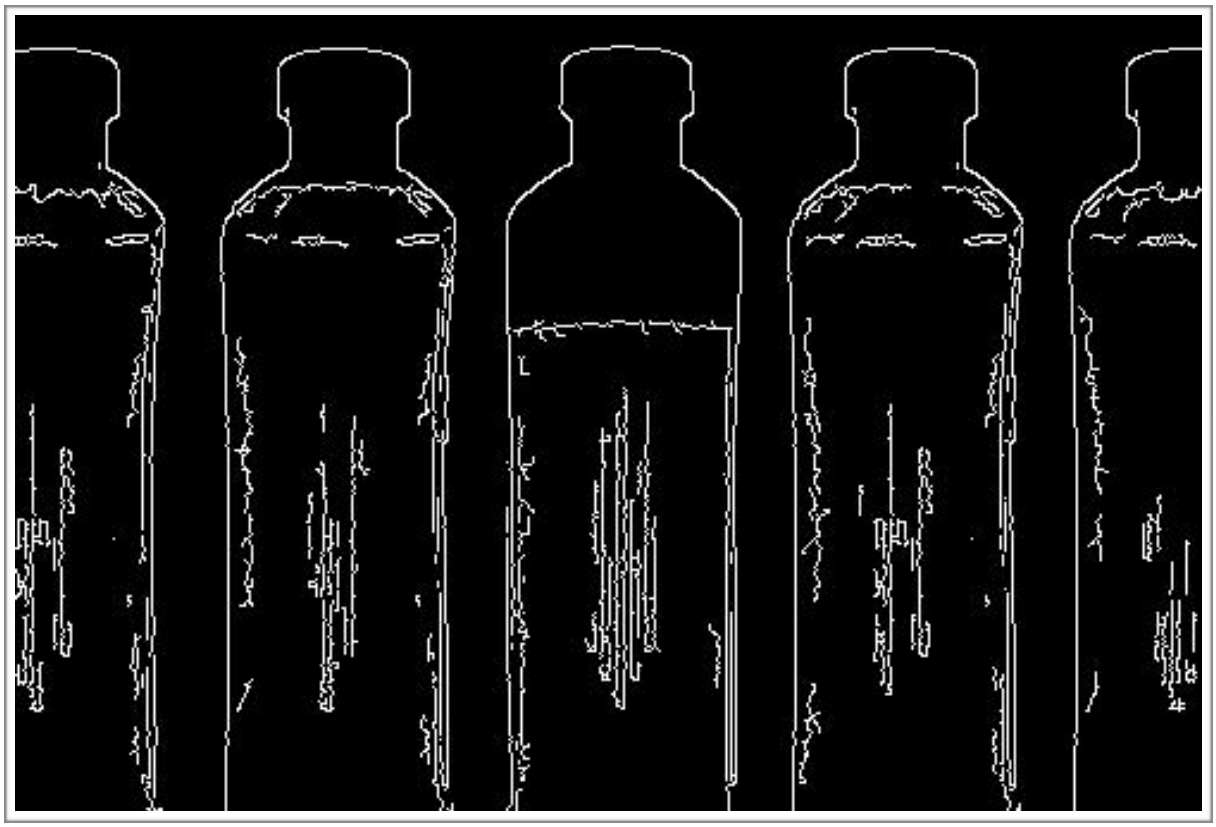
BLG 453E - Project 3 Report

Computer Vision



Tuğba Özkal

150120053

20.11.2017

BLG 453E - Project 3 Report

Computer Vision

1. Canny Edge Detection

Q1.py file is included the first question. It is compiled as:
`python3 Q1.py`

"canny" function is defined for edge detection. Its algorithm is given below.

```
def canny(I, sig, tau):

    M = gaussian_filter(I, sigma=sig)
    I = I.astype('int32')
    sobelx = ndimage.sobel(I, 0)
    sobely = ndimage.sobel(I, 1)

    row, col = I.shape

    E = np.zeros(shape=(row, col))
    A = np.zeros(shape=(row, col))

    for i in range(row):
        for j in range(col):
            E[i][j] = ((sobelx[i][j]**2) + (sobely[i][j]**2))**(1/2)
            if E[i][j] <= tau:
                E[i][j] = 0
            elif E[i][j] > 255:
                E[i][j] = 255

            if sobelx[i][j] != 0:
                A[i][j] = np.arctan(sobely[i][j]/sobelx[i][j])

                if A[i][j] < 22.5 and A[i][j] >= 0:
                    A[i][j] = 0
```

```
            elif A[i][j] < 67.5 and A[i][j] >= 22.5:
                A[i][j] = 45
            elif A[i][j] < 112.5 and A[i][j] >= 67.5:
                A[i][j] = 90
            elif A[i][j] < 157.5 and A[i][j] >= 112.5:
                A[i][j] = 135
            elif A[i][j] <= 180 and A[i][j] >= 157.5:
                A[i][j] = 0

    for i in range(1, row-1):
        for j in range(1, col-1):
