

ROB313 -TP3 : Analyse vidéo et Tracking

ARAUJO BELÉN, Victor
SULA, Julia

Janvier 2020

1 Mean Shift

- 1.1 Expérimenter le suivi réalisé par le code de base Tracking_MeanShift.py fourni qui utilise l'algorithme de Mean Shift, avec la densité marginale sur la composante H de teinte. Rappeler le principe de l'algorithme Mean Shift, et illustrer par vos expériences ses avantages et ses limitations.**

1.1.1 Mean Shift Algorithme

L'algorithme *mean-shift*, est un algorithme qui explore un *frame*, afin de définir le vecteur de déplacement d'un objet entre les *frames*.

Le principe de l'algorithme est de modeler l'objet à suivre, à travers des histogrammes. Lorsqu'il y a un changement de *frame*, la *BackPropagation* est réalisée. Cette dernière permet de calculer la similarité entre les *pixels* d'un *frame* et le modèle créé précédemment, le résultat est une fonction de probabilité pour qu'un pixel fasse partie de l'objet à suivre. Cette fonction peut être vue aussi comme une fonction de densité, ainsi les régions les plus similaires, sont les plus denses, donc à chaque *frame* il y a un déplacement du centre de masse de l'objet à suivre, car il y a eu un changement de la densité dans la région d'intérêt.

1.1.2 Mean Shift Algorithme : Expérimentation

En utilisant le code fourni, nous avons pu observer le *tracking* dans les différents vidéos. Chaque vidéo présente ses propres spécificités, qui deviennent claires pendant le processus de *tracking*.

Antoine_Mug

Cette vidéo nous permet d'observer que si la région d'intérêt n'était pas bien choisie, le *tracking* pourrait diverger. Selon les principes de *Mean-shift*, nous

avons conclu qu'il s'agit possiblement d'une conséquence de le histogramme de la région, autrement dit, si nous choisissons une région qui prenne en considération uniquement la tasse, il est possible que son histogramme soit confondu avec le poster derrière. Ainsi, les régions qui permettraient d'améliorer le *tracking* ont été celles qui incluaient la tasse et la main, voir figure 1.

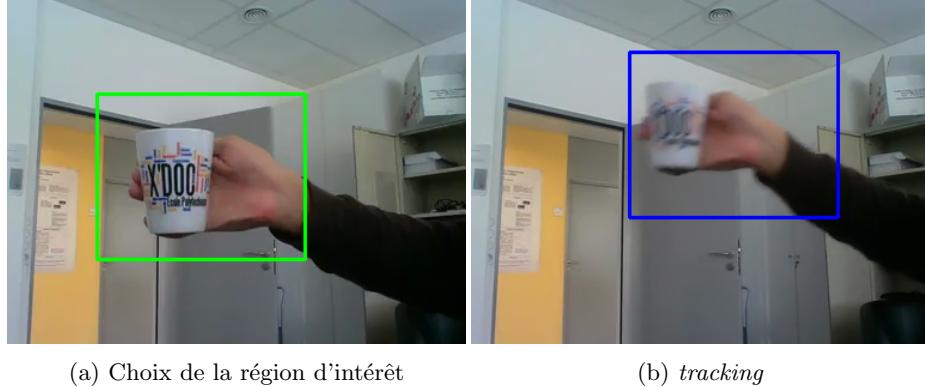


FIGURE 1 – Test de l'algorithme *MeanShift*

VOT-Ball

Dans cette vidéo, de manière similaire à l'antérieure, le choix de la zone d'intérêt a un grand impact dans le *tracking*.

Cependant, dans ce cas la balle a un composant de teinte plus fort que celui de la main de *Antoine.Mug*, par conséquent, le *tracking* est plus efficace, car la différence entre les objets dans le domaine *HSV* est plus grande, et ceci permet d'améliorer la distinction des probabilités projetées .

Une des frames du *tracking* est représentée dans la 4.

VOT-Basket

Cette vidéo possède une particularité, car elle présente des "objets" similaires en mouvement, dans ce cas le joueurs de basket. En conséquence, le *tracking* parfois change de joueur en joueur, et devient instable. De plus, les joueurs changent de positionnement très souvent, et pour le processus de *tracking*, ceci peut être interprété comme une déformation de l'objet. Ces déformations empêchent que l'objet soit facilement identifié, en gênant le *tracking*. Un exemple est montre dans la figure 3

VOT-Car

La vidéo *VOT-Car*, est une vidéo tremblant, par conséquent, les changements des positions de la voiture à suivre sont abruptes. Cela fait que, dans le meilleur



(a) Choix de la région d'intérêt

(b) *tracking*

FIGURE 2 – Test de l'algorithme MeanShift



(a) Choix de la région d'interet

(b) *tracking*

FIGURE 3 – Test de l'algorithme MeanShift

des cas, le *tracking* soit imparfait, car l'algorithme *Mean shift* nécessite par principe d'un consensus du déplacement (nous pouvons le décrire comme le calcul du déplacement moyen à chaque *frame*), par conséquent, les changement abruptes ne sont pas bien traités à travers cet algorithme.

