# ROB313 - TD3
# Analyse vidéo et Tracking

BRAMBILLA Davide Luigi - GOMES DA SILVA Rafael

10 Janvier 2020

Les codes et résultats de ce TP peuvent être aussi accèdes sur le dépôt gitlab suivant:
**https://gitlab.data-ensta.fr/gomesdasilva/rob313/tree/master/TD3**.

## 1 Introduction

Le suivi d'objets est une tâche importante en vision par ordinateur et est appliqué dans plusieurs domaines (sécurité à domicile, véhicules autonomes, militaires, etc). Le suivi peut être défini comme le processus de détermination de l'emplacement des objets dans le temps dans une séquence d'images. Ce suivi se fait à partir de la reconnaissance d'un modèle délimité dans une région d'intérêt (**ROI**) à partir de la comparaison de ses caractéristiques structurelles avec celles des images d'une séquence.

Le suivi peut devenir complexe en raison des facteurs externes lors de la capture de la scène telle que l'intensité de la luminosité, le bruit, le comportement et la forme de l'objet à suivre.

Cependant, il existe plusieurs techniques déjà étudiées qui cherchent à résoudre le problème du suivi des objets, comme la méthode de ***Mean Shift*** et les méthodes utilisant la ***Transformée de Hough***.

Pour la méthode *Mean Shift*, une densité de probabilité d'une certaine variable est considérée. Partant d'une fenêtre initiale contenant un sous-ensemble de points de cette densité, on a que le point de départ est alors considéré comme le barycentre $C_O$ de la fenêtre . La fenêtre est déplacée pour être dirigée vers le centre $C_R$ de la distribution contenue dans la fenêtre, mettant fin à une itération de l'algorithme. Après cela, la définition de $C_O$ et la recherche de $C_R$ sont répétées $N$ fois jusqu'à que la distance entre $C_O$ et $C_R$ soit inférieure à un seuil $r$. La Figure 1.1 ci-dessous illustre le principe de fonctionnement de l'algorithme.

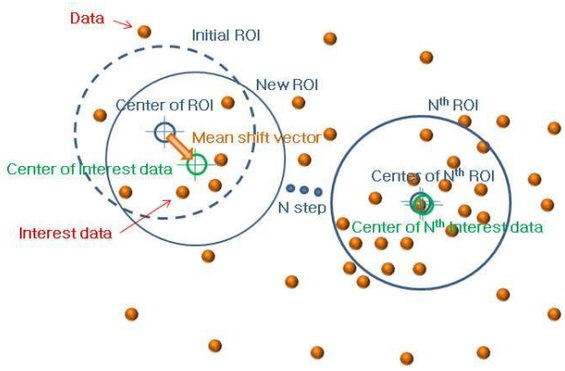
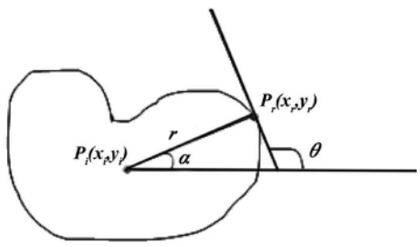



Figure 1.1: Schéma de l'algorithme *Mean Shift*. Source: [1]

La *Transformée de Hough (TH)* est une méthode de détection de formes facilement paramétrables (lignes, cercles, ellipses) dans les images. En général, la transformée est appliquée après que l'image a subi un pré traitement, généralement la détection des bords. Le concept principal de la *TH* est de définir une relation entre l'espace image et l'espace paramètres.
Il existe également une variation de la TH, appelée *Transformée de Hough Généralisée (THG)*. La *THG* est une technique de calcul efficace pour détecter ou localiser des formes arbitraires dans des images binaires, en utilisant une table de recherche de paramètres appelée *R-table*.

Le processus de création de la R-table se fait en définissant un point de référence $P_c = (x_c, y_c)$ (qui est généralement le barycentre du modèle) et les points de contour du modèle $P_i = (x_i, y_i)$. Pour chaque point $P_i$,un pair de paramètres $(r_i, \alpha_i)$ est déterminé et ils sont indexées par l'angle $\theta_i$ de direction de chaque point $P_i$, comme monstre la Figure 1.2.

(a)




(b)

| $\theta_i$ | $(r; \alpha)$ |
|---|---|
| $\Delta\theta$ | $(r_1, \alpha_1), (r_2, \alpha_2)$ |
| $2\Delta\theta$ | $(r_3, \alpha_3)$ |
| $3\Delta\theta$ | $(r_4, \alpha_4), (r_5, \alpha_5), (r_6, \alpha_6)$ |
| ... | ... |

Figure 1.2: Exemple de la R-table utilisée pour la méthode de *THG*. Source: [2]

Le but de ce TP est d'analyser la méthode *Mean Shift* et de vérifier ses utilisations et ses limites par rapport au suivi des objets, et également d'analyser la méthode de suivi des objets à l'aide de la *Transformée de Hough Généralisée*, en vérifiant également ses utilisations et ses limites.