            if A[i][j] == 0:
                if E[i][j] <= E[i-1][j] or E[i][j] <= E[i+1][j]:
                    E[i][j] = 0
            elif A[i][j] == 45:
                if E[i][j] <= E[i+1][j+1] or E[i][j] <= E[i-1][j-1]:
                    E[i][j] = 0
            if A[i][j] == 90:
                if E[i][j] <= E[i][j-1] or E[i][j] <= E[i][j+1]:
                    E[i][j] = 0
            elif A[i][j] == 135:
                if E[i][j] <= E[i-1][j+1] or E[i][j] <= E[i+1][j-1]:
                    E[i][j] = 0

    return E, M, A;
```

Firstly, gaussian filter is applied to image with sigma. And then by using sober, edges in x axis and y axis are detected.

$$g(m,n) = G_\sigma(m,n) * f(m,n)$$

where

$$G_\sigma = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{m^2+n^2}{2\sigma^2}\right)$$

Then, gradient magnitude is computed by using the formula.

$$M(n,n) = \sqrt{g_m^2(m,n) + g_n^2(m,n)}$$

For all points, angles are computed.

$$\theta(m,n) = tan^{-1}[g_n(m,n)/g_m(m,n)]$$

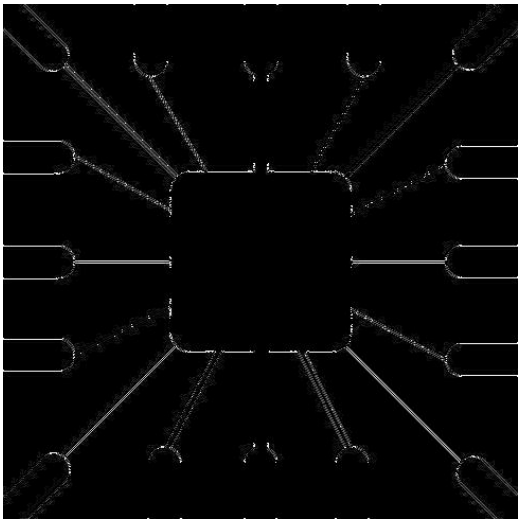
If a magnitude is bigger the 255, it is set to 255. If it is smaller than threshold, it is set to zero.

$$M_T(m,n) = \begin{cases} M(m,n) & \text{if } M(m,n) > T \\ 0 & \text{otherwise} \end{cases}$$

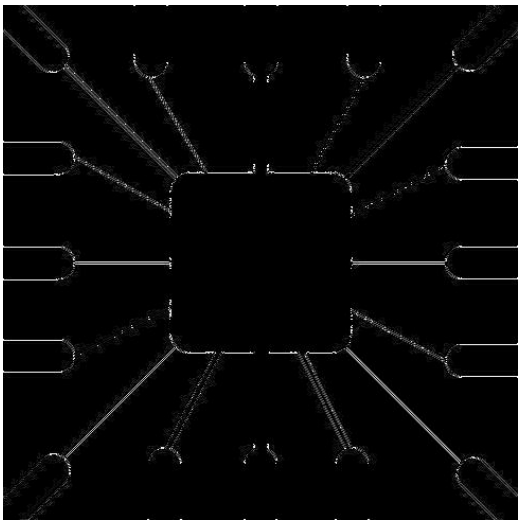
All angles are placed in interval such as 22,5 to 67,5 or 67,5 to 112,5 etc. According to these angles, non-maxima pixels in edges suppressed. After this, edges get thinner.

