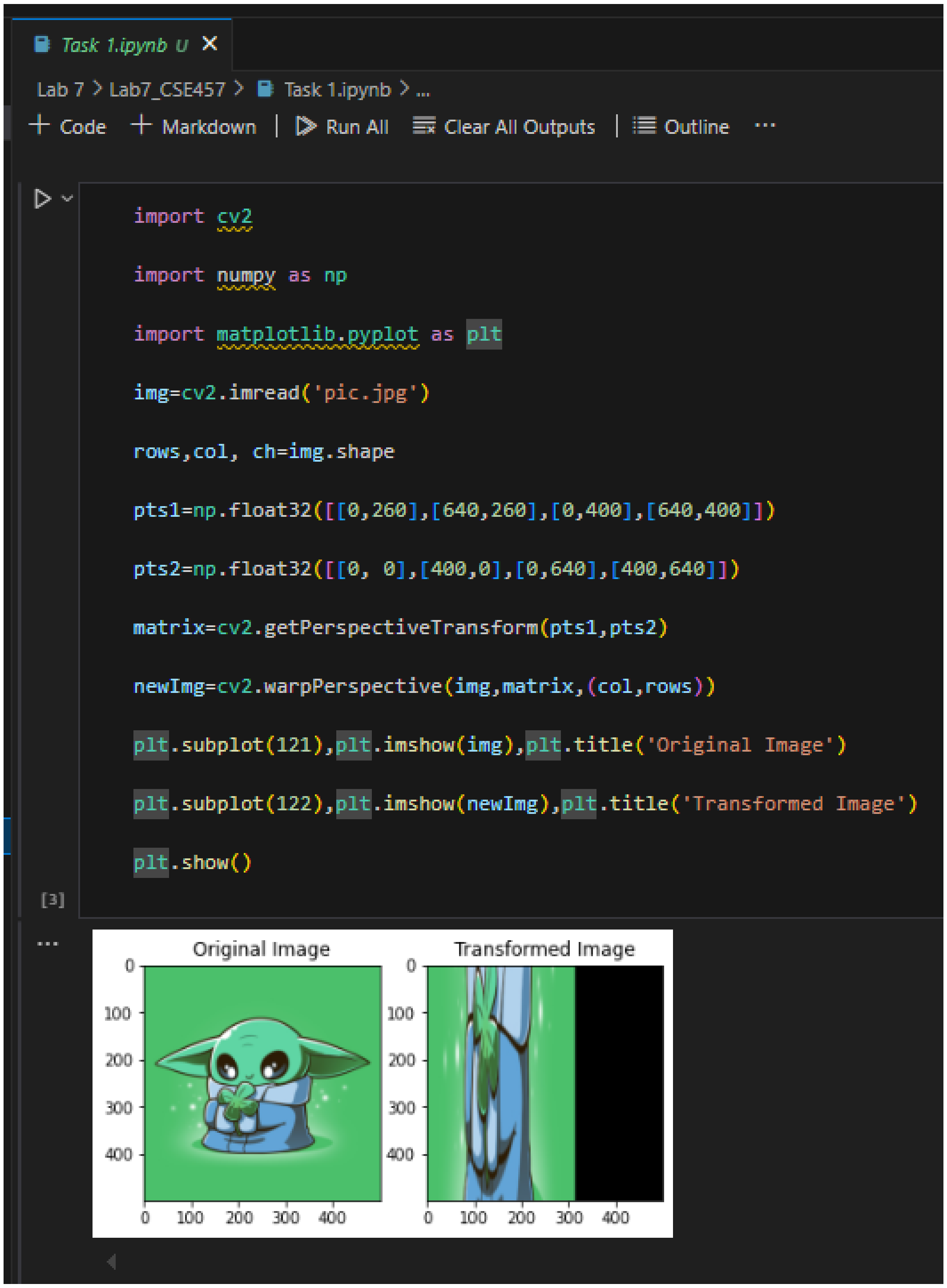