# 2 Mean Shift

Comme mentionné précédemment, la méthode de la *Mean shift* vérifie le déplacement d'un objet en fonction d'une densité de probabilité. Dans notre cas, la densité de probabilité qui on utilisera sera fournie par la fonction *Back Projection* d'OpenCV. Cette fonction compare un histogramme modèle à chaque trame d'une séquence et donne la densité de probabilité qui indique dans quelle région de l'image analysée serait la plus susceptible de trouver le modèle

## 2.1 Question 1: Expérimentation de la suivi par le code *Tracking_MeanShift.py*

Cette première partie du TP a été consacrée à l'exploration de l'algorithme fourni. En regardant le fichier *Tracking_MeanShift.py*, il est possible de vérifier que la structure de la méthode proposé suit les étapes suivantes:

1. Sélectionner la séquence à analyser
2. Délimiter une région d'intérêt à partir de la première image de la séquence ($ROI$), afin de pouvoir déterminer le modèle à suivre
3. Convertir la séquence analysée de $RGB$ en $HSV$, sans tenir compte des intervalles d'image dans le domaine $HSV$ qui ne sont pas importants pour l'analyse
4. Générer un histogramme de la région d'intérêt
5. Pour toutes les images de la séquence:
   (a) Appliquer la fonction Back Projection à l'image actuelle, en prenant comme arguments l'image actuelle et l'histogramme du modèle
   (b) Effectuer la fonction de *Mean Shift* à la densité de probabilité obtenue par la fonction précédente

Pour analyser le fonctionnement de la méthode *Mean Shift* chaque séquence a été analysée selon l'objet qu'on veut suivre. La Figure 2.2 montre les objets que nous essayons de suivre pour ce TD.

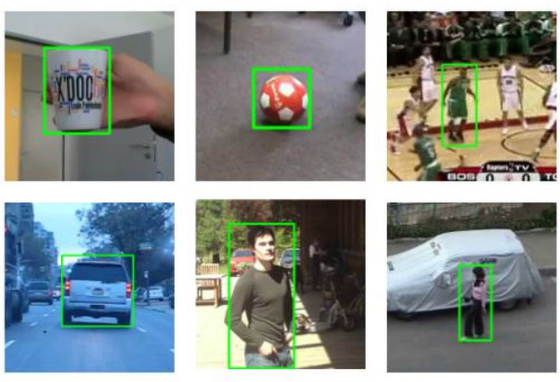


Figure 2.1: Objets à suivre

Après avoir effectué plusieurs tests avec les séquences fournies et avoir pris en compte le fonctionnement de l'algorithme *Mean Shift*, il a été possible de vérifier que plusieurs facteurs peuvent influencer la qualité de suivi du modèle. Ci-dessous, nous énumérerons quelques points importants compte tenu des avantages et des limites de cette méthode.

**Avantages**

- Le *Mean Shift* s'est révélé un tracker visuel rapide, ce que lui fait très adapté aux applications en temps réel.
- Il s'agit d'une méthode non paramétrique, qui est exécutée sans avoir besoin de connaître la forme du modèle (elle n'a besoin que de la densité de probabilité)

**Limitations**

- Si la taille de la fenêtre de suivi n'est pas mise à jour, le *Mean Shift* peut converger vers une région indésirable
- Si l'objet à suivre est partiellement masqué à un moment donné, le suivi obtenu peut présenter des résultats insatisfaisants.
- Le *Mean Shift* n'était pas si robuste au changement soudain de vitesse (caméra ou objet)
- Si la structure du modèle et la structure d'arrière-plan sont similaires, le suivi du modèle devient une tâche difficile.

## 2.2 Question 2: Analyser le résultat et proposer/ programmer des améliorations

Comme il a été possible de le vérifier, la méthode *Mean Shift* présente une série de limitations en fonction du type de vidéo analysé et du modèle choisi, principalement en raison du fait que la fenêtre pour le tracking converge presque toujours vers un maximum local lorsque la détection du modèle échoue dans l'une des scènes de la séquence. Cela veut dire que le lieu de la convergence est vers un **maximum local** de la distribution sous la fenêtre. Différentes tailles de fenêtre trouveront des maxima différents [3].

Cette erreur de convergence entraîne la stagnation de la position de la fenêtre, qui ne peut pas revenir à la détection du modèle.

Cependant, en analysant la densité calculée à partir de la fonction de *Back Projection*, il est à noter que certaines valeurs pourraient être exclues afin de garantir la représentation de la position de l'objet dans le cadre analysé. En conséquence, le risque de détérioration des performances du *Mean Shift* diminue. Les figures ci-dessous montrent la séquence de poids correspondant à rétro-projection de l'histogramme de teinte et montrent également la séquence de suivi du modèle.

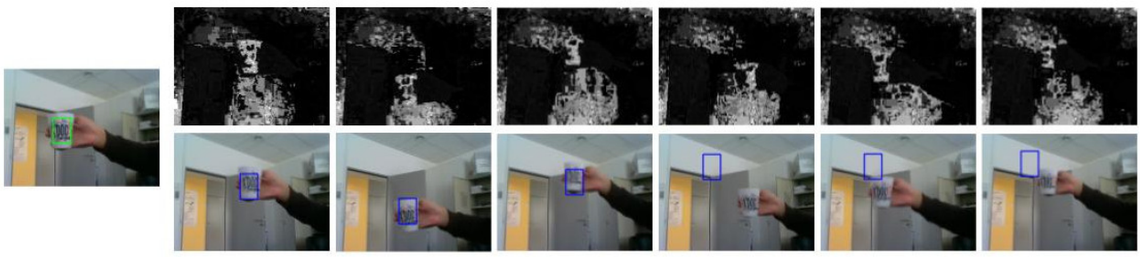


Figure 2.2: Suivi du modèle et densité de probabilité pour la séquence "*Antoine Mug.mp4*"

