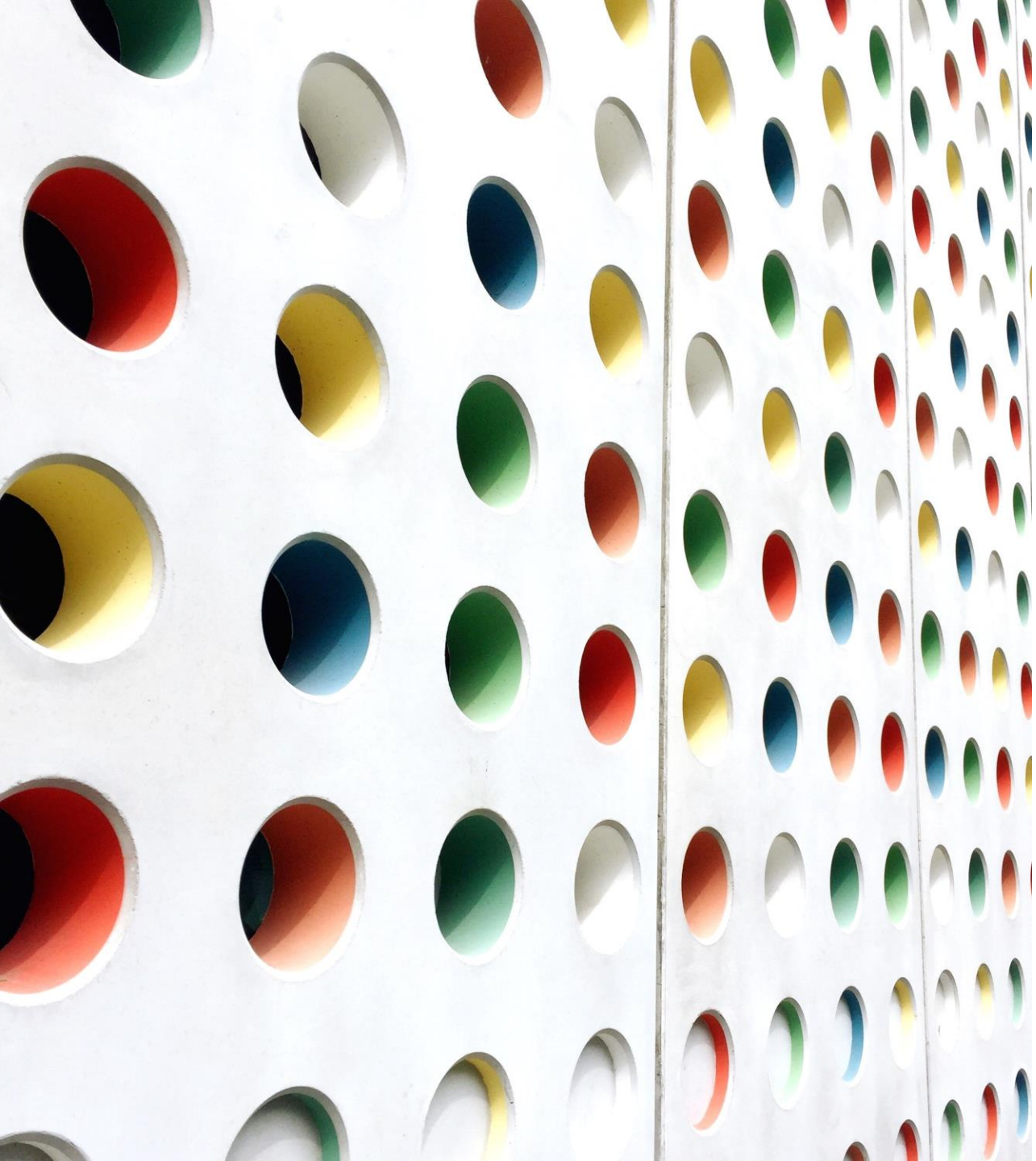

Computer Vision Homework 3

Matt Ferguson

July 27th, 2022

Requirements

For each provided image:

- Load each image.
- Determine the top 100 keypoints using OpenCV goodFeaturesToTrack (Shi-Tomasi corner detection).
- Determine the top 100 keypoints using self-implemented Harris corner detection.
- Print the filename and co-ordinates of the top three corner points for both corner detection methods.
- Draw circles around the top 100 keypoints for both corner detection methods.
- Display the output images for both corner detection images as side-by-side concatenated images.
- Discuss the differences in output.