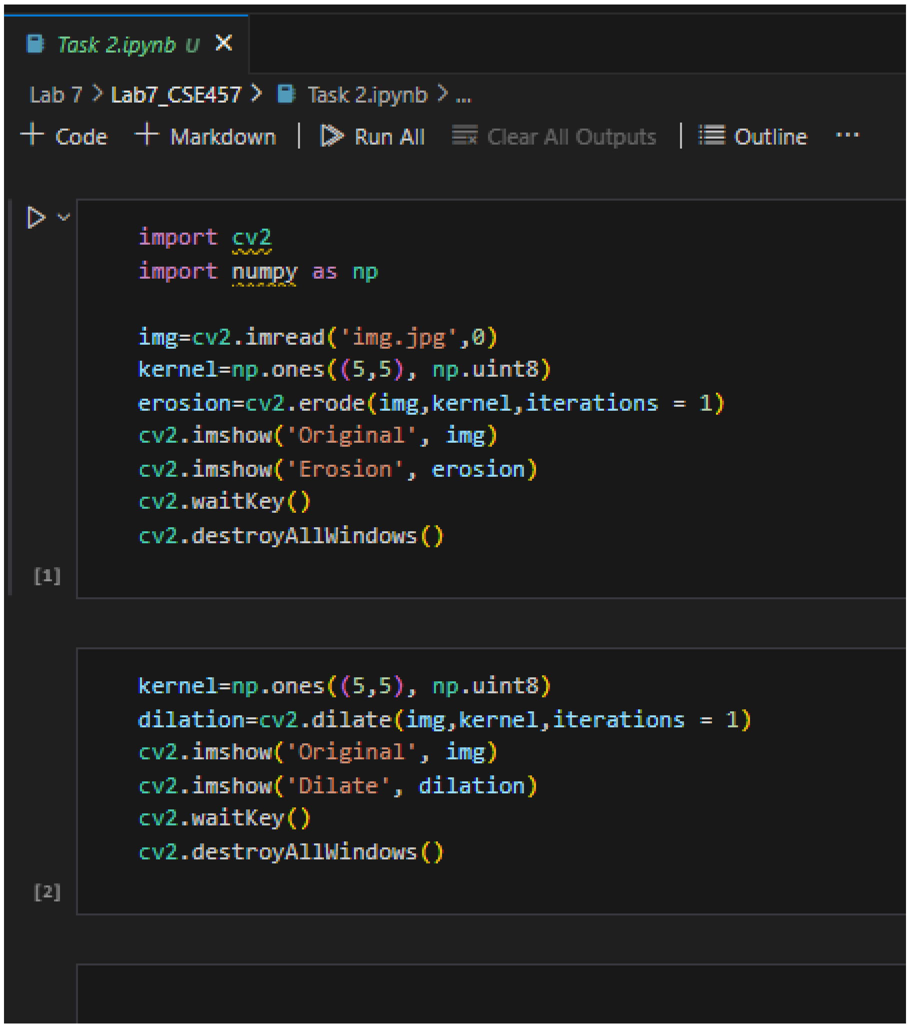
*** Perspective Transformation**