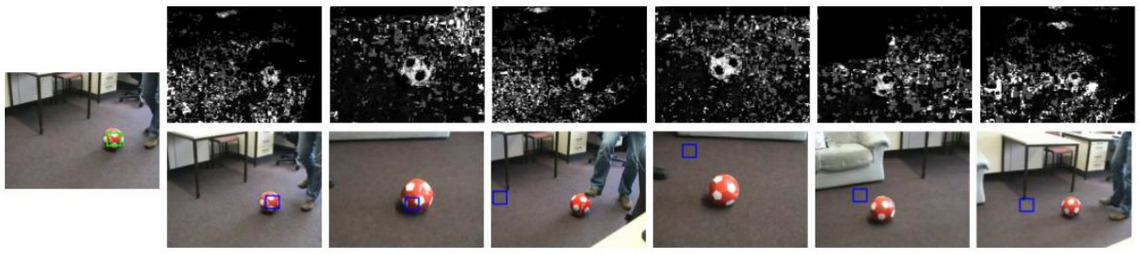


Figure 2.3: Suivi du modèle et densité de probabilité pour la séquence "*VOT-Ball.mp4*"

Figure 2.4: Suivi du modèle et densité de probabilité pour la séquence "*VOT-Basket.mp4*"

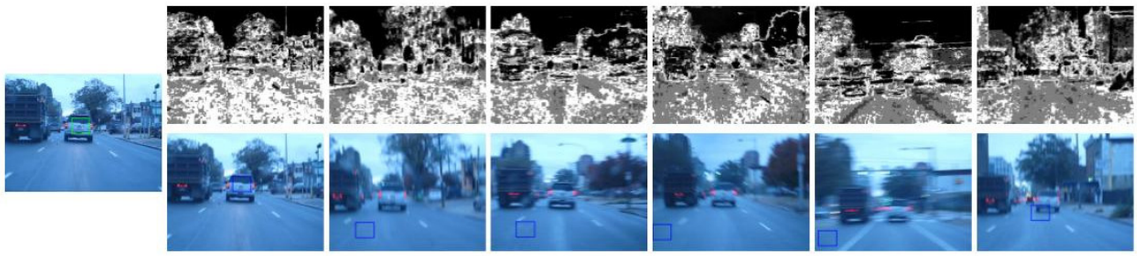


Figure 2.5: Suivi du modèle et densité de probabilité pour la séquence "*VOT-Car.mp4*"

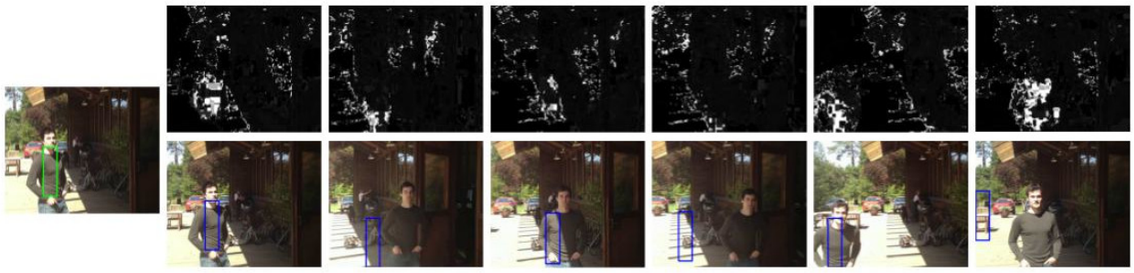


Figure 2.6: Suivi du modèle et densité de probabilité pour la séquence "*VOT-Sunshade.mp4*"

Figure 2.7: Suivi du modèle et densité de probabilité pour la séquence "*VOT-Woman.mp4*"

Par conséquent, nous avons pensé à appliquer *deux techniques* différentes pour résoudre le problème de l'amélioration de la méthode *Mean Shift*:

1. Éliminer certaines valeurs de la densité de probabilité de l'emplacement possible du modèle dans l'un des trames de chaque séquence
2. Mettre à jour l'histogramme du modèle .

### 2.2.1 Élimination de certaines valeurs de densité de probabilité

L'idée de cette étape a été pensée en vérifiant les résultats de l'image *dst*, résultat de la fonction *calcBackProject*. Par conséquent, pour chaque séquence fournie, des tests ont été effectués pour vérifier quel serait un seuil adéquat qui éliminerait les valeurs non significatives liées à d'autres objets dans chaque trame.

**Résultats**

Le tableau 2.1 ci-dessous montre les seuils les mieux adaptés à chaque séquence analysée. Comme on peut le voir, il existe des cas où l'utilisation d'un seuil n'était pas nécessaire, car le choix d'une fenêtre initiale adéquate peut garantir un bon suivi du modèle. Les valeurs indiquées par "**x**" montrent qu'il n'a pas été possible de trouver un seuil pour améliorer le suivi, et les valeurs indiquées par "**-**" montrent qu'il n'était pas nécessaire de trouver un seuil pour améliorer le suivi (la séquence d'origine montrait déjà un bon résultat pour le suivi).

| Sequence | Seuil |
|---|---|
| Antoine_Mug.mp4 | <230 |
| VOT-Ball.mp4 | <180 |
| VOT-Basket.mp4 | <180 |
| VOT-Car.mp4 | x |
| VOT-Sunshade.mp4 | <230 |
| VOT-Woman.mp4 | <60 |

Tableau 2.1: Valeur des seuils pour chaque séquence fournie, pour améliorer le *Mean Shift*

Les images ci-dessous montrent les résultats obtenus pour cette méthode.