"canny" function has 3 parameters: image as I, sigma as sig and threshold as tau. It returns the E (detected edge), M (smoothed gradient magnitude) and A (gradient angle).

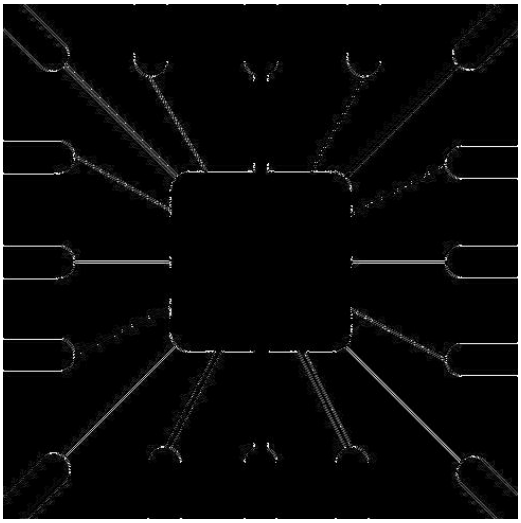
For Fig2wirebond_mask.jpg image with the parameters sig=0.5 and tau=5, result is:



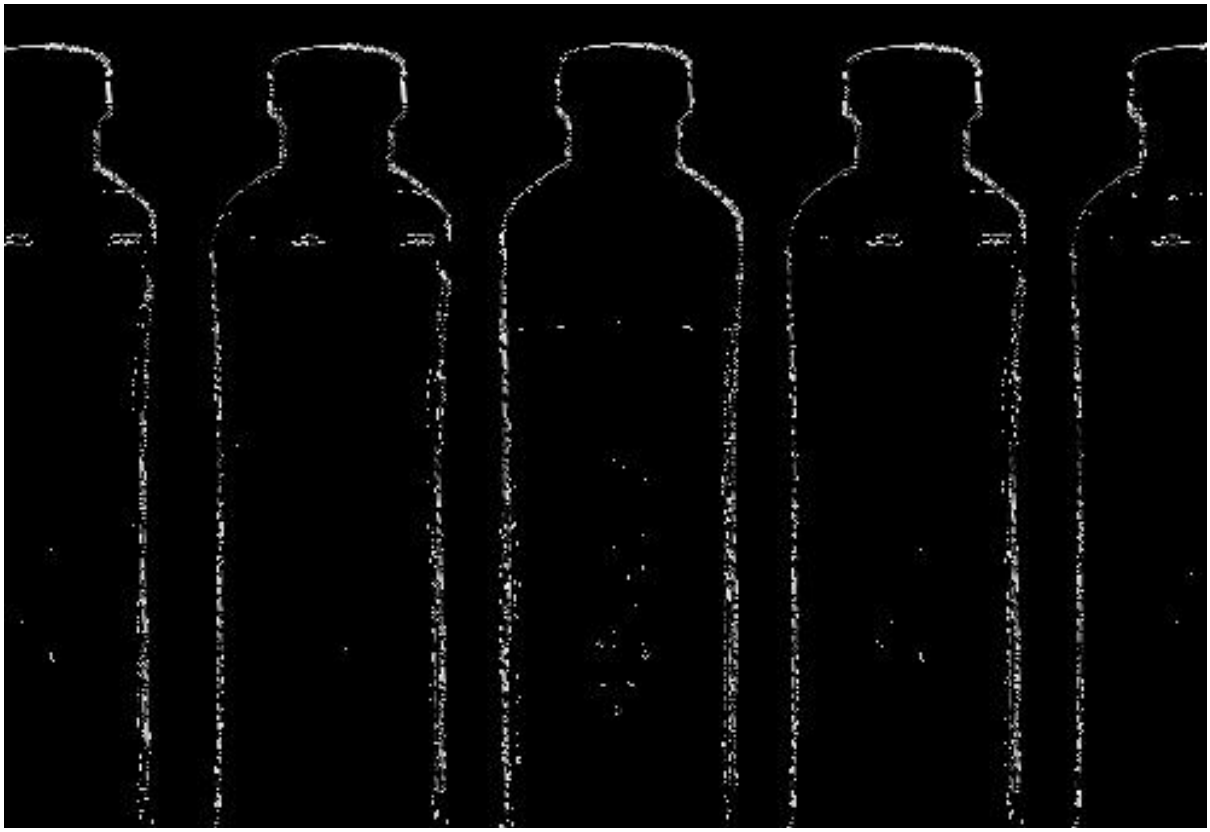
For Fig2wirebond_mask.jpg image with the parameters sig=1 and tau=5, result is:



