cs512 Assignment 3

Megha Tatti
CWID: A20427027
Fall2018

Problem Statement:

1. Corner Detection with matching two similar images.
2. Load and display 2 images or capture by webcam.
3. Estimate image gradients and apply the Harris corner detection algorithm.
4. Obtain a better localization of each corner.
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. Find the exact position of the corners by localizing the corner

3. Find the feature descriptors for the top corners in the image

4. Find 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 the following function.:-

```
def getImage():
    if len(sys.argv) == 3:
            img1 = cv2.imread(sys.argv[1])
            img2 = cv2.imread(sys.argv[2])
                    else:
                    cp = cv2.VideoCapture(0)
                    for i in range(0,15):
```

```
                    rvalue1,img1 = cp.read()
                    rvalue2,img2 = cp.read()
                if rvalue1 and rvalue2:
                    cv2.imwrite("image_captured1.jpg", img1)
                    cv2.imwrite("image_captured2.jpg", img2)
        combine = np.concatenate((img1, img2), axis=1)
        return combine, img1, img2;
```

2. Use an isolate display function to display result for every function.
   It's done using following function

```
print("Press 'H' for help!! Press 'q' to quit:")
    k = input()
    while k != 'q':
        if k == 'h':
            n = input("Enter the variance of Guassian scale:")
            wSize = input("Enter the Window Size :")
            k = input("Enter the weight of the trace in the harris conner
detector(k)[0, 0.5]:")
            threshold = input("Enter the threshold value:")
            print("Result processing.....")
            res = harris(combine, n, wSize, k, threshold)
            showWin(res)
        if k == 'f':
            res = featureVector(img1, img2)
            showWin(res)
        if k == 'b':
            res = betterLocalization(combine)
            showWin(res)
        if k == 'H':
            help()
        print("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.
   This can be done using following function :-

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

4. Press ‘H’ to get help of this program.
5. It required input parameter to get the result of Harris function.
6. Convert the image to grayscale.

```
def cvt2Gray(img):
        img_bw = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        #img_bw = cv2.cvtColor(img_bw,cv2.COLOR_GRAY2BGR)
        cv2.imshow("Display", img_bw)
        return img_bw
```

7. Corner Dectecion function

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

        for y in range(offset, height - offset):
                for x in range(offset, width - offset):
                        wIxx = Ixx[y - offset : y + offset + 1, x - offset : x + offset + 1]
                        wIxy = Ixy[y - offset : y + offset + 1, x - offset : x + offset +
1]
                        wIyy = Iyy[y - offset : y + offset + 1, x - offset : x + offset +
1]
                        Sxx = wIxx.sum()
                        Sxy = wIxy.sum()
                        Syy = wIyy.sum()
                        determinant = (Sxx * Syy) - (Sxy ** 2)
                        trace = Sxx + Syy
```

```
                    r = determinant - 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(img1, img2):
        orb = cv2.ORB_create()# Initiating SIFT detector
        keyp1, des1 = orb.detectAndCompute(img1,None)
        keyp2, des2 = orb.detectAndCompute(img2,None)
        bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)  # creating BFMatcher object
        matches = bf.match(des1,des2)
        matches = sorted(matches, key = lambda x:x.distance)  # Sorting in the order of their distance.
        keyp1List = []
        keyp2List = []
        for m in matches:
                (x1, y1) = keyp1[m.queryIdx].pt
                (x2, y2) = keyp2[m.trainIdx].pt
                keyp1List.append((x1, y1))
                keyp2List.append((x2, y2))
        for i in range(0, 50):
                pt1 = keyp1List[i]
                pt2 = keyp2List[i]
                cv2.putText(img1, str(i), (int(pt1[0]), int(pt1[1])), cv2.FONT_HERSHEY_SIMPLEX, 1, 255, 2)
                cv2.putText(img2, str(i), (int(pt2[0]), int(pt2[1])), cv2.FONT_HERSHEY_SIMPLEX, 1, 255, 2)
        res = np.concatenate((img1, img2), axis=1)
        return res
```