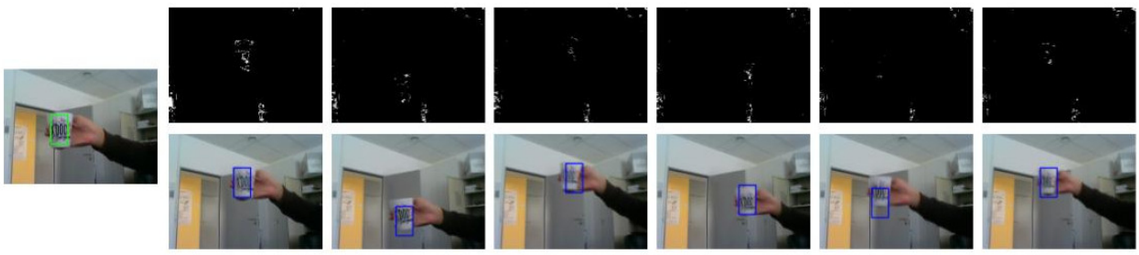


Figure 2.8: Résultats de la première proposition d'amélioration avec la séquence "*Antoine Mug.mp4*", avec un seuil de 230

Figure 2.9: Résultats de la première proposition d'amélioration avec la séquence "*VOT-Ball.mp4*", avec un seuil de 180

Figure 2.10: Résultats de la première proposition d'amélioration avec la séquence "*VOT-Basket.mp4*", avec un seuil de 180

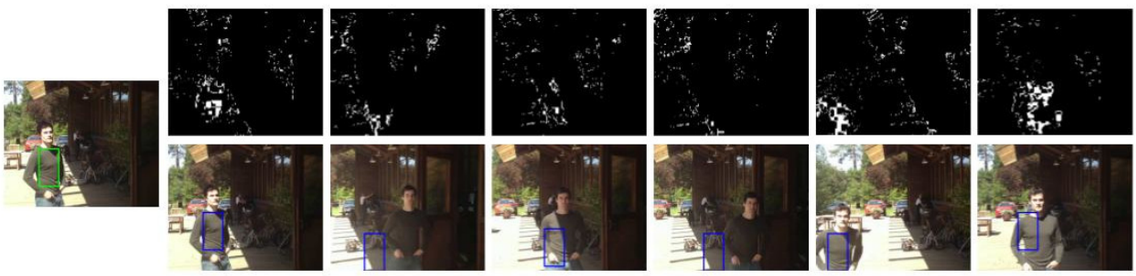


Figure 2.11: Résultats de la première proposition d'amélioration avec la séquence "*VOT-Sunshade.mp4*", avec un seuil de 230

Figure 2.12: Résultats de la première proposition d'amélioration avec la séquence "*VOT-Woman.mp4*", avec un seuil de 60

### 2.2.2 Mise à jour de l'histogramme

L'idée de cette étape était de mettre à jour l'histogramme du modèle pour chaque image de la séquence. Malheureusement, les résultats obtenus n'étaient pas satisfaisants car nous n'avons pas pu trouver un moyen efficace de recalculer l'histogramme.
L'idéal serait de réaliser cette mise à jour considérant que l'objet peut subir des déformations dans le temps, cela veut dire, redimensionner la fenêtre du tracking selon les nouvelles caractéristiques de l'objet déformé. Cependant, le mis à jour de l'histogramme d'objet est une tache compliqué vu que pour cela il faudrait redimensioner la fenêtre de suivi de façon précis pour ne pas avoir des informations que ne appartiennent pas à l'objet désiré. L'implémentation réalisée, donc, n'était pas parfaite parce que on n'a pas redimensionné la fenêtre de suivi avant de mettre a jour l'histogramme du modèle.

**Résultats** Les images ci-dessous montrent deux des résultats obtenus pour cette méthode. Pour les autres cas, les résultats avaient les mêmes caractéristiques: la mise à jour de l'histogramme finit par amener la fenêtre de suivi dans une région dans laquelle le modèle n'est pas présent, et le nouveau modèle d'histogramme est créé dans cette mauvaise région. Cela implique une erreur systématique dans la mise à jour de l'histogramme et entraîne une inefficacité de la méthode.

