



IP PARIS

ROB313

VISION POUR LA ROBOTIQUE

TP3 : Analyse vidéo et Tracking

Réalisé par :

Bastien HUBERT
Mariam MAAZOUN
Maxime FANTINO

Année scolaire

2022/2023

I INTRODUCTION

Le suivi d'objets dans les séquences vidéo peut être un défi technique et scientifique, car il implique de suivre un objet en mouvement dans un environnement en constante évolution.

L'objectif de ce TP est de comprendre les enjeux et difficultés du tracking, et d'expérimenter et programmer les solutions fondées sur les algorithmes de Mean Shift et de Transformées de Hough Généralisées.

II MEAN SHIFT

II.1 Question 1 :

L'algorithme **Mean Shift** est une technique de suivi d'objet qui consiste à déplacer itérativement une fenêtre sur une image en direction du pic de densité de la distribution des pixels de l'image. Le centre de la fenêtre est recalculé à chaque itération en prenant la moyenne pondérée des coordonnées des pixels situés à l'intérieur de la fenêtre, en fonction de leur valeur de densité. L'algorithme s'arrête lorsque le centre de la fenêtre ne se déplace plus ou que le nombre d'itérations maximales est atteint.

Dans le code fourni, l'algorithme Mean Shift est utilisé pour suivre un objet défini par l'utilisateur dans la première image d'une séquence vidéo. La densité marginale f_H sur la composante H de teinte est utilisée pour calculer la distribution des pixels de l'objet, qui est ensuite utilisée pour le suivi. L'algorithme calcule une image de retour (backprojected image) basée sur la distribution de l'objet, puis applique Mean Shift sur cette image pour trouver la nouvelle position de l'objet dans chaque image de la séquence vidéo.

La structure du code *Tracking MeanShift.py* suit les étapes principales ci-dessous :

1. Définition de la fenêtre de l'image pour la détection de la ROI en utilisant "cv2.namedWindow" et "cv2.setMouseCallback".
2. Boucle "while True" pour afficher l'image et attendre que l'utilisateur définisse la région d'intérêt en cliquant sur l'image. Lorsque la ROI est définie, un rectangle vert est dessiné autour de la région d'intérêt.
3. Lorsque la touche "q" est pressée, la boucle est interrompue et la variable "*track_window*" est définie à partir des coordonnées de la ROI.
4. Définition de la région d'intérêt pour le suivi à partir des coordonnées de la ROI.
5. Conversion de la région d'intérêt de l'espace couleur RGB à l'espace couleur HSV.
6. Calcul puis normalisation de l'histogramme de la région d'intérêt en utilisant la composante de teinte (H) de l'espace couleur HSV.
7. Pour chaque image dans la séquence :
 - Convertir l'image en image HSV
 - Calculer la densité de probabilité marginale f_H de la composante H de l'image à l'aide de *cv2.calcBackProject()* en utilisant l'histogramme créé précédemment.
 - Appliquer l'algorithme Mean Shift à f_H en utilisant *cv2.meanShift()* pour mettre à jour la position de la ROI.

Pour analyser le fonctionnement de cet algorithme, on va le tester sur plusieurs séquences après avoir identifier à chaque fois les objets à suivre .

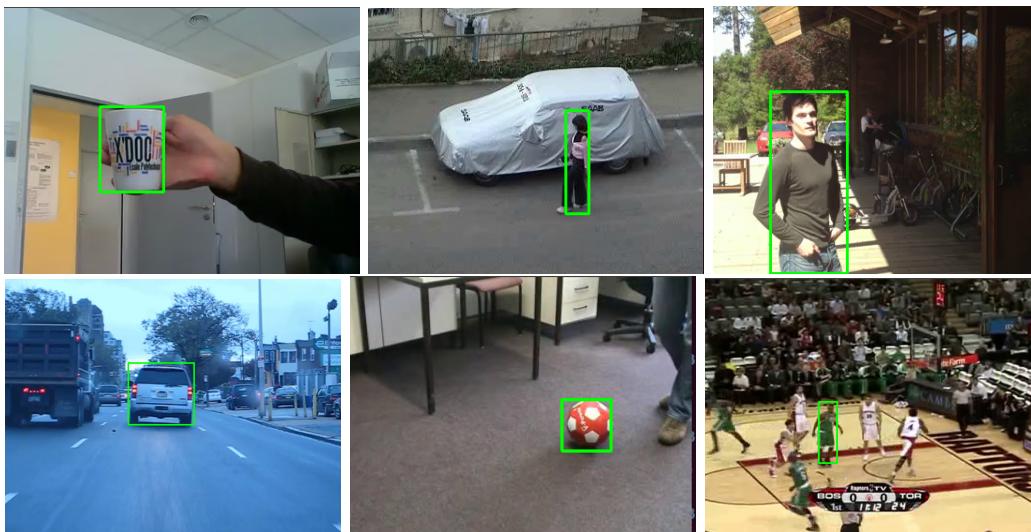


FIGURE 1 – Les objets à suivre

Avantages :

Les avantages de l'algorithme Mean Shift sont sa simplicité et sa capacité à suivre des objets dont la forme et la couleur varient dans le temps.

Il est également relativement rapide par rapport à d'autres techniques de suivi d'objet.

Limitations :

Cependant, l'algorithme Mean Shift a également certaines limitations.