Task 1.ipynb U
Lab 7 > Lab7_CSE457 > Task 1.ipynb > ...
+ Code + Markdown | Run All Clear All Outputs | Outline ...
[3]
Original Image
Transformed Image
0
100
200
300
400

```
import cv2

import numpy as np

import matplotlib.pyplot as plt

img=cv2.imread('pic.jpg')

rows,col, ch=img.shape

pts1=np.float32([[0,260],[640,260],[0,400],[640,400]])

pts2=np.float32([[0, 0],[400,0],[0,640],[400,640]])

matrix=cv2.getPerspectiveTransform(pts1,pts2)

newImg=cv2.warpPerspective(img,matrix,(col,rows))

plt.subplot(121),plt.imshow(img),plt.title('Original Image')

plt.subplot(122),plt.imshow(newImg),plt.title('Transformed Image')

plt.show()
```

*** Morphological Transformation**


Task 2.ipynb U
Lab 7 > Lab7_CSE457 > Task 2.ipynb > ...
+ Code + Markdown | Run All Clear All Outputs | Outline ...

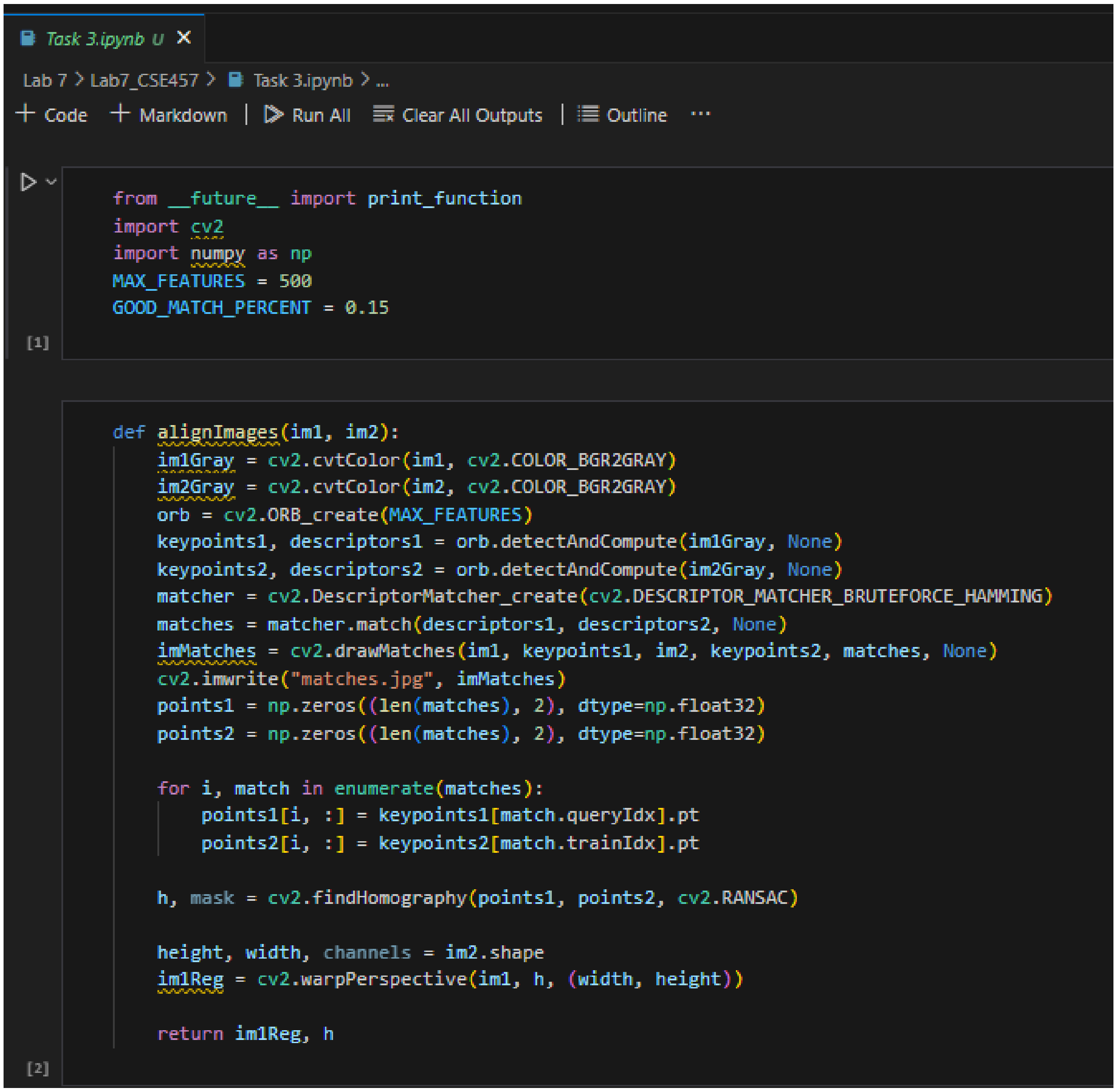
```python
import cv2
import numpy as np

img=cv2.imread('img.jpg',0)
kernel=np.ones((5,5), np.uint8)
erosion=cv2.erode(img,kernel,iterations = 1)
cv2.imshow('Original', img)
cv2.imshow('Erosion', erosion)
cv2.waitKey()
cv2.destroyAllWindows()
```