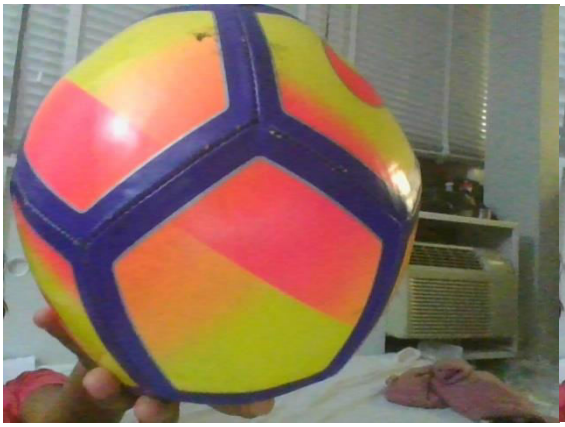
9. Better localization function:

```
def betterLocalization(img):
        #gray = cvt2Gray(img)
        gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
        gray = np.float32(gray)
        dist = cv2.cornerHarris(gray,2,3,0.04)
        dist = cv2.dilate(dist,None)
```

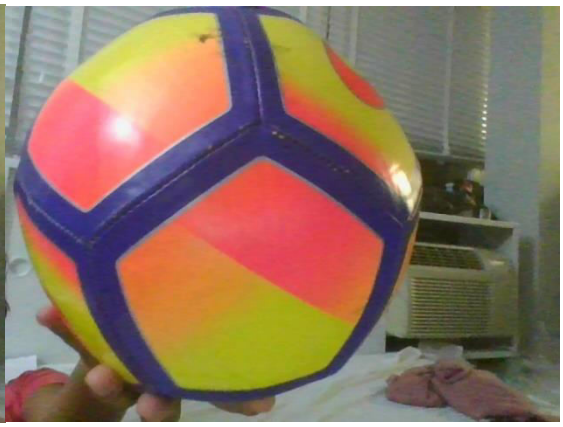
```
        rt, dist = cv2.threshold(dist,0.01*dist.max(),255,0)
        dist = np.uint8(dist)

        rt, labels, stats, centroids = cv2.connectedComponentsWithStats(dist)

        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)

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

Results and Discussions

- Loading the image



Image_captured1.jpg



Image_captured2.jpg

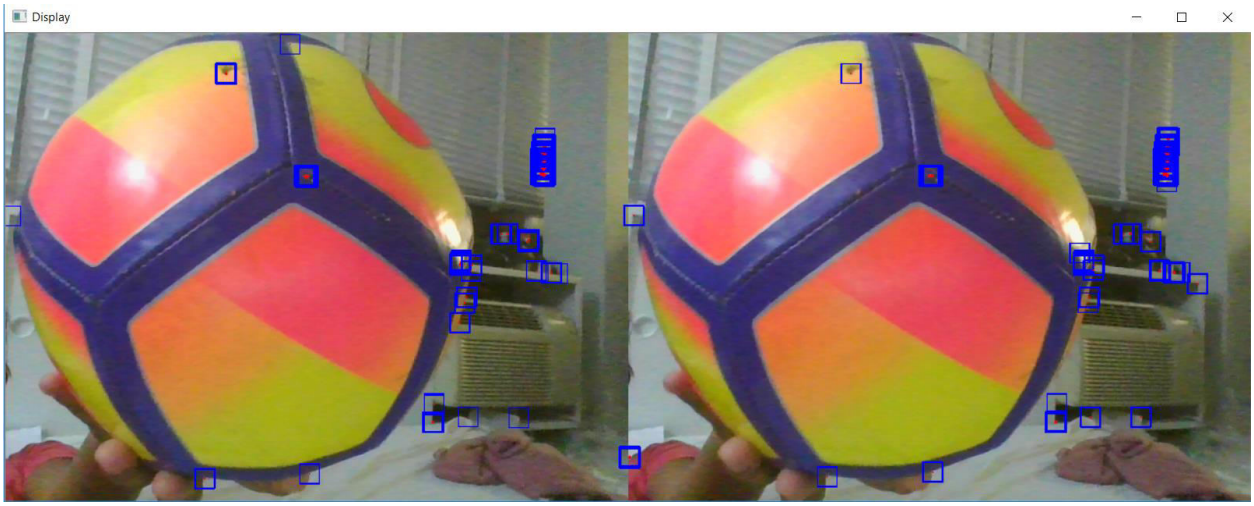
- Including the help key describing the functionality

