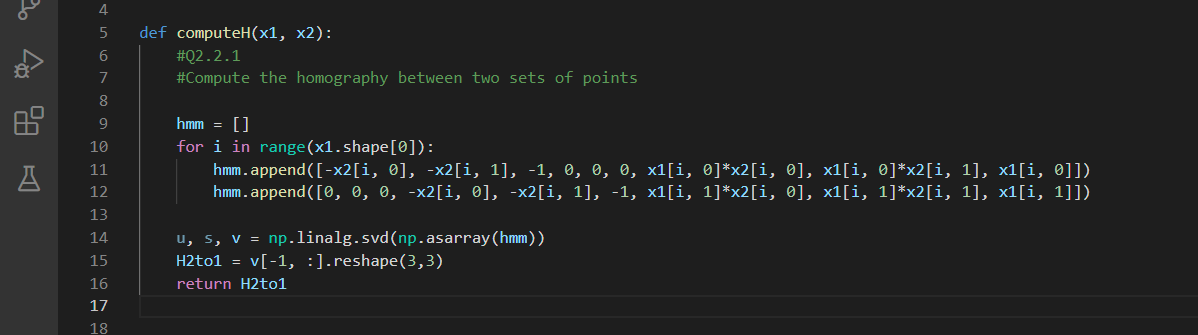


From the histogram, it can be seen as the image is rotated the number of features mapped dips significantly. From this we can conclude that even if brief descriptor is fast in computation, it is unable to detect similar features therefore not good in feature matching.

We can also see that there occurs maximum amount of matching when the orientation is the same as the original image (at 360° and 0°) and very less in most other cases. The amount of matches we see in these angles overshadow the ones appearing in the other angles.

**Problem 2.2.1**

Code Snipped – computeH (planarH.py)



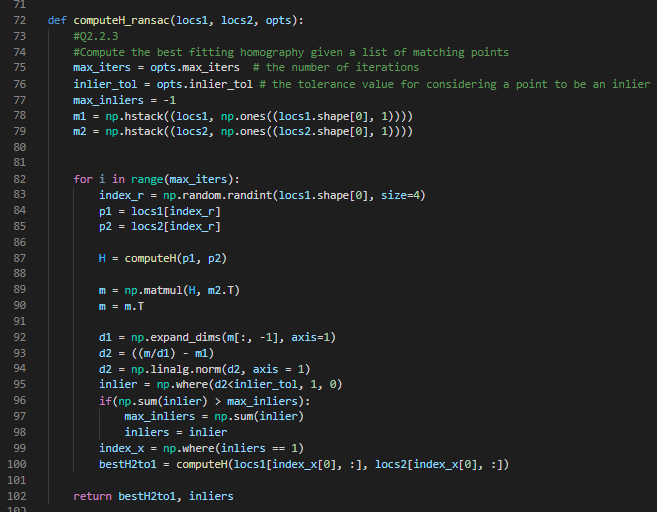
**Problem 2.2.2**

Code Snippet – compute\_norm (planarH.py)



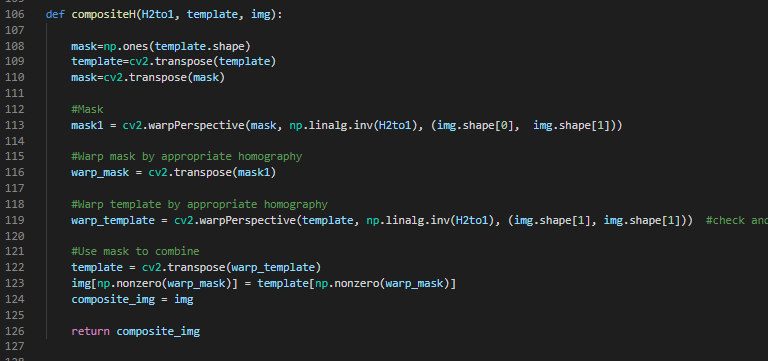
**Problem 2.2.3**

Code Snippet – computeH\_ransac (planarH.py)