[1]

```python
kernel=np.ones((5,5), np.uint8)
dilation=cv2.dilate(img,kernel,iterations = 1)
cv2.imshow('Original', img)
cv2.imshow('Dilate', dilation)
cv2.waitKey()
cv2.destroyAllWindows()
```

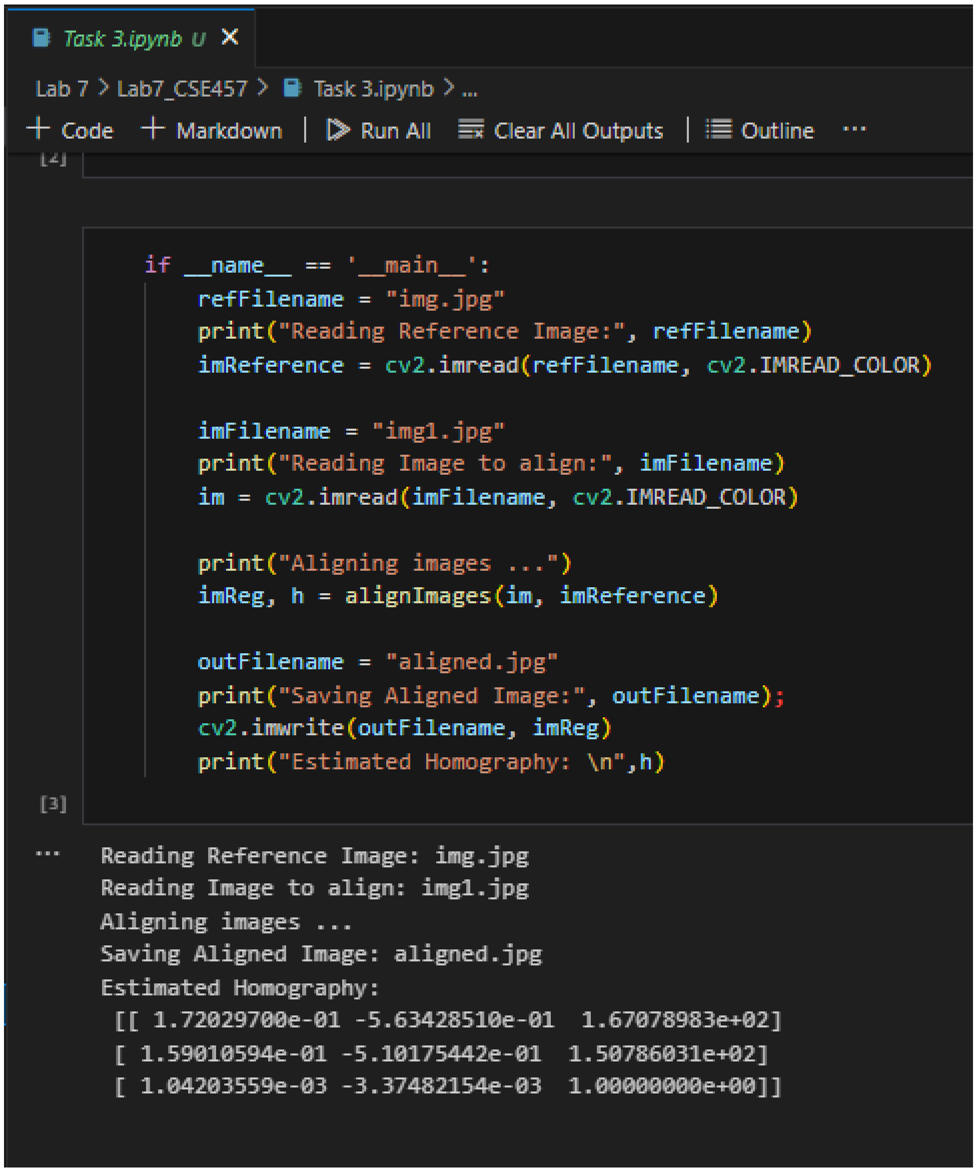
[2]