```
(base) C:\Users\megha\Desktop>python CornerDetection.py
Press 'H' for help!! Press 'q' to quit:
H
'h': Estimate image gradients and apply Harris corner detection algorithm.
'b': Obtain a better localization of each corner.
'f': Compute a feature vector for each corner were detected.
```

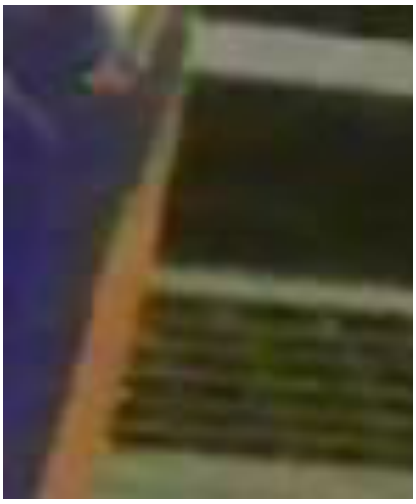
- Taking parameters to calculate Harris corner detection

```
(base) C:\Users\megha\Desktop>python CornerDetection.py
Press 'H' for help!! Press 'q' to quit:
H
'h': Estimate image gradients and apply Harris corner detection algorithm.
'b': Obtain a better localization of each corner.
'f': Compute a feature vector for each corner were detected.

Press 'H' for help!! Press 'q' to quit:
h
Enter the variance of Guassian scale:4
Enter the Window Size :2
Enter the weight of the trace in the harris conner detector(k)[0, 0.5]:0.07
Enter the threshold value:6700000
Result processing..........
```

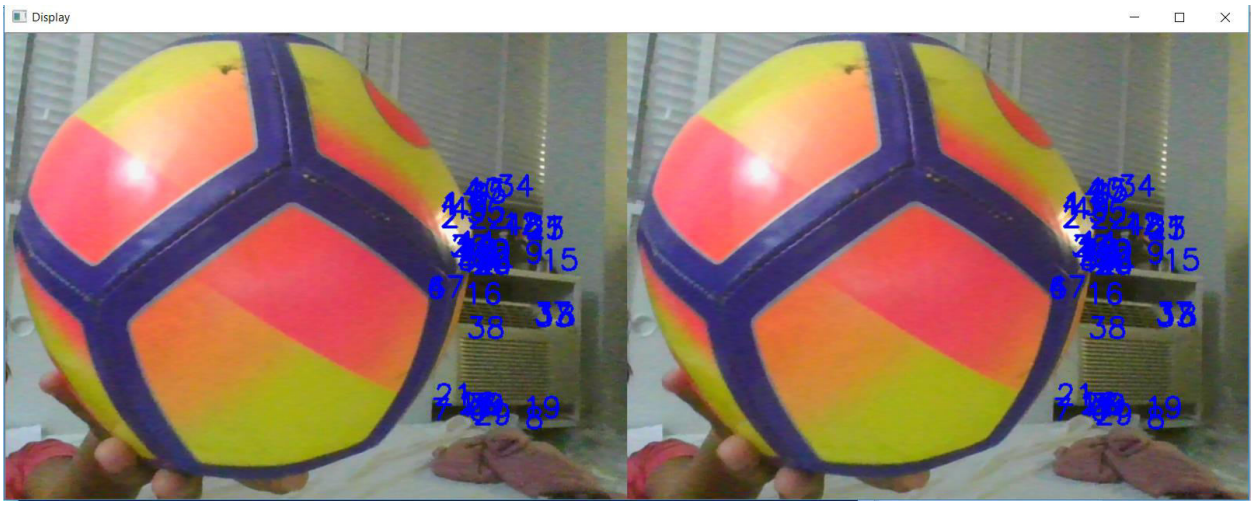
- Harris Corner Detection



- 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://en.wikipedia.org/wiki/OpenCV
- https://www.w3schools.com/python/