COMPUTER VISION

CS512 Assignment 3

Rakshith Churchagundi Amarnath
CWID: A20424771
Fall 2018

Problem Statement:

1. Loading two images and finding the similarities of them with corner detection.
2. Load and display two images or capture by webcam.
3. Estimate image gradients and applying the Harris corner detection algorithm.
4. Obtain a better localization of each corner found in the image.
5. Compute a feature vector for each corner point.
6. Display the corners by drawing empty rectangles over the original image centered at locations where corners were detected.

Proposed solution:

1. Use Harris corner detection.

   R= det(M)-k(trace(M))^2

   Where

   -det(M)=lamda1.lamda2
   -trace(M)=lamda1+lamda2
   -lamda1 and lamda2 are eigen values

2. Locating the exact position of the corners by localizing an corner

3. Locating the feature descriptors for the top corners in an image

4. Finding the matching corners by comparing the feature descriptors of the images.

Implementation Details:

1. Use webcam take two photos also give you two similar pictures.
   This can be done by using the function getImage() - getimage function saves a bit image of specified region into memory, region can be any rectangle.

```
def getImage():
    if len(sys.argv) == 3:
            im1 = cv2.imread(sys.argv[1])
```

```
        im2 = cv2.imread(sys.argv[2])
    else:
            captr = cv2.VideoCapture(0)
            for i in range(0,15):
                returnval1,im1 = captr.read()
                returnval2,im2 = captr.read()
            if returnval1 and returnval2:
                cv2.imwrite("capture1.jpg", im1)
                cv2.imwrite("capture2.jpg", im2)
    combine = np.concatenate((im1, im2), axis=1)
    return combine, im1, im2;
```

2. Use an isolate display function to display result for every function.

   It's done using following function

```
print("Input key(press 'H' for help, press 'q' to quit):")
    k = input()
    while k != 'q':
        if k == 'h':
            n = input("The variance of Guassian scale:")
            winSize = input("Window Size :")
            k = input("the weight of the trace in the harris conner detector[0,
            0.5]:")
            threshold = input("threshold value:")
            rslt = harris(combine, n, winSize, k, threshold)
            showWin(rslt)
        if k == 'f':
            rslt = featureVector(im1, im2)
            showWin(rslt)
        if k == 'b':
            rslt = betterLocalization(combine)
            showWin(rslt)
        if k == 'H':
            help()
        print("Input key (press 'H' for help, press 'q' to quit):")
        k = input()
```

3. When taken parameter into some function, we need to know the type and change it into right type to compute. There will be a couple of print commands to make sure the user can execute the respective function.
This can be done using following function :-

```
def help():
    print("'h': Estimate image gradients and apply Harris corner detection algorithm to your image.")
    print("'b': Obtain a better localization of each corner.")
    print("'f': Compute a feature vector for each corner were detected.\n")
```

3. Press 'H' to get help of this program.
   It displays the information about which key should be pressed for its respective task to get executed.

4. It required input parameter to get the result of Harris function.

6. Convert the image to grayscale.

```
def cvt2Gray(im):
im_bw = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
cv2.imshow("Display window", im_bw)
return im_bw
```

7. Corner Dectecion function

```
def harris(im, n, winSize, k, threshold):
    n = int(n)
    winSize = int(winSize)
    k = float(k)
    threshold = int(threshold)
    copy = im.copy()
    rList = []
    height = im.shape[0]
    width = im.shape[1]
    offset = int(winSize / 2)
    im = cvt2Gray(im)
    im = np.float32(im)
    im = smooth(im, n)
    dy, dx = np.gradient(im)
    Ixx = dx ** 2
    Ixy = dy * dx
    Iyy = dy ** 2
```

```
                for y in range(offset, height - offset):
                for x in range(offset, width - offset):
                        windowIxx = Ixx[y - offset : y + offset + 1, x - offset : x + offset + 1]
                        windowIxy = Ixy[y - offset : y + offset + 1, x - offset : x + offset + 1]
                        windowIyy = Iyy[y - offset : y + offset + 1, x - offset : x + offset + 1]
                                Sxx = windowIxx.sum()
                                Sxy = windowIxy.sum()
                                Syy = windowIyy.sum()
                                det = (Sxx * Syy) - (Sxy ** 2)
                                trace = Sxx + Syy
                                r = det - k *(trace ** 2)
                                rList.append([x, y, r])
                                if r > threshold:
                                copy.itemset((y, x, 0), 0)
                                copy.itemset((y, x, 1), 0)
                                copy.itemset((y, x, 2), 255)
                cv2.rectangle(copy, (x + 10, y + 10), (x - 10, y - 10), (255, 0, 0), 1)
                return copy
```