VOT-Sunshade

En testant le code fourni dans cette vidéo, nous pouvons rapidement observer que le *tracking* n'est pas efficace dans le zone obscure.

Ceci démontre les limites de l'invariance au changement de la lumière, qui est fournie par l'utilisation de la composante de teinte du domaine *HSV*. Autrement dit, malgré la réduction de la sensibilité apportée par ce choix, le changement des *luma*, et la saturation présentée dans cette vidéo altère la teinte suffisamment

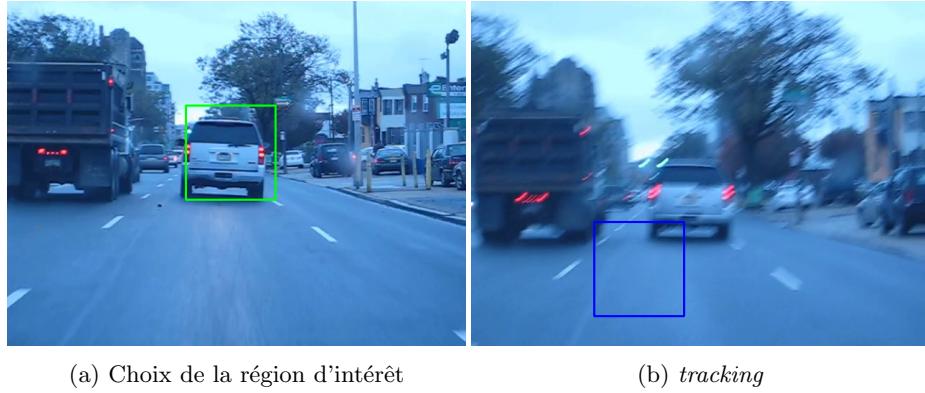


FIGURE 4 – Test de l'algorithme MeanShift

pour empêcher la reconnaissance de l'objet dans la zone obscure. Cela peut-être observé dans la figure 5.

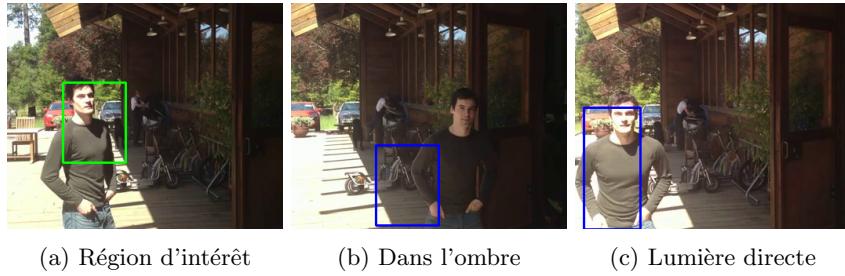
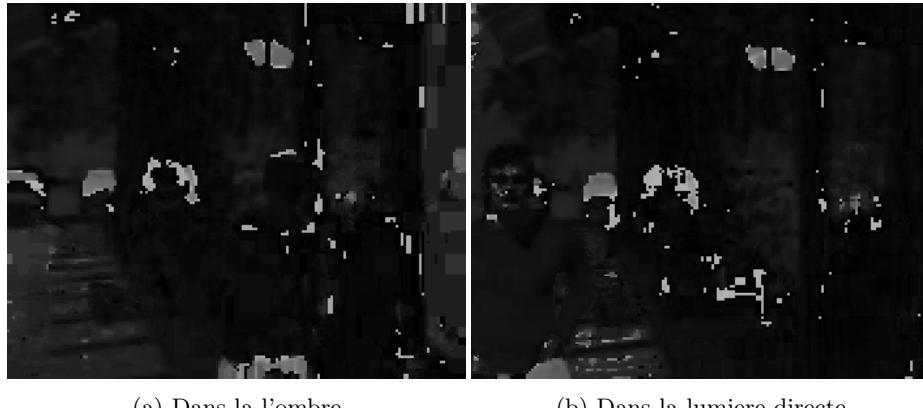


FIGURE 5 – Test de l'algorithme *MeanShift*

Cette altération de teinte dans les différences lumières peut-être vue dans les figures 6 .