Load and convert image

Find keypoints using the OpenCV function

Custom implementation of the Harris corner detector

Print both sets of results to the console

Annotate both images properly

Concatenate output image horizontally and write to file

Discussion of differences between two images

Code Overview

We iterate through each of the provided files and load the image as float32. We convert the image to monochrome by constructing a 2d-array of mean value per pixel. We blur our image to reduce the impact that noise will have on keypoint detection. Then we invoke OpenCV corner detection function 'goodFeaturesToTrack()' and pass our monochrome image as a function input in order to find the top 100 keypoints. Then we circle the 100 Shi-Tomasi keypoints in green. Next, we proceed to our self-implemented Harris keypoint detection function. We perform Sobel operations in x and y to get our derivatives. We then form the structure matrix, the determinant, and the trace using linear algebra. Our Harris function corner map is obtained by subtracting the determinant by the scaled trace. A suppression algorithm is applied so only the highest single maxima will exist in a 10x10 neighborhood. Then the top 100 key points are obtained by sorting the Harris corner map by key point strength. The top 100 Harris keypoints are marked with red circles on the original image. Finally, the Shi-Tomasi and Harris output images are concatenated and output as a combined image in a single png file.

```
# Computer Vision HW3
import cv2
import numpy as np

files='AllmanBrothers','CalvinAndHobbes','Chartres','Elvis1956'
for file in files:
    print(file+'.png')
    original_img = cv2.imread(r'C:\Users\Matt\OneDrive\Virginia Tech\CV\Corner Detection\\'+file+'.png')
    original_img=original_img.astype(np.float32)
    img=np.mean(original_img.copy(),2)
    img=cv2.GaussianBlur(img,(15,15),0)

    # Shi Tomasi Library Function
    corners=cv2.goodFeaturesToTrack(img,100,0.01,10)
    corners=corners[:,0,:]
    print(corners[:3])
    corners=np.intp(corners)

    shit_corner_img=original_img.copy()
    for i in corners:
        x,y=i.ravel()
        cv2.circle(shit_corner_img,(x,y),5,(0,254,0))

    img_out=cv2.imwrite(r'C:\Users\Matt\Desktop\Results\\'+file+'Shi_Tomasi_Corners.png', shit_corner_img)

    # Harris Function
    harris_corner_img=original_img.copy()
    img=np.mean(original_img,2)

    ix=cv2.Sobel(src=img, ddepth=cv2.CV_64F, dx=1, dy=0, ksize=5)
    iy=cv2.Sobel(src=img, ddepth=cv2.CV_64F, dx=0, dy=1, ksize =5)
    ixx=ix**2
    iyy=iy**2
    ixy=ix*iy
    ixx=cv2.GaussianBlur(ixx,(3,3),1)
    iyy=cv2.GaussianBlur(iyy,(3,3),1)
    ixy=cv2.GaussianBlur(ixy,(3,3),1)
    corner_map=(ixx*iyy-ixy**2)-0.05*(ixx+iyy)**2

    dX, dY = 10,10
    M, N = img.shape
    for x in range(0,M-dX+1):
        for y in range(0,N-dY+1):
            window = corner_map[x:x+dX, y:y+dY]
            if np.sum(window)==0:
                localMax=0
            else:
                localMax = np.amax(window)
            maxCoord=np.unravel_index(np.argmax(window), window.shape) + np.array((x,y))
            #suppress everything
            corner_map[x:x+dX, y:y+dY]=0
            #reset only the max
            corner_map[tuple(maxCoord)] = localMax

    row,column=img.shape[:2]
    max_corner_map=np.empty((3,row*column))
    counter=0
    for r in range(row):
        for c in range(column):
            max_corner_map[:,counter] = [corner_map[r,c],r,c]
            counter+=1

    max_corner_map = np.transpose(max_corner_map)
    max_corner_map = max_corner_map[~np.all(max_corner_map == 0, axis=1)]
    max_corner_map= max_corner_map[max_corner_map[:, 0].argsort()]
    max_corner_map=max_corner_map[-100:]
    print(np.fliplr(max_corner_map[-3:,1:]))

    for r in range(100):
            y=int(max_corner_map[r,1])
            x=int(max_corner_map[r,2])
            cv2.circle(harris_corner_img, (x,y),5,(0,0,254))
    img_out_2=cv2.imwrite(r'C:\Users\Matt\Desktop\Results\\'+file+'_Harris_Corners.png',harris_corner_img)

    output=np.concatenate((shit_corner_img,harris_corner_img),axis=1)
    img_out=cv2.imwrite(r'C:\Users\Matt\Desktop\Results\\'+file+'_Concatenate.png',output)
```

