Mujtaba Shahid Faizi

BSCS-5A

#131818

Lab 4 of Computer Vision

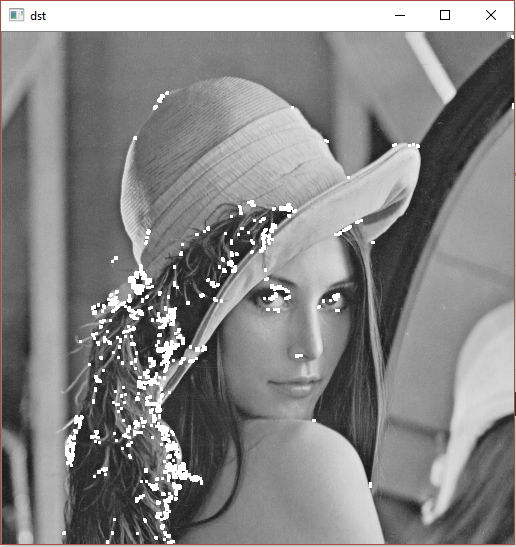
**All the explanations/results are mentioned in the code as comments.**

**Task 1:  
CODE:**

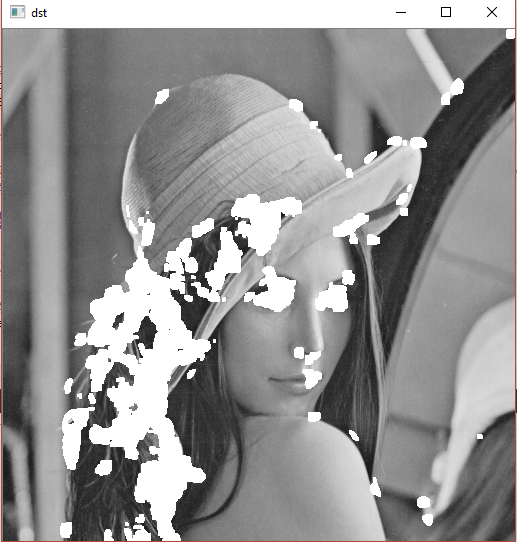
**import** cv2  
**import** numpy **as** np  
**from** matplotlib **import** pyplot **as** plt  
  
*#we convert to grayscale to increase the processing speed*img = cv2.imread(**'lena.png'** ,0)  
gray = np.float32(img)  
  
dst = cv2.cornerHarris(gray,2,3,0.04)  
  
*#For every pixel p , the function cornerEigenValsAndVecs considers a blockSize \* blockSize neighborhood S(p) .  
#It calculates the covariation matrix of derivatives over the neighborhood  
#increasing the blocksize parameter gives wider corners.  
# dst = cv2.cornerHarris(gray,10,3,0.04)  
  
#Decreasing the kernal size for sobel detector parameter gives poor results, and the best results are obtained for 3\*3  
# dst = cv2.cornerHarris(gray,2,1,0.04)  
  
#result is dilated for marking the corners, not important*dst = cv2.dilate(dst,**None**)  
  
*# Threshold for an optimal value, it may vary depending on the image.*img[dst>0.01\*dst.max()]=[255]  
  
cv2.imshow(**'dst'**,img)  
  