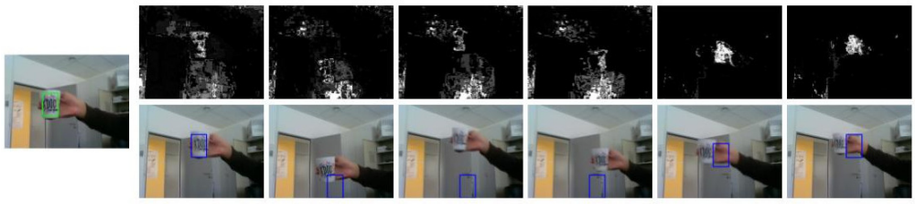


Figure 2.13: Résultats de la deuxième proposition d'amélioration avec la séquence "*Antoine Mug.mp4*"

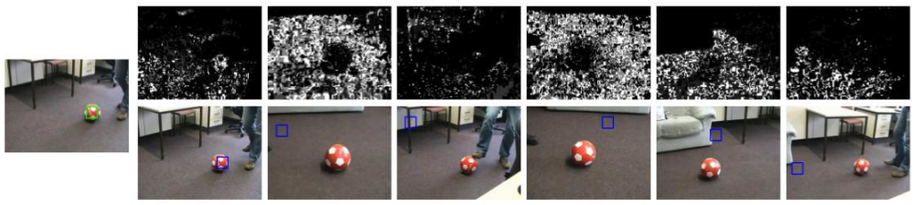


Figure 2.14: Résultats de la deuxième proposition d'amélioration avec la séquence "*VOT-Ball.mp4*"

### 2.2.3 Conclusion

Parmi les deux méthodes proposées ci-dessus, seule la première a présenté des résultats satisfaisants. En effet, la mise à jour de l'histogramme n'a pas été effectuée compte tenu de la déformation de l'objet, elle n'a été effectuée qu'en tenant compte des nouveaux éléments présents dans le *ROI* initial. La déformation de l'objet était un point difficile à mettre en œuvre car elle nécessiterait plus d'informations sur le modèle, et pas seulement son histogramme.

Pour la première méthode proposée, bien que les résultats obtenus pour la plupart des vidéos ont été améliorés, nous n'avons pas pu avoir des bons résultats pour les vidéos *"VOT-Car.mp4"* ni *"VOT-Sunshade.mp4"*.
Pour la vidéo *"VOT-Sunshade.mp4"*, le fait qu'il y a un échange d'éclairage pourrait avoir influencé les résultats, car en observant la densité de probabilité pour cette séquence, la présence de deux régions est notée: une avec une forte lumière et une avec de l'ombre.

Pour la vidéo *"VOT-Car.mp4"*, les histogrammes du modèle analysé étaient toujours très similaire à l'histogramme des trames du plan en arrière pour chaque séquence, ce qui faisait que la *Back Projection* renvoyait toujours une densité avec des valeurs très proches et homogènes, donc difficiles à dissocier en utilisant seulement un seuillage.

### 2.2.4 Pistes d'amélioration

Pour une étude plus approfondie et plus détaillée de l'utilisation du *Mean Shift* pour la suivi des objets, il serait important de prendre en compte les différents changements structurels dans chaque séquence (comme la vitesse de la caméra, la vitesse de l'objet, la similitude de texture, etc.) ainsi que le modèle à suivre (compte tenu de sa déformation et de ses nouvelles fonctionnalités au fil du temps). Dans la littérature, il est possible de trouver des méthodes et des fonctions proposées pour améliorer le *Mean Shift*, comme *CamShift* [4], et une combinaison d'un filtre de Kalman avec le *Mean Shift* [5] [6].

# 3 Transformée de Hough

Le calcul de l'orientation locale pour chaque trame de l'image est une étape essentielle pour effectuer le calcul de la **Transformée de Hough (TH)**, car comme vu précédemment, cette transformation fonctionne en détectant la forme des objets. De plus, étant donné que les objets à détecter dans notre cas sont des objets qui ont des formes non simples, nous utiliserons le processus de la *Transformée de Hough généralisé* pour trouver le modèle à l'aide d'une *R-table*. Après cela, nous allons vérifier deux façons différentes de faire la suivi d'objets:

1. En utilisant la valeur maximale de l'accumulateur de la transformation de Hough
2. En utilisant le Mean shift sur la distribution obtenue avec l'accumulateur obtenu à partir de la transformée de Hough.

### 3.0.1 Calcul de l'orientation locale

Pour calculer l'orientation locale de chaque image, la fonction **S*obel(src_gray, ddepth, x_ordre, y_ordre)*** du OpenCV a été utilisée, où:

- **src_gray** est l'image d'entrée en niveaux de gris
- **ddepth** est la profondeur de l'image de sortie
- **x_ordre** est l'ordre de la dérivée dans la direction x
- **y_ordre** est l'ordre de la dérivée dans la direction y

Cette fonction calcule la dérivée de l'image soit dans la direction x (valeurs des deux derniers paramètres: 1, 0) soit dans la direction y (valeurs des deux derniers paramètres: 1, 0 ). Après cela, la norme de gradient est calculée comme étant $\|\nabla I\| = \sqrt{I_x^2 + I_y^2}$.