*** Image Alignment**


Task 3.ipynb U
Lab 7 > Lab7_CSE457 > Task 3.ipynb > ...
+ Code + Markdown | Run All Clear All Outputs | Outline ...

```python
from __future__ import print_function
import cv2
import numpy as np
MAX_FEATURES = 500
GOOD_MATCH_PERCENT = 0.15
```
[1]

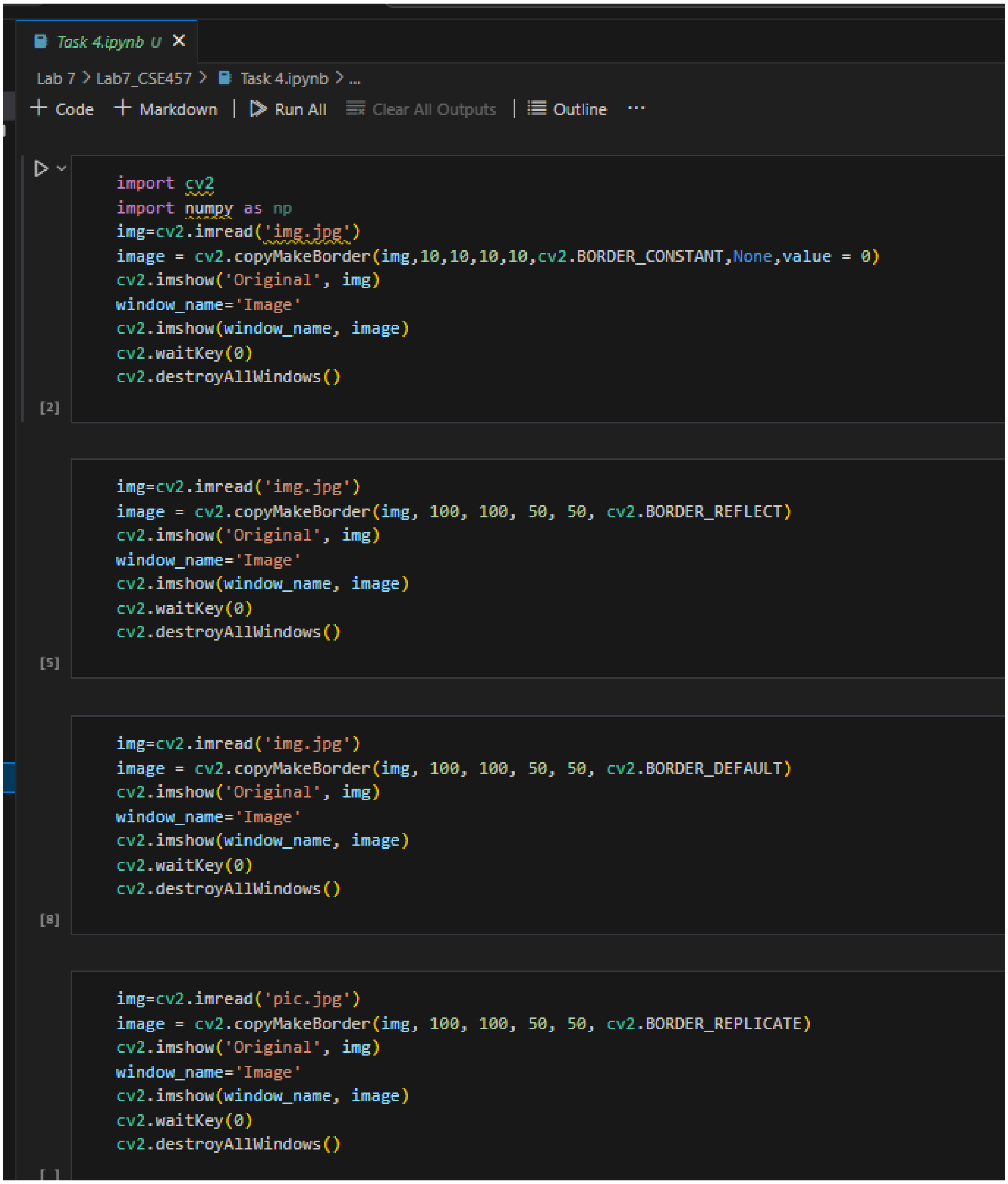
```python
def alignImages(im1, im2):
    im1Gray = cv2.cvtColor(im1, cv2.COLOR_BGR2GRAY)
    im2Gray = cv2.cvtColor(im2, cv2.COLOR_BGR2GRAY)
    orb = cv2.ORB_create(MAX_FEATURES)
    keypoints1, descriptors1 = orb.detectAndCompute(im1Gray, None)
    keypoints2, descriptors2 = orb.detectAndCompute(im2Gray, None)
    matcher = cv2.DescriptorMatcher_create(cv2.DESCRIPTOR_MATCHER_BRUTEFORCE_HAMMING)
    matches = matcher.match(descriptors1, descriptors2, None)
    imMatches = cv2.drawMatches(im1, keypoints1, im2, keypoints2, matches, None)
    cv2.imwrite("matches.jpg", imMatches)
    points1 = np.zeros((len(matches), 2), dtype=np.float32)
    points2 = np.zeros((len(matches), 2), dtype=np.float32)

    for i, match in enumerate(matches):
        points1[i, :] = keypoints1[match.queryIdx].pt
        points2[i, :] = keypoints2[match.trainIdx].pt

    h, mask = cv2.findHomography(points1, points2, cv2.RANSAC)

    height, width, channels = im2.shape
    im1Reg = cv2.warpPerspective(im1, h, (width, height))

    return im1Reg, h
```
[2]



```
if __name__ == '__main__':
    refFilename = "img.jpg"
    print("Reading Reference Image:", refFilename)
    imReference = cv2.imread(refFilename, cv2.IMREAD_COLOR)

    imFilename = "img1.jpg"
    print("Reading Image to align:", imFilename)
    im = cv2.imread(imFilename, cv2.IMREAD_COLOR)

    print("Aligning images ...")
    imReg, h = alignImages(im, imReference)

    outFilename = "aligned.jpg"
    print("Saving Aligned Image:", outFilename);
    cv2.imwrite(outFilename, imReg)
    print("Estimated Homography: \n",h)
```

```
Reading Reference Image: img.jpg
Reading Image to align: img1.jpg
Aligning images ...
Saving Aligned Image: aligned.jpg
Estimated Homography:
 [[ 1.72029700e-01 -5.63428510e-01  1.67078983e+02]
 [ 1.59010594e-01 -5.10175442e-01  1.50786031e+02]
 [ 1.04203559e-03 -3.37482154e-03  1.00000000e+00]]
```