Keypoint Display 1

The Harris function better found the corners on the interior keys of the piano. The left ridge of the piano deep in the page of the image is much better defined by Shi-Tomasi, but perhaps it would be equally identified by Harris if we were to increase key points. Both algorithms found all the buttons on the front-left subject's torso. The central microphone in the background is marked by Harris, and not marked by Shi-Tomasi. The microphone in front of the left subject's face is better identified by Shi-Tomasi. Overall, the remaining instruments in the background, the guitar, drum set, and the top of the amplifier are all marked to a degree. The microphone and the boom are well defined by both methods. Notably the subject on the right has very few keypoints, perhaps a low intensity gradient from the subject's low contrast against the crowd is the cause. Harris was slightly more interested in the crowd than Shi-Tomasi.

Shi-Tomasi:

AllmanBrothers.png
[[219, 371]
[21, 64]
[444, 229]]

Harris Keypoints:

AllmanBrothers.png
[[245, 39]
[496, 77]
[530, 229]]

Keypoint Display 2

The left subject has both eyes marked as a keypoint with the Harris function. Only a single eye of the left subject is marked by Shi-Tomasi. The right subject's eyes are marked in both detection functions. The left subject's angular whiskers are well defined by both Shi-Tomasi and Harris corner detection. Overall, there is a good identification of both subjects' heads, arms, torsos, and legs (when not obscured) by both corner detection functions. Notably the right subject's arms are much better defined in Harris than Shi-Tomasi. The right subject is obscured by a hat and a map, and both objects are identified by Shi-Tomasi and Harris. Many keypoints are found across both algorithms in the grass and terrain along the bottom of the image. If we were to extend the keypoint list the similarity of these two maps will likely increase given the simplistic nature of the image. The image is binary or nearly binary and there are limited edges in comparison to the other images we analyze. The image overall is clean with little noise. Note the images are in fact concatenated with the significant whitespace left intact and text allowed to overflow onto the whitespace for formatting purposes.

Shi-Tomasi:

CalvinAndHobbes.png
[[529, 552]
[518, 403]
[390, 472]]

Harris Keypoints:

CalvinAndHobbes.png
[[635, 554]
[538, 381]
[392, 308]]

Keypoint Display 3

Interestingly, Shi-Tomasi better identified the central portion of the circular windows, whereas Harris better identified the outer portion of the circular windows. Harris found more keypoints along the left spire. Both methods failed to find any keypoints along the right spire which may be due to the gradient in intensity being too slight against the sky. In comparison, the left spire has those dark protrusions that give good gradients against the high intensity sky. Both algorithms identified many if not most key points as being within the forest, the street, and individuals within the street. It is interesting how these algorithms are drawn to the periphery. These peripheral objects have many high frequency changes and are filled with many large gradients, whereas the low frequency church provides less opportunities for large gradients to appear. The church is of highly uniform color and has similar grayscale intensity to the sky giving it less comparative keypoints. A human would be hard pressed to ignore the geometry of this beautiful church to the same degree.

Shi-Tomasi Keypoints:

Chartres.png
[[17, 581]
 [11, 614]
 [13, 596]]

Harris Keypoints:

Chartres.png
[[410, 411]
 [42, 479]
 [319, 745]]

Keypoint Display 4

The front subject’s facial features are modestly identified by both algorithms. The eyes of this subject fail to map to keypoints and no keypoints appear in the subject’s hair in Shi-Tomasi. The microphone the front subject is holding is well mapped by both algorithms. The guitar and drum set in the background are detected by both algorithms. The subject's shoes, and jacket are identified by both algorithms. The main subject’s hand gripping the microphone is well identified by both algorithms. However, the opposite hand is not well identified, and perhaps a single keypoint is identified and only by Harris. Notably the curtains in the background and the floor of the stage do not contain any keypoints. This is likely due to the low gradient of the curtains and background. It does not help that the frequency of these features is low as well giving less opportunity for large gradients. Our algorithms certainly performed well in distinguishing the foreground and background.

Shi-Tomasi:

Elivis1956.png
[[430, 438]
 [550, 769]
 [728, 1206]]

Harris Keypoints:

Elvis1956.png
[[593, 425]
 [579, 411]
 [616, 426]]

Works Referenced

1: OpenCV: Shi-Tomasi Corner Detector & Good Features to Track

2: Harris Corner Detector implementation in python – Muthukrishnan

3: OpenCV - Non local maxima suppression in python - Stack Overflow