Code 1: *Calcul de l'orientation locale*

```
# ORIENTATION LOCALE
img = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
grad_x = cv2.Sobel(img,cv2.CV_32F,1,0) # x
grad_y = cv2.Sobel(img,cv2.CV_32F,0,1) # y
gradient = np.hypot(grad_x, grad_y)/256
img2 = cv2.cvtColor(gradient, cv2.COLOR_GRAY2BGR)
img2[np.where(((img2[:,:,:]<0.75)).all(axis=2))] = [0,0,255]
```

Les seuils ont été définis selon le Tableau 3.1 ci-dessous, car chaque séquence donnée a généré un résultat différent par rapport aux pixels significatifs pour notre étude, et il a ensuite fallu adapter le seuil à chaque situation.

| Sequence | Seuil |
|---|---|
| Antoine_Mug.mp4 | <0.35 |
| VOT-Ball.mp4 | <0.35 |
| VOT-Basket.mp4 | <0.7 |
| VOT-Car.mp4 | <0.5 |
| VOT-Sunshade.mp4 | <0.8 |
| VOT-Woman.mp4 | <0.75 |

Tableau 3.1: Valeur des seuils pour chaque séquence fournie, pour déterminer l'orientation locale de chaque trame

Les images présentées de Figure 3.1 en Figure 3.6 montrent les résultats obtenus pour cette étape. Pour chaque image, les *pixels significatifs* sont affichés en *blanc*, tandis que les *pixels non significatifs* sont affichés en *rouge*. Comme vous pouvez le voir, les images de chaque cadre montrent maintenant les contours des objets les plus évidents dans chaque cadre, et cela sera utile à la fois pour déterminer le modèle pour composer la table $R$ et pour trouver l'objet à suivre.


Figure 3.1: Orientation locale pour l'une des images de la séquence "*Antoine_Mug.mp4*" à partir des pixels les plus significatifs.

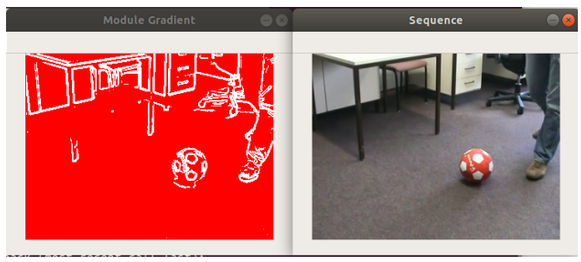



Figure 3.2: Orientation locale pour l'une des images de la séquence "*VOT-Ball.mp4*" à partir des pixels les plus significatifs.

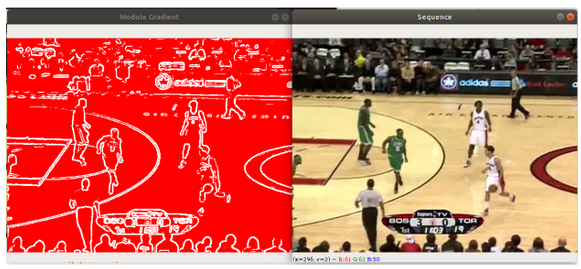



Figure 3.3: Orientation locale pour l'une des images de la séquence "*VOT-Basket.mp4*" à partir des pixels les plus significatifs.

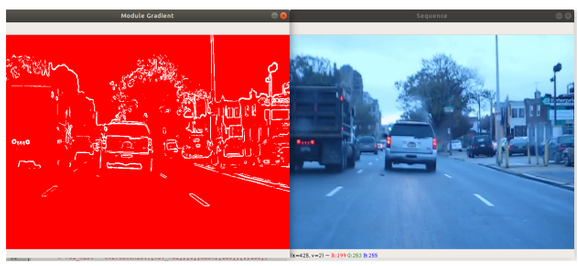



Figure 3.4: Orientation locale pour l'une des images de la séquence "*VOT-Car.mp4*" à partir des pixels les plus significatifs.

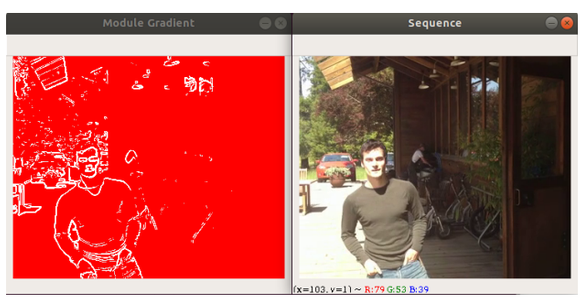



Figure 3.5: Orientation locale pour l'une des images de la séquence "*VOT-Sunshade.mp4*" à partir des pixels les plus significatifs.

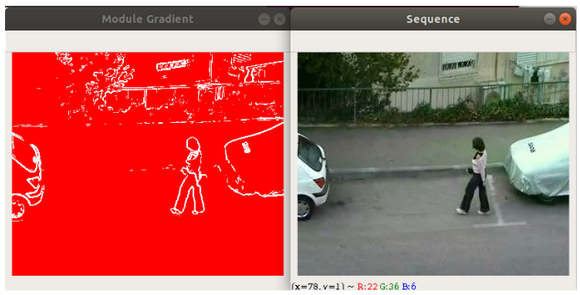



Figure 3.6: Orientation locale pour l'une des images de la séquence "*VOT-Woman.mp4*" à partir des pixels les plus significatifs.

### 3.0.2 Transformée de Hough généralisée

Dans cette partie, les étapes suivies et les résultats obtenus pour suivre un modèle à l'aide de *THG* seront présentés.

Pour commencer, après avoir obtenu un modèle souhaité, la *R-table* est créée pour représenter le modèle sous une forme implicite indexée par rapport à son orientation.