VOT-Woman

Finalement, dans la dernière vidéo, l'algorithme *MeanShift* s'est montré efficace, et ceci a permis de suivre la femme dans la plupart du temps.



(a) Dans la l'ombre

(b) Dans la lumiere directe

FIGURE 6 – Test de l'algorithme MeanShift



(a) Choix de la région d'interet

(b) *tracking*

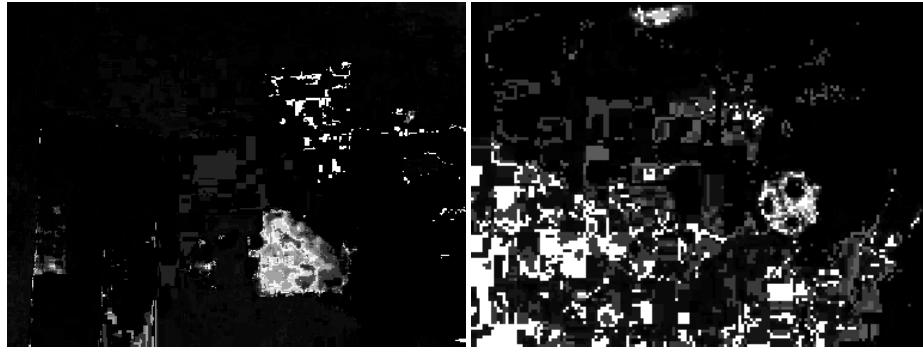
FIGURE 7 – Test de l'algorithme MeanShift

1.2 Analyser plus finement le résultat en affichant la séquence des poids correspondant à la rétro-projection de l'histogramme de teinte. Proposer et programmer des améliorations, en changeant la densité calculée et/ou en mettant en oeuvre une stratégie de mise à jour de l'histogramme modèle.

1.2.1 Séquence de poids

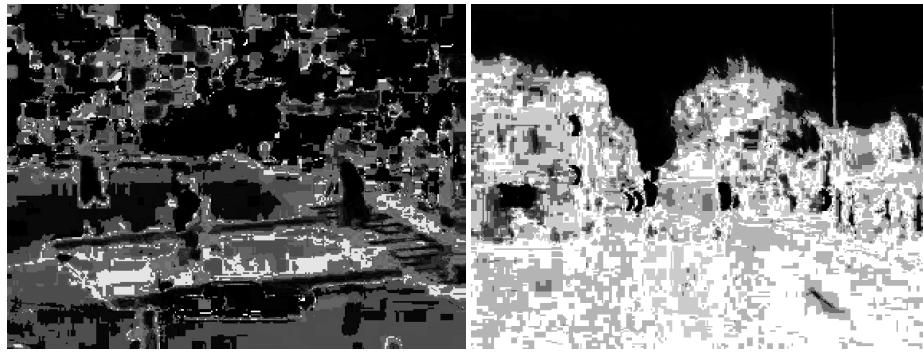
Il est possible d'observer le résultat de la rétro-projection des différentes vidéos dans les figures 8a,8b,9a,9b ,10a et 10b. En analysant ces figures, nous

pouvons observer que dans le cas de vidéos *Antoine Mug*, *VOT-Woman* et *VOT-Ball*, la région de plus haute densité est facilement reconnue. Cette région constitue l'objet à suivre dans la vidéo. Autrement dit, la plus haute probabilité est dans l'objet à suivre. Cependant pour les autres vidéos, la région de plus haute densité n'est pas évidente, ceci explique pourquoi l'algorithme ne réussit pas à suivre les objets indiqués.



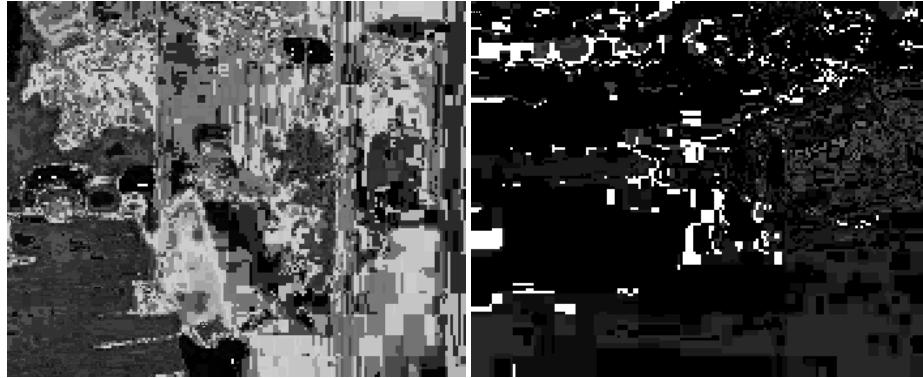
(a) Rétro-projection de l'histogramme de teinte Vidéo Antoine Mug (b) Rétro-projection de l'histogramme de teinte Vidéo VOT Ball