8. Feature Vector function:

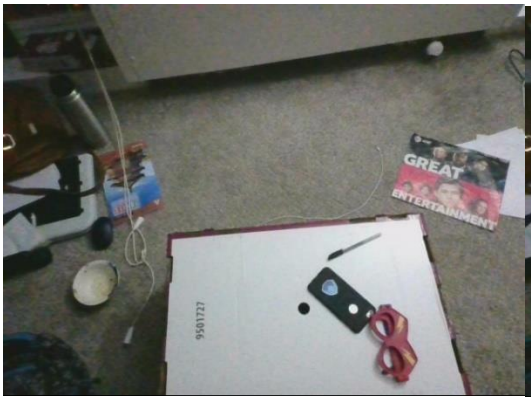
```
        def featureVector(im1, im2):
                orb = cv2.ORB_create()
                kpt1, des1 = orb.detectAndCompute(im1,None)
                kpt2, des2 = orb.detectAndCompute(im2,None)
                bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
                matches = bf.match(des1,des2)
                matches = sorted(matches, key = lambda x:x.distance)
                kpt1List = []
                kpt2List = []
                for m in matches:
                        (x1, y1) = kpt1[m.queryIdx].pt
                        (x2, y2) = kpt2[m.trainIdx].pt
                        kpt1List.append((x1, y1))
                        kpt2List.append((x2, y2))
                        for i in range(0, 50):
                        point1 = kpt1List[i]
                        point2 = kpt2List[i]
cv2.putText(im1, str(i), (int(point1[0]), int(point1[1])),  cv2.FONT_HERSHEY_SIMPLEX, 1, 255, 2)
cv2.putText(im2, str(i), (int(point2[0]), int(point2[1])),  cv2.FONT_HERSHEY_SIMPLEX, 1, 255, 2)
                rslt = np.concatenate((im1, im2), axis=1)
                return rslt
```

9. Better localization function:

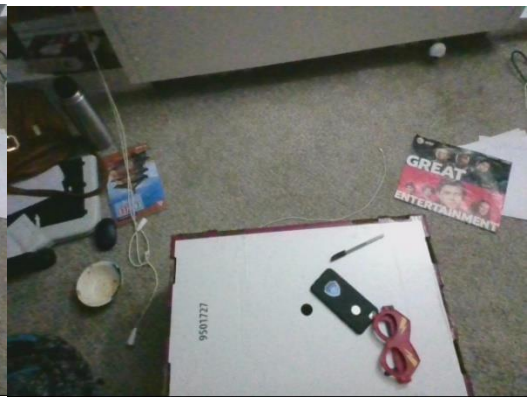
```
def betterLocalization(im):
gray = cvt2Gray(im)
gray = np.float32(gray)
distance1 = cv2.cornerHarris(gray,2,3,0.04)
distance1 = cv2.dilate(distance1,None)
ret, distance1 = cv2.threshold(distance1,0.01*distance1.max(),255,0)
distance1 = np.uint8(distance1)

ret, labels, stats, centroids = cv2.connectedComponentsWithStats(distance1)

criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 100, 0.001)
corners = cv2.cornerSubPix(gray,np.float32(centroids),(5,5),(-1,-1),criteria)

rslt = np.hstack((centroids,corners))
rslt = np.int0(rslt)
im[rslt[:,1],rslt[:,0]]=[0,0,255]
im[rslt[:,3],rslt[:,2]] = [0,255,0]
return im
```

Results and Discussions

- Loading the image
  The image is captured from the camera and saved at the specified location.