For Fig2wirebond_mask.jpg image with the parameters sig=3 and tau=5, result is:

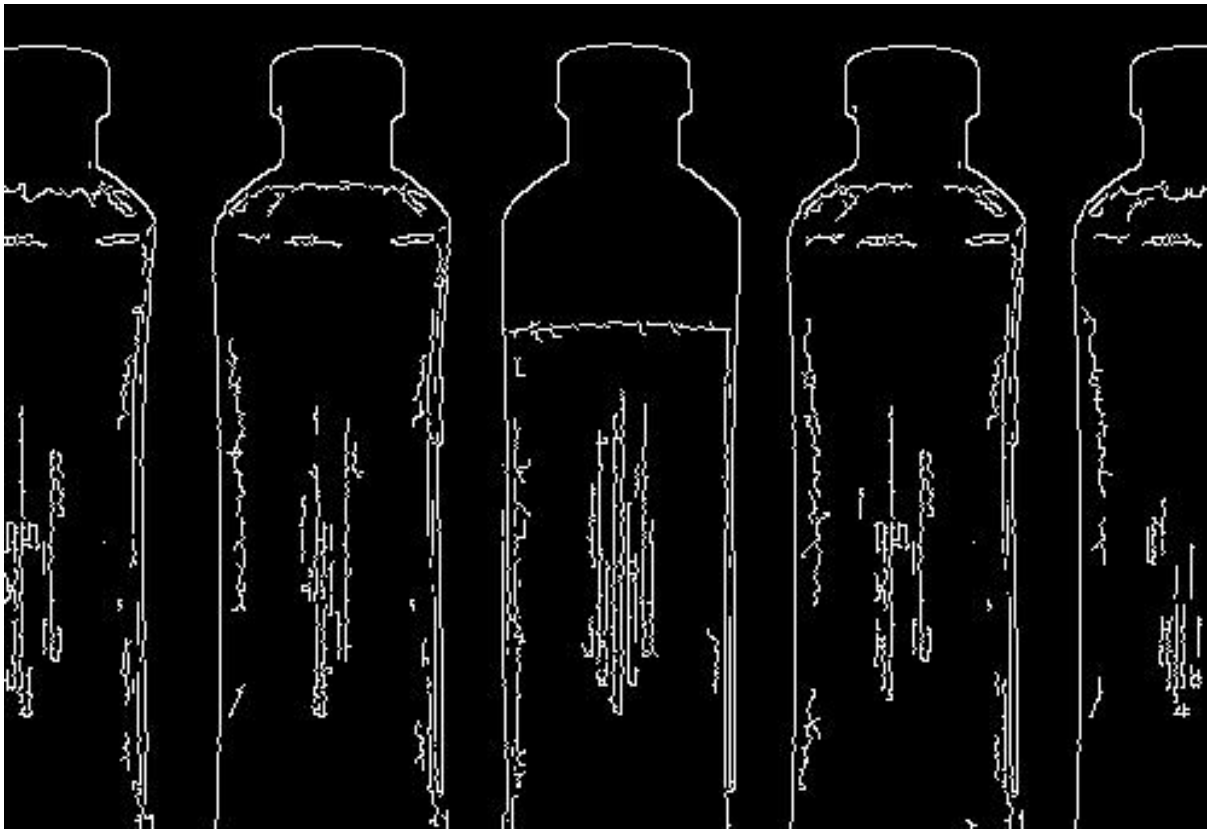


If we apply the canny edge detection function on Fig3bottles.jpg image with sig=7.5 and tau=200, we have the result:



I found the best result by trying. High sigma value and high threshold help to detect real edges. In example, liquid in bottles is lost for this image by using high sigma and threshold.