FIGURE 8 – Rétro-projection



(a) Rétro-projection de l'histogramme de teinte Vidéo VOT-Basket (b) Rétro-projection de l'histogramme de teinte Vidéo VOT-Car

FIGURE 9 – Rétro-projection



(a) Rétro-projection de l'histogramme de teinte Vidéo VOT-Sunshade (b) Rétro-projection de l'histogramme de teinte Vidéo VOT-Woman

FIGURE 10 – Rétro-projection

1.2.2 Améliorations

Mise à jour

La première amélioration mise en oeuvre, a été la mise à jour du modèle de l'objet (l'histogramme) si la valeur de la somme des *dst* devenait plus basse qu'un certain seuil. Cette amélioration permet de raffiner l'histogramme, et de le rendre plus robuste aux déformations de l'objet à suivre. Cependant, si le seuil n'est pas bien choisi le *tracking* peut devenir instable.

Le code pour cette mise à jour est présenté ci-dessous.

```
if ret == True:
    if(abs(np.sum(dst))<100000):
        roi = frame[x:x+w,y:y+h]
        hsv_roi = cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)
}
```

Minima et Maxima de la rétro-propagation

Une deuxième amélioration implémentée, est la division des seuils de la rétro-projection , c'est-à-dire , ne pas considérer les valeurs trop petites en ajoutant plus de précision au *tracking*.

Dans le cas de la vidéo *VOT-Ball*, cette implementation à permis d'augmenter la précision de suivre en éliminant les moments de dérive, où la suivie était erronée.

1.2.3 Autres

Une deuxième amélioration envisagée, était d'utiliser aussi les informations de texture pour raffiner le *tracking*. Ceci serait possible en utilisant des algorithmes comme le *Local binary pattern (LBP)*.

Encore plus loin, nous pourrions implémenter la méthode de *STRUCK (Structured Output Tracking with Kernels)*, basée sur la classification d'image par l'apprentissage à travers l'algorithme de *SVM*. Il s'agit d'un classifier qui prend les caractéristiques du fond de l'objet, et à travers une fonction de prédiction, il prédit la transformation de l'objet dans le plan 2D à un instant donné.

Un modèle un peu plus simple pourrait être le *tracker DPM (Deformable Part Models)*, qui est la combinaison de modèles déterministes utilisés pour l'entraînement, et un filtre de Kalmann, et ainsi pouvoir réaliser une prédiction par une détection sémantique. Dans le processus de pré-entraînement, il est possible de dénifir une quantité d'objets par classe : animal (types d'animaux), véhicule (types de véhicule). Ceci serait un approche plus limité pour les vidéos au hasard, mais plus efficace pour les tâches bien cyclées. L'idée est de représenter chaque objet en sous-parties. Un modèle peut avoir plusieurs point de vues, ce qui permettrait de représenter une vue de face et une de profil, par exemple.

2 Transformée de Hough

2.1 Calculer à chaque trame, l'orientation locale, i.e. l'argument du gradient des *pixels* de l'image, ainsi que le module du gradient. Définir un seuil sur le module du gradient pour masquer les *pixels* dont l'orientation n'est pas significative. Afficher ainsi la séquence des orientations où les *pixels* masqués apparaissent en rouge

Premièrement, pour calculer l'orientation locale d'une image, nous avons défini dans le code, les dérivées partielles comme il est montré ci-dessous :

```
img=cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
img6 = cv2.copyMakeBorder(img,0,0,0,0,cv2.BORDER_REPLICATE)
img7 = cv2.copyMakeBorder(frame,0,0,0,0,cv2.BORDER_REPLICATE)
img11=cv2.copyMakeBorder(frame,0,0,0,0,cv2.BORDER_REPLICATE)
img10=cv2.copyMakeBorder(frame,0,0,0,0,cv2.BORDER_REPLICATE)

kernel1 = np.array([[-1,0,1],[-2,0,2],[-1,0,1]])
img11 = cv2.filter2D(img,-1,kernel1)
kernel2 = np.array([[ -1,-2,-1],[0,0,0],[1,2,1]])
img10 = cv2.filter2D(img,-1,kernel2)
```

Ensuite, le calcul du gradient a été fait comme il est montré ci-dessous.

```
img6= np.hypot(img11,img10)/255
```

Et finalement, nous avons calculé l'argument du gradient.

```
img7=2*np.arctan2(np.float32(img11),np.float32(img10))
```

À travers le calcul du gradient, il est possible d'observer que pour la majorité de l'image, la valeur du gradient n'est pas significative. Autrement dit, il n'y a pas une forte variation du niveau de gris autour d'un point.