*** Create Border around Images**


Task 4.ipynb U
Lab 7 > Lab7_CSE457 > Task 4.ipynb > ...
+ Code + Markdown | Run All Clear All Outputs | Outline ...

```
import cv2
import numpy as np
img=cv2.imread('img.jpg')
image = cv2.copyMakeBorder(img,10,10,10,10,cv2.BORDER_CONSTANT,None,value = 0)
cv2.imshow('Original', img)
window_name='Image'
cv2.imshow(window_name, image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

[2]

```
img=cv2.imread('img.jpg')
image = cv2.copyMakeBorder(img, 100, 100, 50, 50, cv2.BORDER_REFLECT)
cv2.imshow('Original', img)
window_name='Image'
cv2.imshow(window_name, image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

[5]

```
img=cv2.imread('img.jpg')
image = cv2.copyMakeBorder(img, 100, 100, 50, 50, cv2.BORDER_DEFAULT)
cv2.imshow('Original', img)
window_name='Image'
cv2.imshow(window_name, image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

[8]

```
img=cv2.imread('pic.jpg')
image = cv2.copyMakeBorder(img, 100, 100, 50, 50, cv2.BORDER_REPLICATE)
cv2.imshow('Original', img)
window_name='Image'
cv2.imshow(window_name, image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

[ ]