**if** cv2.waitKey(0) & 0xff == 27:  
 cv2.destroyAllWindows()

**SCREENSHOT:**

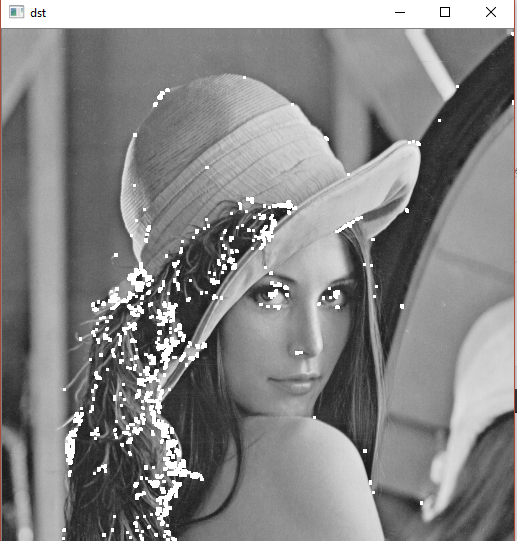
Blocksize=2\*2 , kernelsize=3\*3



Blocksize=10\*10 , kernelsize=3\*3



Blocksize=2\*2 , kernelsize=1\*1



**Task 2:  
CODE:**

**import** cv2 **as** cv  
img = cv.imread(**'lena.png'**)  
gray= cv.cvtColor(img,cv.COLOR\_BGR2GRAY)  
sift = cv.xfeatures2d.SIFT\_create()  
kp = sift.detect(gray,**None**)  
*#it will draw a circle with size of keypoint and it will even show its orientation*img=cv.drawKeypoints(gray,kp,img,flags=cv.DRAW\_MATCHES\_FLAGS\_DRAW\_RICH\_KEYPOINTS)  
cv.imshow(**'dst'**,img)  
**if** cv.waitKey(0) & 0xff == 27:  
 cv.destroyAllWindows()  
sift = cv.xfeatures2d.SIFT\_create()  
  
*#Here kp will be a list of keypoints and des is a numpy array of shape Number\_of\_Keypoints×128.  
# Here kp will be a list of keypoints and des is a numpy array of shape Number\_of\_Keypoints×128.*kp, des = sift.detectAndCompute(gray,**None**)  
print(**"Descriptors representation:"**,des)  
**for** a **in** kp:  
 print (a)

**EXPLANATION:**

The SIFT works as follows:

1. Scale-space Extrema Detection

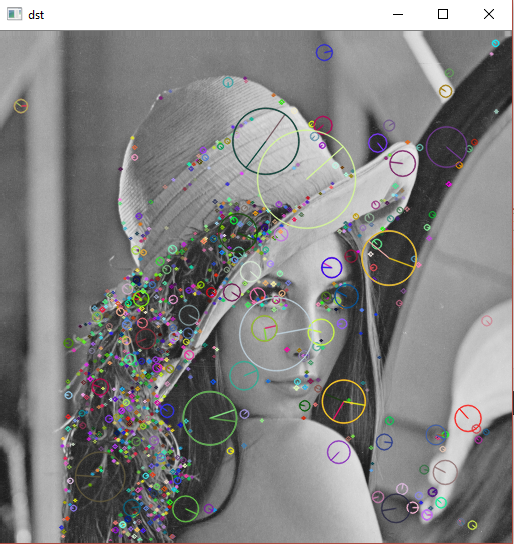
### 2. Keypoint Localization

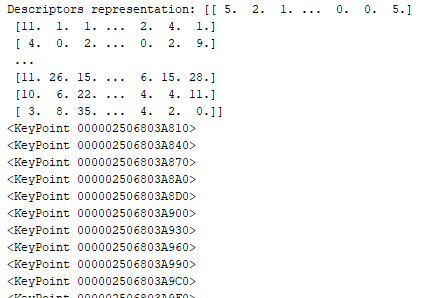
### 3. Orientation Assignment

### 4. Keypoint Descriptor

### 5. Keypoint Matching

**SCREENSHOT:**



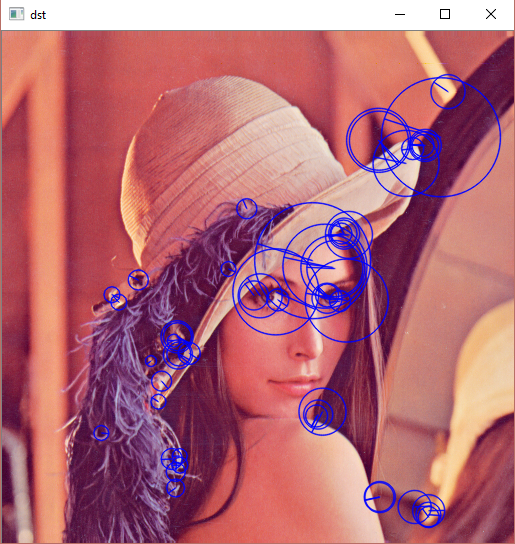


**Task 3:  
CODE:**

**import** cv2  
img = cv2.imread(**'lena.png'**)  
*# Create SURF object*surf = cv2.xfeatures2d.SURF\_create(5000)  
  
*# Here I decreased Hessian Threshold to 1000, which increases the no. of features found in the image.  
# surf = cv2.xfeatures2d.SURF\_create(1000)  
  
# Find keypoints and descriptors directly*kp, des = surf.detectAndCompute(img,**None**)  
  
img = cv2.drawKeypoints(img,kp,**None**,(255,0,0),4)  
  
cv2.imshow(**'dst'**,img)  
**if** cv2.waitKey(0) & 0xff == 27:  
 cv2.destroyAllWindows()  
  
print( **"Descriptor Size : "**,surf.descriptorSize() )  
print( **"No. of features : "**,len(kp) )