Cela nous permet de simplifier la représentation des gradients, et de déterminer les *pixels* qui seront pris en compte pour le calcul du *tracking*. Ainsi, un seuil a été choisi pour déterminer ces *pixels* : tous les *pixels* de module de gradient plus petit que 0.1 seront considérés non significatifs.

Cette considération permet de passer de l'image dans la figure 11a à la figure 11b, où seulement une petit pourcentage des pixels est défini significatif.

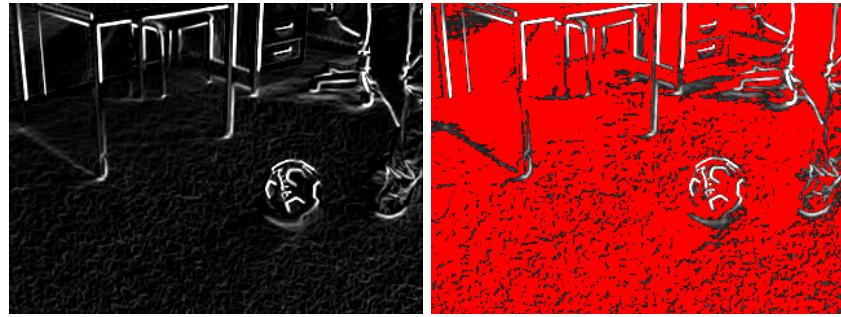


(a) Calcul de gradient de chaque frame, en considérant tous les pixels (b) Calcul de gradient, les pixels en rouge ont le module plus petit que 0.1

FIGURE 11 – Calcul de gradient des frames

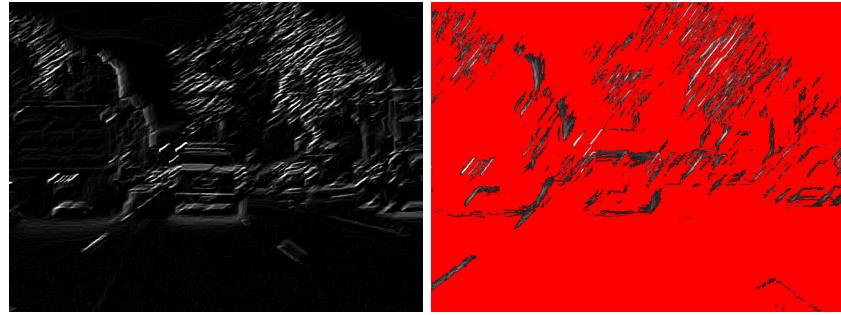
Le même calcul a été réalisé pour chaque vidéo les résultats sont montré dans la suite.

Le calcul des orientations du gradient permet de vérifier les directions de changement du niveau de gris. Dans la figure 17, nous pouvons observer l'histogramme de cette orientation, c'est-a-dire, le nombre de *pixels* qui ont un certain angle.



(a) Calcul de gradient de chaque frame, (b) Calcul de gradient, les *pixels* en rouge ont le module plus petit que 0.1 en considérant tous les pixels

FIGURE 12 – Calcul de gradient des *frames*



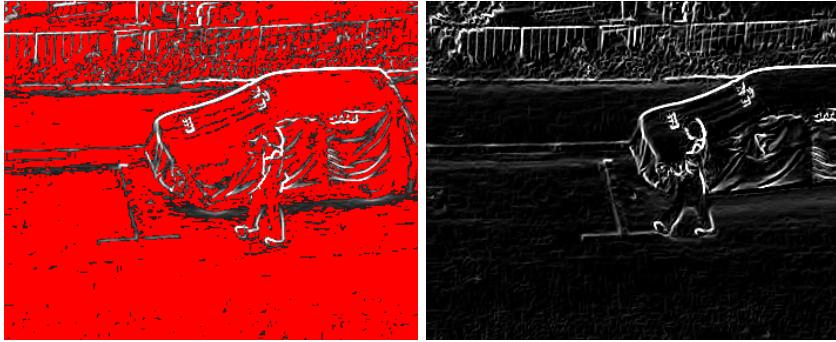
(a) Calcul du gradient de chaque frame, en considérant tous les *pixels* (b) Calcul du gradient,les *pixels* en rouge ont le module plus petit que 0.1