Code 2: *Creation de la R-table*

```
# CREATES THE R-TABLE
# Creates the list for the R-table
table = [[0 for j in range(1)] for i in range(90)]

# Computes the center of the image (reference)
center_img = [int(shape(roi_hough)[0]/2), int(shape(roi_hough)[1]/2)]

# Computes the paramters of the model
for x1 in range(shape(roi_hough)[0]):
   for y1 in range(shape(roi_hough)[1]):
      if (roi_hough[x1, y1] != 0): # If the point is in the border of the model
         # Point of referencce
         x2 = center_img[0]
         y2 = center_img[1]

         # finds the paramters for each angle
         r = [(x2-x1), (y2-y1)]
         if (x2-x1 != 0):
            # Finds the value of theta
            theta = int(np.rad2deg(np.arctan(int((y2-y1)/(x2-x1)))))
         else:
            theta = 0
            r = 0
         if (r != 0):
            # Adds the value of r to the R-table at the position of the angle
            table[np.absolute(theta)].append(r)
for i in range(len(table)):
   # After completing the table, eliminates the null values
   table[i].pop(0)
```

Après cela, le calcule de la transformée de Hough associée à toutes les images de la séquence se fait en consultant la *R-table*, et un accumulateur est créé pour vérifier quels paramètres sont répétés le plus souvent.

Code 3: *Relie l'image d'un trame au modèle en se référant à la table R*

```
    # Comparing with the R-table
    accumulator = np.zeros((shape(gradient)[0]+int(shape(gradient)[0]*0.2), shape(gradient)[1]
        +int(shape(gradient)[0]*0.2)))

    for x_comp in range(1, shape(gradient)[0]):
       for y_comp in range(shape(gradient)[1]):
          if gradient[x_comp, y_comp] != 0:
             if (x_comp != 0):
                theta = int(np.rad2deg(np.arctan(int(y_comp/x_comp))))
             else:
                theta = 0
             orientations = table[theta]
             for count_ori in orientations:
                accumulator[count_ori[0]+x_comp, count_ori[1]+y_comp] += 1
```

Enfin, la valeur maximale de l'accumulateur créé à partir de THG est trouvée et la fenêtre de suivi est centrée à ce point.

Code 4: Trouve la valeur maximale de l'accumulateur

```
# Finds the maximum value of the Hough Transform
x_acc, y_acc = np.unravel_index(np.argmax(accumulator), shape(accumulator))
frame_tracked = cv2.rectangle(frame, (y_acc-int(h/2),x_acc-int(w/2)), (y_acc + int(h/2),x_acc
    + int(w/2)), (255,0,0) ,2)
```

Les figures ci-dessous montrent les résultats obtenus pour le suivi à l'aide de la valeur maximale de l'accumulateur à l'aide du *THG*.

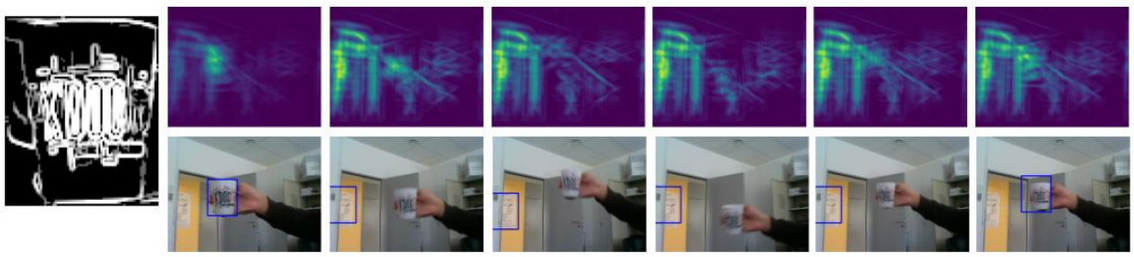


Figure 3.7: Résultats pour le suivi à l'aide de THG pour la séquence "*Antoine_Mug.mp4*"

Figure 3.8: Résultats pour le suivi à l'aide de THG pour la séquence "*VOT-Car.mp4*"

On a pu voir que le suivi est plus stable que celui utilisant l'histogramme du modèle. Cependant, comme il s'agit d'une analyse dense, cette méthode est coûteuse en termes de temps de calcul par rapport à la méthode de *Mean Shift*.

Certains détails importants à noter sont que, comme le choix de l'objet est fait initialement, il y a des moments où les détails capturés initialement ne sont pas visibles dans l'un des trames. Cela provoque parfois une erreur de suivi. Cependant, contrairement à la méthode précédente, étant donné que l'image contient une représentation de modèle similaire à celle initialement choisie, la fenêtre de suivi revient à nouveau vers l'objet - dans le cas du *Mean Shift*, revenir à l'objet après avoir perdu la région de la densité de probabilité appropriée était dificile, car cela cause une stagnation de la position de la fenêtre du tracker.

D'un autre côté, cette méthode s'est revelée efficace pour détecter des objets qui ont des caractéristiques d'orientation locale constante entre toutes les images, comme c'est le cas de la séquence automobile. En utilisant le *THG*, il a été possible de détecter la voiture dans la plupart des trames, sauf dans le cas où il y avait un mouvement très soudain de la caméra, qui a entraîné la déformation des détails de la voiture.

### 3.0.3 Mean Shift et la transformée de Hough

Enfin, la méthode du *Mean Shift* a été combinée avec la *THG*, comme montre le code ci-dessous.

Code 5: *Mean Shift* sur l'accumulateur

```
# Mean Shift on the the Hough Transform
  ret, track_window = cv2.meanShift(accumulator, track_window, term_crit)
```

Il a été possible d'observer que cette combinaison générait des résultats plus stables par rapport à l'utilisation du shift moyen avec la densité de probabilité générée par l'algorithme *Back Projection*. Cependant, il reste le problème d'adapter la taille de la fenêtre de suivi par rapport au changement des valeurs présentes dans l'accumulateur.