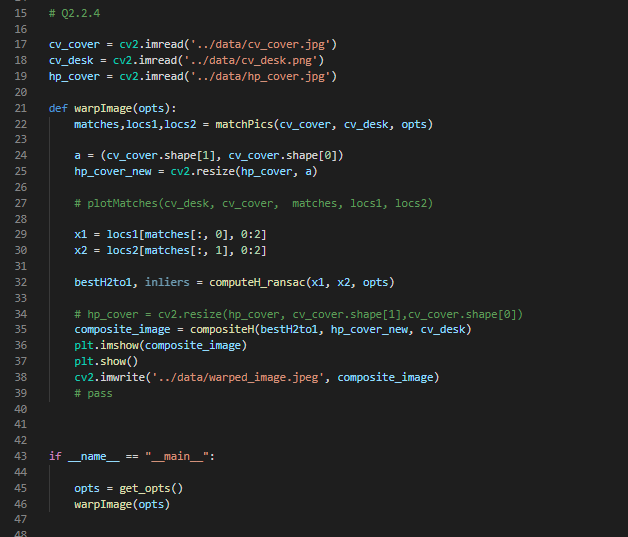


**Problem 2.2.4**

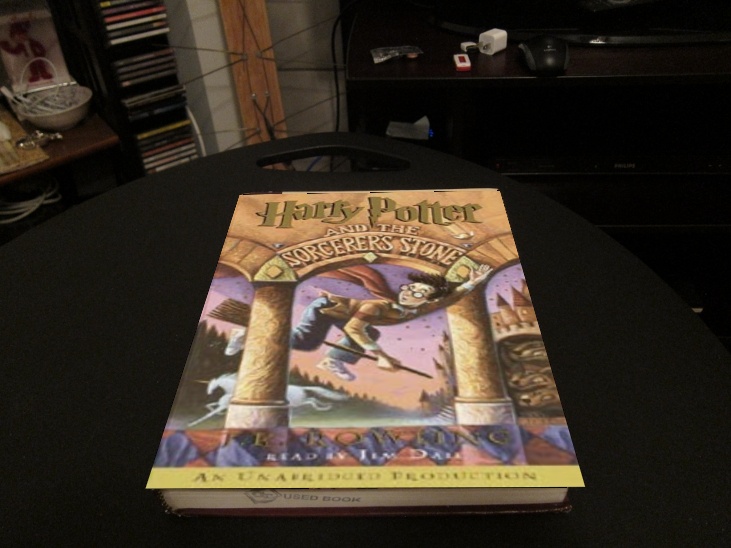
Code Snippet – composite (planarH.py)



Code Snippet – HarryPotterize.py



**Problem 2.2.5**

****   
Initial image without parameter changes

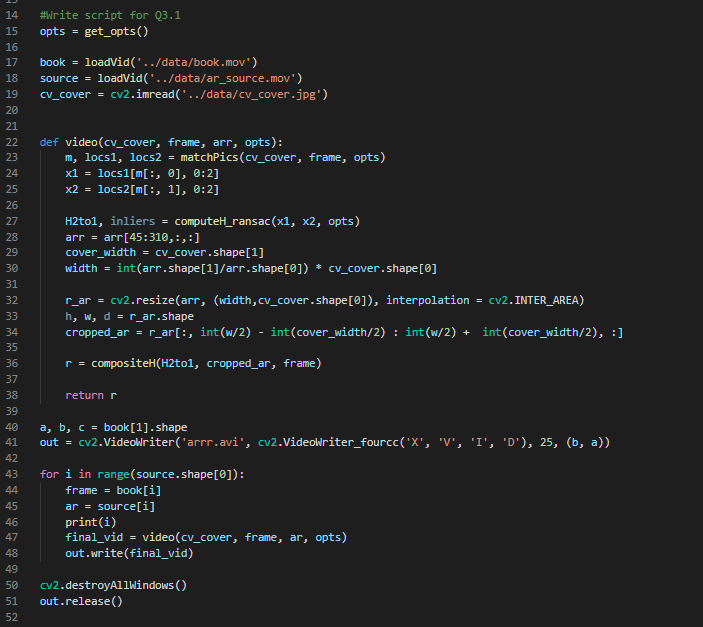
|  |  |
| --- | --- |
| **Max\_iters: 100 inlier\_tol: 2** | **Max\_iters: 500 inlier\_tol: 0.2** |
| **Max\_iters: 2500 inlier\_tol:40** | **Max\_iters: 5000 inlier\_tol: 20** |

Some conclusions that we can boil down are:

* As the number of iterations are increased the number inliers is not affected up to a certain value.
* If the tolerance is high, large number of points are used to calculate the homography and hence fewer no of iterations will be required for doing the same
* However, if the tolerance is very small, even a large no of iterations can’t help in calculating the homography, there is no change in the no of inliers detected.
* On increasing the inlier tolerance tremendously, the number of inliers increases but the image gets distorted