FIGURE 13 – Calcul du gradient des *frames*



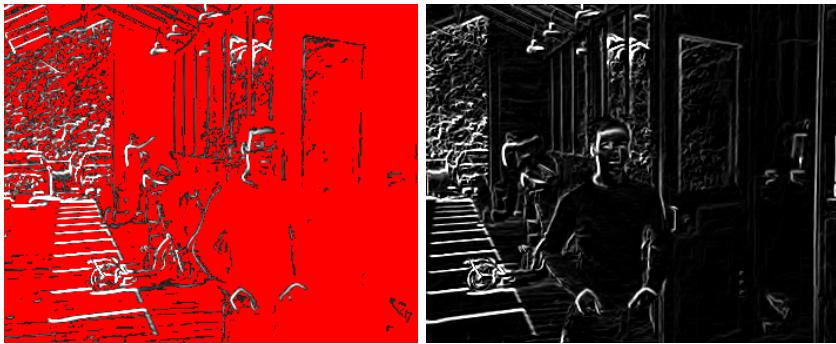
(a) Calcul du gradient de chaque frame, en considérant tous les *pixels* (b) Calcul du gradient,les *pixels* en rouge ont le module plus petit que 0.1

FIGURE 14 – Calcul du gradient des *frames*



(a) Calcul du gradient de chaque frame, en considérant tous les pixels
 (b) Calcul du gradient, les pixels en rouge ont le module plus petit que 0.1

FIGURE 15 – Calcul de gradient des frames



(a) Calcul de gradient de chaque frame, en considérant tous les pixels
 (b) Calcul du gradient, les pixels en rouge ont le module plus petit que 0.1

FIGURE 16 – Calcul du gradient des frames

2.2 Construire un modèle de l'objet défini initialement sous la forme d'un modèle implicite indexé sur l'orientation (R-Table). Puis calculer la transformée de Hough associée sur toutes les images de la séquence. Calculer le suivi correspondant à la valeur maximale de la transformée de Hough à chaque image. Commenter et critiquer le résultat obtenu.

La construction de la R-Table a été faite en 2 parties. La première partie a été la définition d'un premier tableau qui contient l'argument du gradient, et la position des pixels. Cela a été fait à travers le code montré ci-dessous.

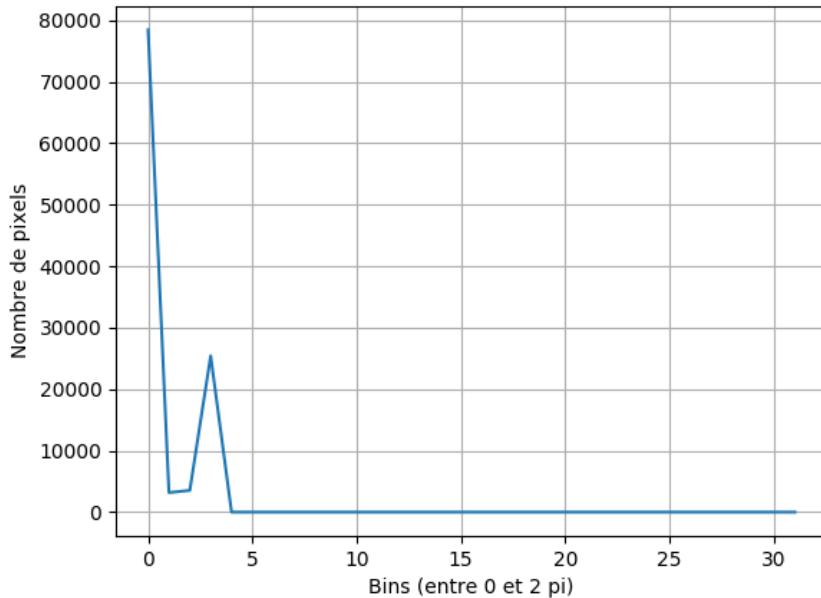


FIGURE 17 – Histogramme des orientations du gradient

```

roi_Rtable = img7[c:c+w, r:r+h]
mod_Roi=img6[c:c+w, r:r+h]
l=+1
imgf=cv2.cvtColor(np.float32( mod_Roi ), cv2.COLOR_GRAY2BGR)
imgf[mod_Roi<0.1]=[0, 0, 255]
img_taille=mod_Roi[mod_Roi >epsilon]
#print(np.shape(img_taille))

preTable=np.zeros((np.shape(img_taille)[0], 3))
print(np.shape(mod_Roi))
print(np.shape(roi))
orientation=np.zeros(np.shape(roi_Rtable
    )[0]*np.shape(roi_Rtable )[1])
for i in range (0, np.shape(roi_Rtable )[0]-1):
    for j in range (0, np.shape(roi_Rtable )[1]-1):
        if(mod_Roi[i,j]>epsilon):
            #print(k)
            preTable[k,:]=[roi_Rtable[i,j], i, j]
            orientation[k]=roi_Rtable[i,j]
            k+=1

