ROB313-Tracking

Mengyu PAN

January 2021

1 Introduction

Dans ce TP, on implémente les algorithmes de Mean Shift et de Transformées de Hough Généralisées pour bien comprendre leur fonctions et leur différences. Il y a cinq questions dans ce TP. Les deux permières sont l'implémentation du Mean Shift et les autres sont les Transformées de Hough.

2 Mean Shift

2.1 Q1

L'algorithme de Mean Shift est l'algorithme pour l'apprentissage automatique. D'abord, il calcule la centroïde des donées et son target de décalage. Ensuite, il se déplace à son target de décalage. Puis il fait l'itération de permière étape et deuxième étape jusqu'à l'arrive de son déstination. Le résultat est indiqué dans la figure 1. L'advantage de MeanShift : 1) il y a moins de calcul; 2) le temps d'éxecution est petit 3) il peut suivre l'objet rapidement L'inconvénient est qu'il ne peut pas suivre l'objet précisément. Si l'objet se deplace très rapidement, le fenetre perd son objet.

2.2 Q2

Je trouve que dans l'image de rétroprojection les pixels de la porte ont les valeur plus élevés que les pixels de tasse. Donc, il perd son objet quand il suit la tasse. Pour amélioer, je change le canal de HSV de canal H(hue) à S(saturation). Le résultat est dans la figure 2.

```
#la fonction principale
roi_hist = cv2.calcHist([hsv_roi],[1],mask,[180],[0,180])
```