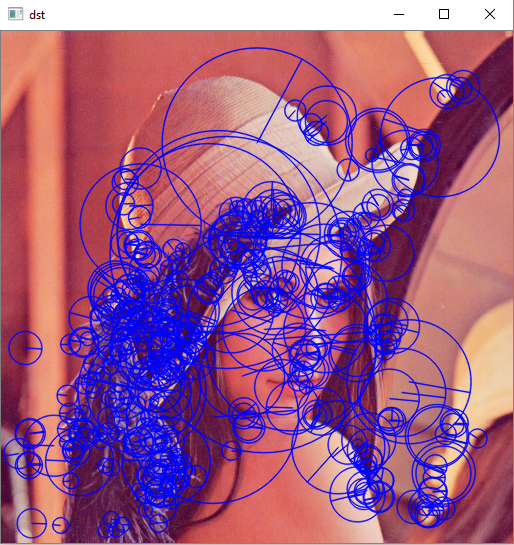
**SCREENSHOT:**

Hessian Threshold = 5000





Hessian Threshold = 1000

**Task 4,5:  
CODE:**

**import** cv2  
  
img = cv2.imread(**'lena.png'**)  
gray= cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)  
  
sift = cv2.xfeatures2d.SIFT\_create()  
kp = sift.detect(gray,**None**)  
*#it will draw a circle with size of keypoint and it will even show its orientation*img1=cv2.drawKeypoints(gray,kp,img,flags=cv2.DRAW\_MATCHES\_FLAGS\_DRAW\_RICH\_KEYPOINTS)  
cv2.imshow(**'dst'**,img1)  
**if** cv2.waitKey(0) & 0xff == 27:  
 cv2.destroyAllWindows()  
sift = cv2.xfeatures2d.SIFT\_create(2000) *#Hessian Threshold = 2000  
  
#Here kp will be a list of keypoints and des is a numpy array of shape Number\_of\_Keypoints×128.  
# Here kp will be a list of keypoints and des is a numpy array of shape Number\_of\_Keypoints×128.*kp, des = sift.detectAndCompute(gray,**None**)  
print( **"Descriptor Size (SIFT) : "**,sift.descriptorSize() )  
print( **"No. of features (SIFT): "**,len(kp) )  
  
*# Create SURF object*surf = cv2.xfeatures2d.SURF\_create(2000) *#Hessian Threshold = 2000  
  
# Find keypoints and descriptors directly*kp, des = surf.detectAndCompute(gray,**None**)  
  
img2 = cv2.drawKeypoints(gray,kp,img,flags=cv2.DRAW\_MATCHES\_FLAGS\_DRAW\_RICH\_KEYPOINTS)  
  
cv2.imshow(**'dst'**,img2)  
**if** cv2.waitKey(0) & 0xff == 27:  
 cv2.destroyAllWindows()  
  
print( **"Descriptor Size (SURF): "**,surf.descriptorSize() )  
print( **"No. of features (SURF): "**,len(kp) )

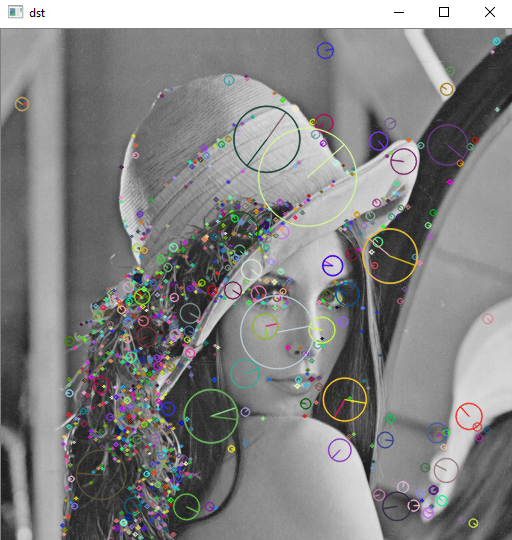
**EXPLANATION:**

In SIFT, Lowe approximated Laplacian of Gaussian with Difference of Gaussian for finding scale-space. SURF goes a little further and approximates LoG with Box Filter, which in-turn makes it faster than SIFT. Also the SURF rely on determinant of Hessian matrix for both scale and location.

SIFT achieves better results by determining more feature points than SURF, but it is slower. Also, the descriptor size of SIFT is 128 whereas SURF descriptor size is 64. So, the SIFT descriptor carries more information than SURF.

**SCREENSHOT:**

**SIFT**



**SURF**