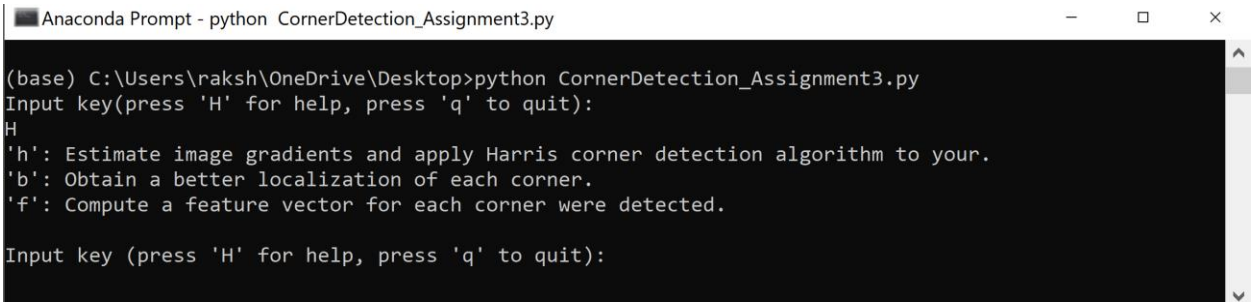


capture1.jpg

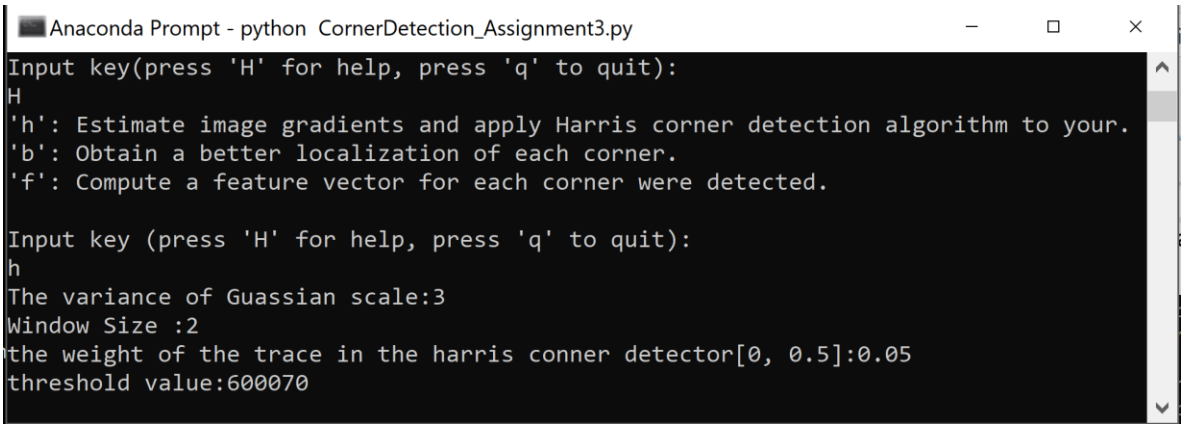


captured.jpg

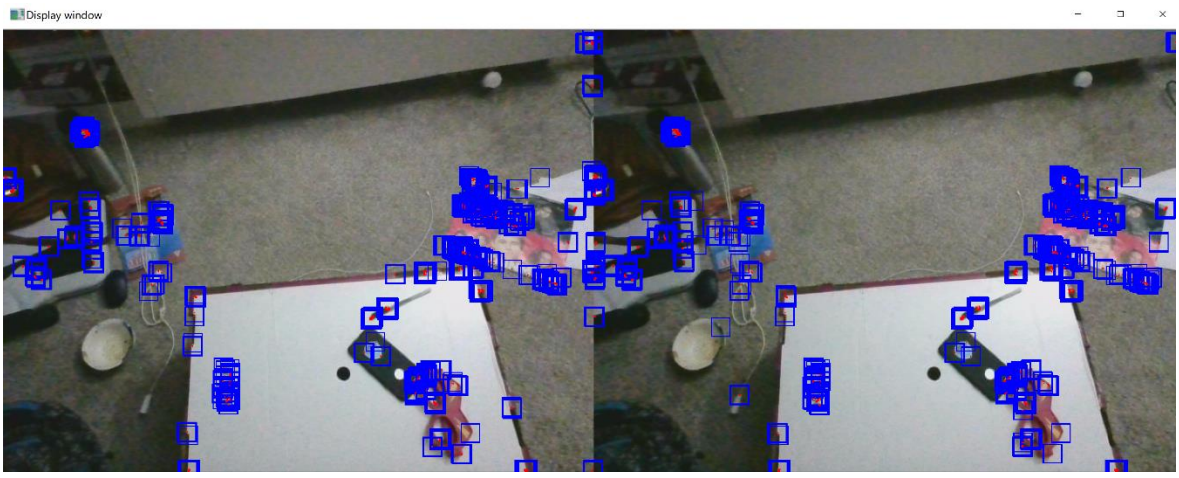
- Including the help key describing the functionality



- Taking parameters to calculate Harris corner detection



- Harris Corner Detection



- Better Localization

- **Computing the feature vector**

References:

- https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_features_harris/py_features_harris.html
- https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_shi_tomasi/py_shi_tomasi.html

- https://en.wikipedia.org/wiki/OpenCV
- https://en.wikipedia.org/wiki/Corner_detection
-