Result of canny function of OpenCV library:



2. Feature Matching using SIFT

Firstly, keypoints and descriptors are found by using SIFT. Then, BFMatcher.knnMatch() is used to get k best matches. In this question, I take k=2 so that we can apply ratio test.

cameraman1.jpg

cameraman2.jpg

Algorithm:

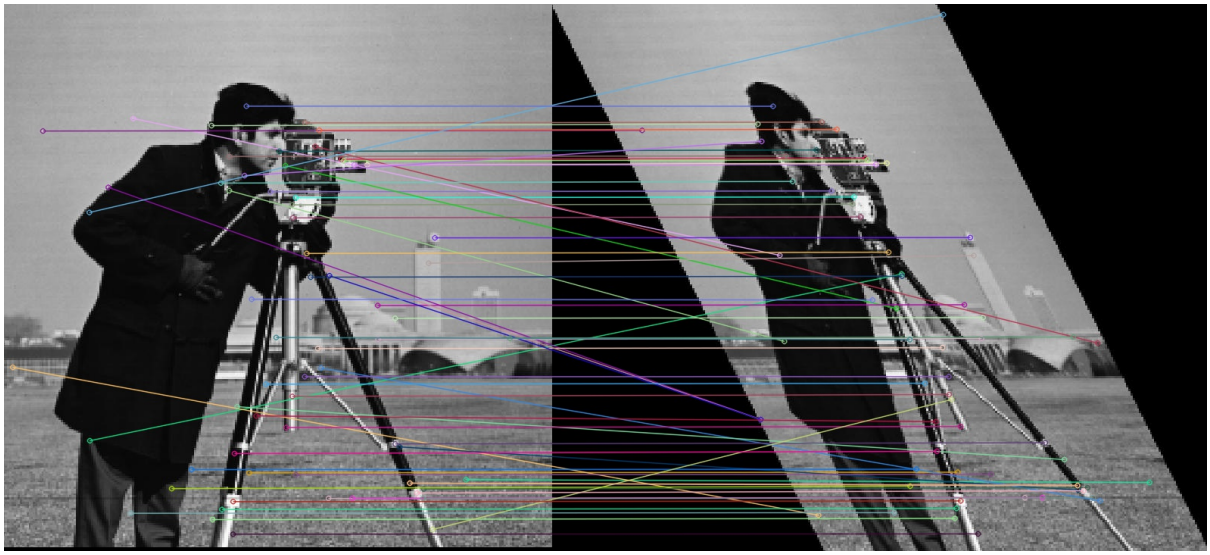
```
sift = cv2.xfeatures2d.SIFT_create()
kp1, des1 = sift.detectAndCompute(gray1,None)
kp2, des2 = sift.detectAndCompute(gray2,None)

bf = cv2.BFMatcher()
matches = bf.knnMatch(des1,des2, k=2)

good = []
for m,n in matches:
    if m.distance < 0.75*n.distance:
        good.append([m])

img3 = cv2.drawMatchesKnn(img1,kp1,img2,kp2,good, None, flags=2)

img1=cv2.drawKeypoints(gray1,kp1, img1)
img2=cv2.drawKeypoints(gray2,kp2, img2)
```



matched points

The matched points of two image is given above. Results are written as image which have names "sift_keypoints1.jpg", "sift_keypoints2.jpg" and "matchedpoint.jpg" seperately.

2d affine transform matrix is estimated.

Transformation matrixes are given below.

pts1 = [[0,0],[100,50],[50,200]]

pts2 = [0,0],[100,50],[130,200]]

```
row, col = gray1.shape
pts1 = np.float32([[0,0],[100,50],[50,200]])
pts2 = np.float32([[0,0],[100,50],[130,200]])

M = cv2.getAffineTransform(pts1,pts2)
dst = cv2.warpAffine(gray1,M,(col+143,row))

plt.subplot(121),plt.imshow(gray1),plt.title('Input')
plt.subplot(122),plt.imshow(dst),plt.title('Output')
plt.show()
```

The result is here.