**Problem 3.1**

Code Snippet



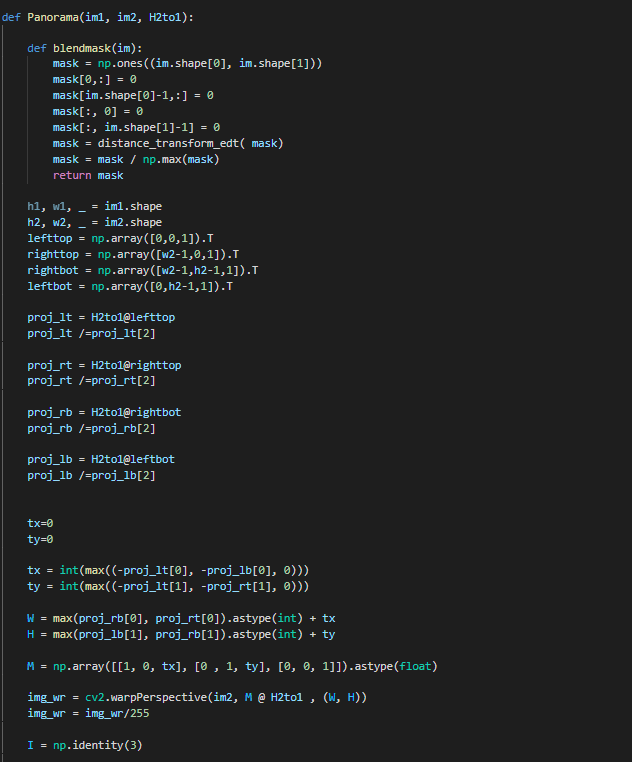
****

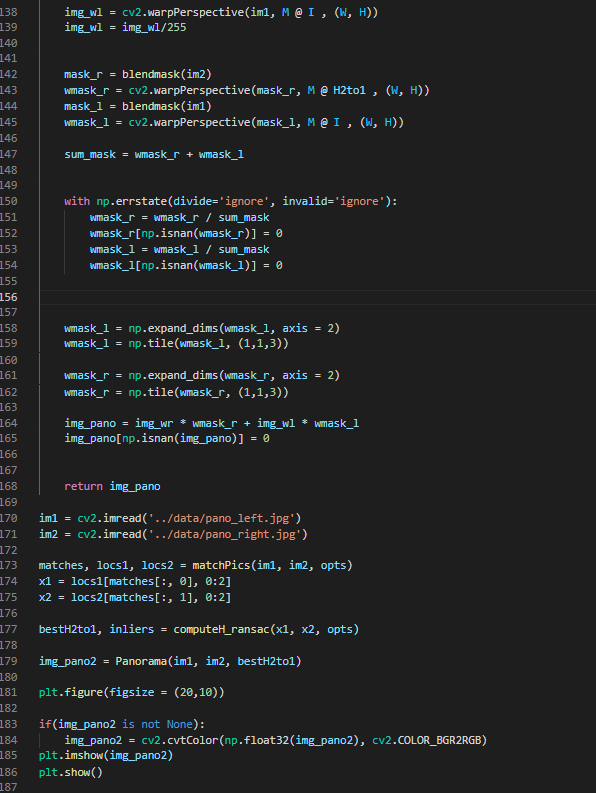
****

****

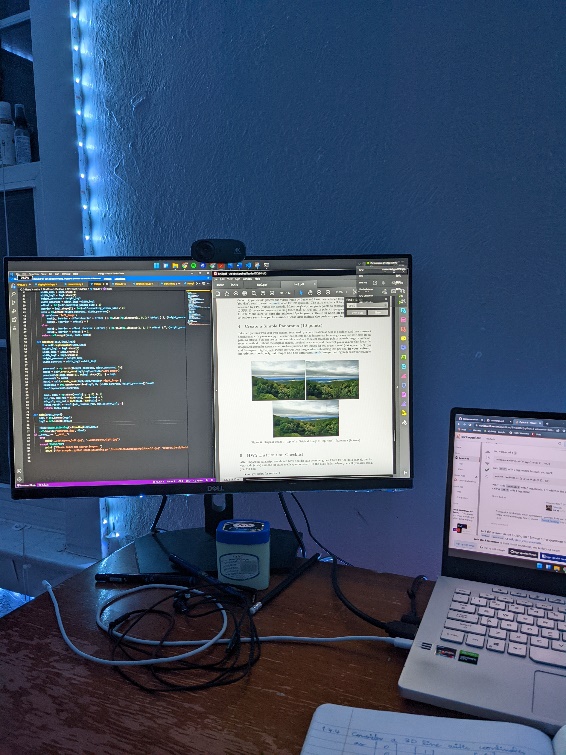
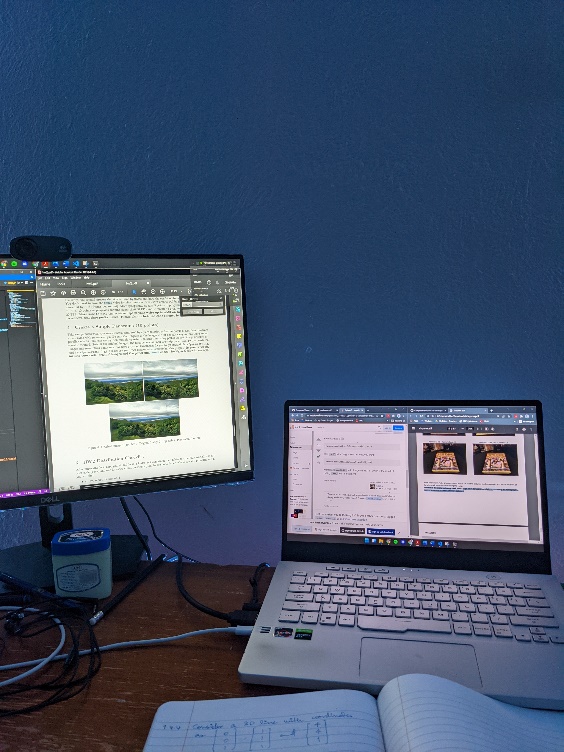
**Problem 4**

Code Snippet





Initial Images

Panaroma