```

Dans la deuxième partie, à travers du tableau d'orientation, nous avons

calculé tous les angles uniques . Cela nous permet de construire la R-Table à partir du code ci-dessous.

```

angle= list(set(orientation)) #unique values of orientation
print('unique'+str(np.shape(angle)))
rTable1=[]
rTable=[]
l=0

print(np.shape(preTableSorted))
for i in angle:
    rTable1.append(i)
    l+=1
    for j in range (0, np.shape(preTableSorted)[0]):
        if(i==preTableSorted[j,0]):
            rTable1.append((preTableSorted[j,1]-c,
                           preTableSorted[j,2]-r))
            #print(rTable)
            #print(local)
    rTable.append(rTable1)
    rTable1=[]
print((rTable[1]))

```

Pendant ce calcul il est important d'observer que les seuls *pixels* considérés ont été ceux dont le module du gradient était plus grand que le seuil. De plus, le modèle de l'objet ne considère que les *pixels* qui appartient à la région d'intérêt.

Dans la suite, pour les transformées de *Hough*, le procédure suivie est décrite ci-dessous :

- Pour chaque pixel de l'image au dessus du seuil imposé, nous vérifions son argument du gradient à travers la R-table.
- À la position de ce pixel, nous ajoutons les vecteurs contenus par le R-table.
- Le pixel atteint à partir de cette somme gagne un vote.

Le code qui met en oeuvre cette procédure est le suivant :

```

for i in range (0, np.shape(img7)[0]-1):
    for j in range (0, np.shape(img7)[1]-1):
        if(img6[i,j]>epsilon):
            #print(k)
            preTable[k,:]=[img7[i,j], i, j]
            k+=1

hough_vote= np.zeros((np.shape(img7)[0],np.shape(img7)[1]))
for i in range (0,np.shape(angle)[0]-1):
    for j in range (0, np.shape(preTable)[0]):
        if(abs(rTable[i][0]-preTable[j][0])<0.01):

```

```

for k in range (1, np.shape(rTable[i])[0]):
    if((preTable[j][1]+rTable[i][k][0])<np.shape(img7)[0]
       and (preTable[j][2]+rTable[i][k][1])<
       np.shape(img7)[1]):
        hough_vote[int(preTable[j][1]+rTable[i]
                        [k][0]),int(preTable[j][2]+rTable[i][k][1])]+=1
    hough_vote=hough_vote-
        np.min(hough_vote)/(np.max(hough_vote)-np.min(hough_vote))
cv2.imshow('hough_vote', hough_vote)

```

Finalement, les votes permettent le calcul de l'argument maximal, qui représente le "centre" de masse de l'objet, où nous allons placer notre rectangle de *tracking*. Une représentation de cette transformée de Hough est représentée par la figure ??

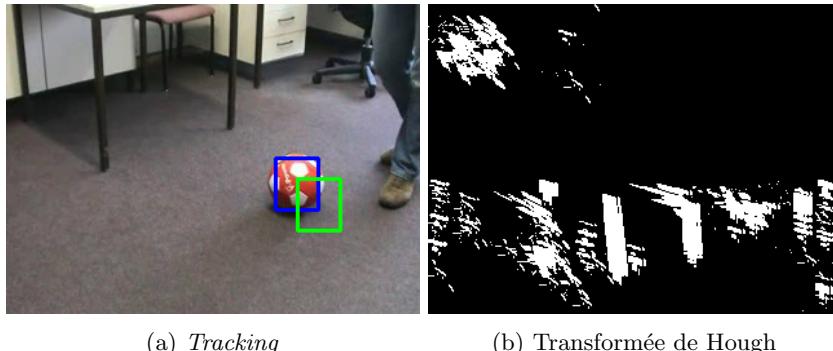


FIGURE 18 – Calcul du gradient des *frames*

Implémentant cette procédure, nous avons obtenu les résultats suivants. Notamment, l'utilisation du argument maximale fait que il y a une grande changement de position à chaque interaction.