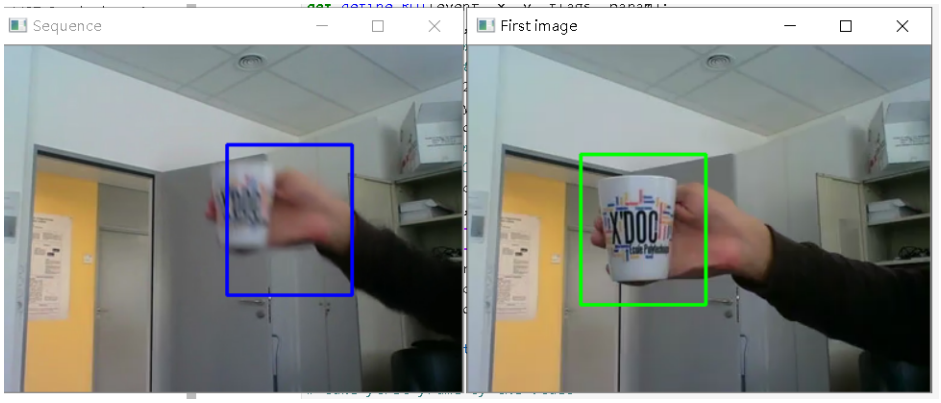

Figure 1: Résultat de Q1

Figure 2: Résultat de Q2

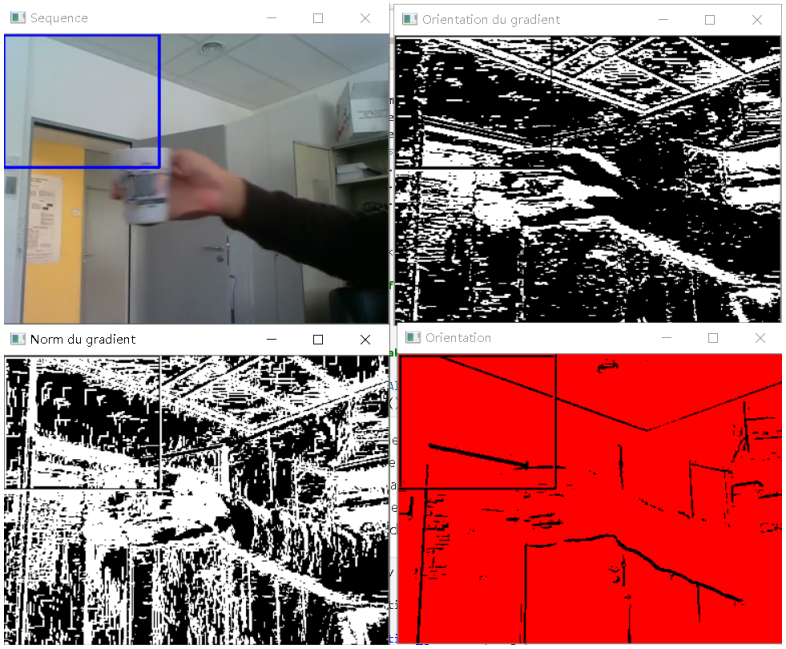

Figure 3: Résultat de Q3

3 Transformées de Hough

3.1 Q3

Pour calculer l'orientation du gradient, il faut calculer la direction horizontale et la direction verticale avec deux coeurs. Ensuite, j'utilise la fonction cv2.magnitude() pour calculer le norm du gradient. Puis, je peinte les pixels qui sont moins que 50 avec la couleur rouge.

Donc il y a quatre images affichantes dans la figure 3 : 1) "sequence image", 2)"orientation du gradient", 3)"norm du gradient", 4)"orientation" (image avec mask).

3.2 Q4

Pour achievez la transformation hough,je définis quatre fonctions:

1 orientation gradient(image): calculez l'orientation du gradient, returnez l'angle (dégrée)

```
def orientation_gradient(image):
    Image = image
    #orientation x
    kernel = np.array([[-1, 0, 1],[-1, 0, 1],[-1, 0, 1]])
    Ix = cv2.filter2D(image,-1,kernel)
```

```
    Ix = np.array(Ix, np.float32)
    #oritentaion y
    kernel = np.array([[-1, -1, -1],[0, 0, 0],[1, 1, 1]])
    Iy = cv2.filter2D(image,-1,kernel)
    Iy = np.array(Iy, np.float32)
    #angle calculation
    angle = cv2.phase(Ix,Iy,angleInDegrees=True)
    angle = angle*180/np.pi #change to degree
    return angle
```

2 RTable(image,ref): il faut un image grey et un point de réference. D'about il calcule l'orientation du gradient avec la fonction orientation gradient(image). Ensuite, il construit une RTable pour ceux qui ont la même angle avec un vecteur entre ce point et le point de réference.

```
def RTable(image,ref):#need gray image and reference point(x,y)
    Image = image
    edge = cv2.Canny(Image,10,1000)#find the line in the image
    angle = orientation_gradient(image)
    rTable = defaultdict(list)
    for (i,j),value in np.ndenumerate(edge):
        rTable[angle[i,j]].append((ref[0]-i,ref[1]-j))#store the
            vector in each angle
    return rTable
```

3 detection(image,rTable): il faut utiliser le résultat de fonction RTable(image,ref). Il trouve la similarité entre cet image et la RTable de l'autre image par l'orientation du gradient.Pour tous les points qui ont la même orientation, augmentez la graphique de vote.

```
def detection(image,rTable):#construct the vote map H(x)
    Image = image
    edge = cv2.Canny(image,10,1000)#find the line in the image
    angle = orientation_gradient(image)

    hough = np.zeros(image.shape)
    for (i,j),value in np.ndenumerate(edge):
        for r in rTable[angle[i,j]]:
            x = r[0] + i
            y = r[1] + j
            if x < hough.shape[0] and y < hough.shape[1]:
                hough[i,j] += 1 #suppose weight is 1
    return hough
```

4 Hmax(hough): selectez la position plus possible par choisir arg max H(x), retournez la valeur maximale et la position(x,y).

Figure 4: Résultat de Q4

```
def Hmax(hough):#find the arg max H(x)
    indices = hough.ravel().argsort()[-1:]
    indices = (np.unravel_index(i, hough.shape) for i in indices)
    return [(hough[i], i) for i in indices]
```

Il faut prendre beaucoup plus de temps pour afficher le résultat et le résultat obtenu est affiché dans la figure 4. Malheursement, le code est toujours bloqué pendant l'éxecution donc je sais seulement que au début il ne suit bien que la méthode Mean-Shift.

3.3 Q5

Je remplace le calcul du maximum par l'application du Mean Shift sur la transformée de Hough comme le code indique mais le résultat obtenu n'est pas meilleur que le résultat in Q4 car le code est bloqué donc je ne vois pas trop de différence.

```
frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
hough = detection(frame_gray,rTable)

# apply meanshift to dst to get the new location
ret, track_window = cv2.meanShift(hough, track_window, term_crit)
```