D'une part, cette méthode élimine le problème du déplacement brut de la fenêtre de suivi vers la valeur maximale (comme dans le cas précédent). En revanche, si la valeur maximale indique un faux positif par rapport à la position du modèle. Si la fenêtre de suivi n'est pas suffisamment grande pour revenir à l'objet lorsque la valeur maximale indique la position réelle du modèle, il se peut que notre suivi soit altéré et ne puisse pas revenir sur l'objet.

**Pistes d'amélioration:** Un moyen possible d'améliorer la suivi d'objets est de combiner la méthode (1) qui utilise la valeur de maxima obtenue à partir de la *Transformée de Hough Generalisée* (THG maximum) et la méthode (2) qui utilise le *Mean Shift* de la distribution de l'accumulateur obtenue à partir de la

*Transformée de Hough* (Mean Shift THG).

Ci dessous nous allons proposer le comportement de cette nouvelle méthode.

On commence le processus de suivi avec (1), car les scènes initiales dans toutes les séquences étudiées ont tendance à présenter un bon résultat de suivi. Après cela, pour l'analyse des *frames* suivants, la position du point de maxima de la *THG* de la trame précédente est mémorisée et comparée: si la distance entre les maxima est supérieure à un seuil, cela peut signifier qu'il y a eu une perte de suivi de l'objet.

Dans le cas où (1) ne répond pas au critère de seuil, on passe à la méthode (2) afin d'éviter que la fenêtre de suivi ne soit pas trop éloignée de l'objet. On continue a utiliser (2) jusqu'à que $abs(P_{m_i} - P_{m_{i-1}})$ soit inférieur au seuil $Th$, où:

- $P_{m_i}$: Point de maxima obtenu avec THG pour la trame $i$
- $P_{m_{i-1}}$: Point de maxima obtenu avec THG pour la trame $i-1$
- $Th$: Seuil choisi

La Figure 3.9 illustre l'idée proposé.

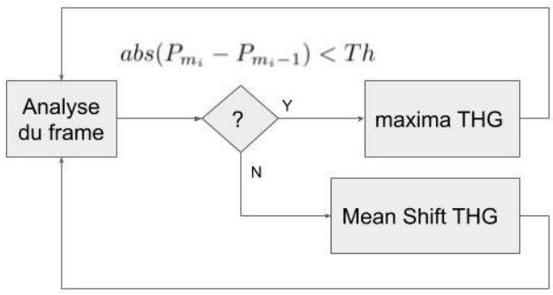



Figure 3.9: Proposition d'amélioration de la méthode de suivi

La méthode liée à la *THG* serait préférable car elle résiste mieux aux faux positifs par rapport au positionnement de la fenêtre de suivi.

Pour mettre à jour le modèle, il faudrait alors vérifier la qualité du positionnement de la fenêtre de suivi comparé avec les informations des trames précédentes, et également vérifier la corrélation *corr* entre le modèle et l'objet détecté à l'intérieur de la fenêtre, ayant $corr \in [0, 1]$, avec 0 étant des objets complètement différents, et 1 étant des objets égaux.

L'adaptation du modèle a pour objectif la pris en compte des évolutions dans le temps, afin de continuer à détecter le même objet même après des possibles déformations. Le modèle est mis à jour en fonction du contenu détecté dans la fenêtre. Par conséquent, avant d'effectuer des modifications, nous devons nous assurer que:

1. La fenêtre est bien située et l'objet souhaité est dedans cette fenêtre. Cela peut être fait sur la base des critères établis par la méthode proposée, qui prend en compte les informations de la trame actuelle et également de l'ancienne trame.
2. L'objet dans la fenêtre de la trame est le même objet que celui de la fenêtre du modèle. Cela peut être fait en vérifiant la corrélation entre le contenu de la fenêtre du modèle actuelle et la fenêtre de trame en cours de vérification

# References


[1] Bong-Cheol Seo, Sung-Soo Kim, and Dong-Youm Lee. "Target-tracking system for mobile surveillance robot using CAMShift image processing technique". In: *Transactions of the Korean Society of Mechanical Engineers A* 38.2 (2014), pp. 129–136.

[2] A Ouled Zaid et al. "Improved localization of coronary stents based on image enhancement". In: *International journal of biomedical science: IJBS* 4.3 (2008), p. 212.

[3] Adrian Kaehler and Gary Bradski. *Learning OpenCV 3: computer vision in C++ with the OpenCV library.* " O'Reilly Media, Inc.", 2016.

[4] Zhaowen Wang et al. "CamShift guided particle filter for visual tracking". In: *Pattern Recognition Letters* 30.4 (2009), pp. 407–413.

[5] Ravi Kumar Jatoth, Sampad Shubhra, and Ali Ejaz. "Performance comparison of Kalman filter and mean shift algorithm for object tracking". In: *International Journal of Information Engineering and Electronic Business* 5.5 (2013), p. 17.

[6] Liu Jiyan, Pan Jianshou, and WU Yapeng. "Mean-shift tracking algorithm combined with kalman filter". In: *Computer Engineering and Application.* 45.12 (2009), pp. 184–186.