2.3 Remplacer le calcul du maximum par l'application du Mean Shift sur la transformée de Hough. Interpréter le résultat et le comparer avec le précédent. Proposer une stratégie de mise à jour du modèle qui permette de prendre en compte les déformations de l'objet.

Finalement, nous avons appliqué l'algorithme *Mean Shift* aux transformées de *Hough* calculées. Le résultat devient plus stable, spécialement, si le seuil du gradient est bien choisi. Cela est cohérent avec ce que nous espérerions, car l'algorithme *Mean Shift* permet d'interpréter la transformée de *Hough* à travers des fonctions de probabilités de appartenance. Ainsi les changements de "centre de masse" d'un objet sont plus subtils et en accord avec le *tracking*.

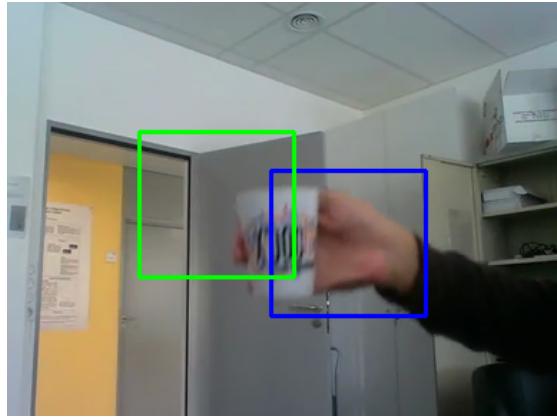


FIGURE 19 – Résultat argument maximale

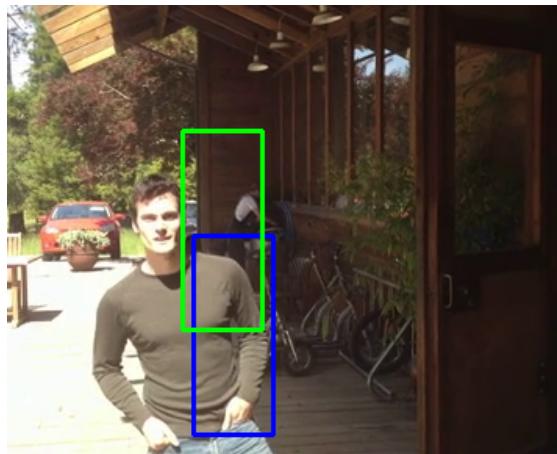


FIGURE 20 – Résultat argument maximale



FIGURE 21 – Résultat de l’application du *Mean Shift* sur la transformée de *Hough*

Dans les figures suivantes, nous avons quelques exemples de l’application de notre algorithme.

Cependant dans les vidéos comme VOT-Basket, les déformations du objet ne permettent pas une bonne *tracking*. Par conséquence, une possible amélioration de l’algorithme serait mettre à jour le modèle du objet.

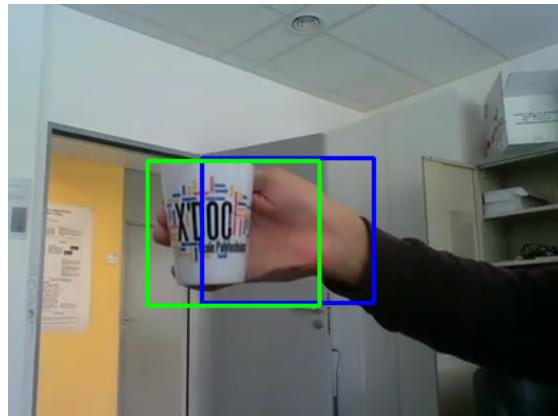


FIGURE 22 – Résultat de l'application du *Mean Shift* sur la transformée de *Hough*

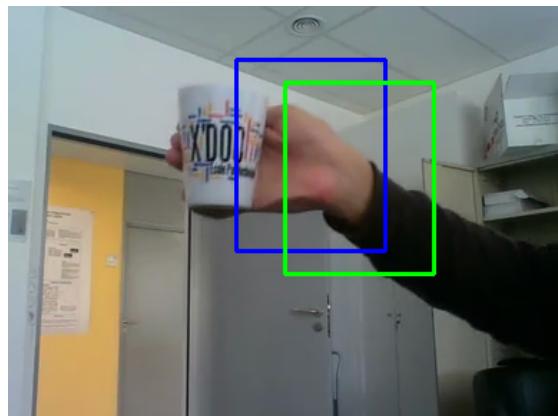


FIGURE 23 – Résultat de l'application du *Mean Shift* sur la transformée de *Hough*