- Il peut avoir des difficultés à suivre des objets qui se déplacent rapidement ou qui subissent des changements de forme importants : L'utilisation de l'algorithme du Mean Shift est influencée par la vitesse du target. Un objet qui se déplace rapidement et de manière imprévisible par rapport à la vitesse de la caméra peut être difficile à détecter avec l'algorithme du Mean Shift. Ceci est illustré par l'expérience de la vidéo Antoine_Mug, dont les séquences sont présentées dans la figure 2. Au début de la vidéo, lorsque la vitesse du mug est faible, l'algorithme parvient à détecter parfaitement le target dans chaque frame . Cependant, dès que la vitesse du mug augmente, l'algorithme perd sa trace (séquences 2 et 3). Finalement, lorsque la vitesse du target se stabilise et redévient faible et presque constante, le Mean Shift permet à nouveau de localiser le target correctement pour chaque frame (séquence 4).



FIGURE 2 – Un exemple de suivi pour la vidéo Antoine_Mug.mp4

- On remarque que l'algorithme est sensible aux perturbations dans l'environnement, comme l'éclairage ou les changements de fond, qui peuvent modifier la distribution des pixels de l'objet et affecter le suivi.

De même, lorsque l'arrière-plan de la séquence est complexe et contient plusieurs objets similaires, il devient difficile d'obtenir des performances de suivi optimales. Pour illustrer les variations de lumière et d'ombre, comme montré dans la figure 3, on remarque que lorsqu'un sujet se déplace entre des zones ensoleillées et ombragées, l'algorithme peut ne pas être en mesure de suivre son visage correctement.

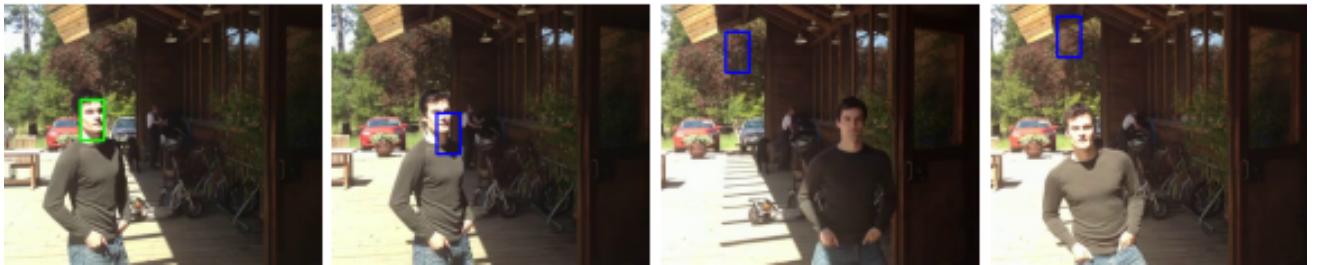


FIGURE 3 – Un exemple de suivi pour la vidéo VOT-Sunshade

- L'algorithme du Mean Shift présente une limitation importante en termes de robustesse, en raison de la propriété de la zone d'intérêt qui est introduite manuellement avec des dimensions et une forme fixes qui restent inchangées tout au long de la boucle principale. Cette limitation se manifeste particulièrement lors de changements d'échelle, ce qui réduit l'efficacité de l'algorithme et peut compromettre ses résultats.



FIGURE 4 – Un exemple de suivi pour la vidéo VOT-Woman

II.2 Question 2 :

Pour examiner plus minutieusement le résultat, on va s'intéresser à la rétro-projection qui peut être exploitée pour segmenter l'image, ce qui nous permet de localiser la région d'intérêt. Chaque pixel de l'image de sortie reflète la probabilité que le point correspondant appartienne à la région d'intérêt (RoI). En effet, plus la valeur du pixel est élevée, plus il est probable que le point appartienne à l'objet. Pour afficher la séquence des composantes couleur et des poids de la rétro-projection, on peut ajouter le code suivant à la boucle principale du script :

```
#Q2-----
b, g, r = cv2.split(frame)
# Display the three color channels and the backprojection weight
cv2.imshow('Blue channel', b)
cv2.imshow('Green channel', g)
cv2.imshow('Red channel', r)
cv2.imshow('BackProject', dst)
#-----
```

FIGURE 5 – Affichage de la séquence des composantes couleur

"*dst*" , calculée à partir de la fonction *cv2.calcBackProject* est l' image de retour de projection qui contient des valeurs de poids qui représentent la probabilité que chaque pixel de l'image actuelle appartienne à l'objet tracké.

Cela affichera les quatre images côte à côté dans une même fenêtre comme le montre la figure ci-dessous.



FIGURE 6 – Rétro-projection $R_H(x, y) = f_H(H(x, y))$

On remarque que l'algorithme ne parvient pas toujours à suivre l'objet. Cela est principalement dû au fait que la fenêtre de suivi converge presque toujours vers un maximum local lorsqu'il y a échec de la détection du modèle dans l'une des scènes de la séquence. Cette convergence vers un maximum local signifie que la fenêtre de suivi ne parvient pas à retrouver la position du modèle.

Pour améliorer le résultat, on va tester deux types d'améliorations :

Amélioration avec la mise à jour de l'histogramme

La mise à jour de l'histogramme se fait pour s'adapter aux changements de l'apparence de l'objet suivi au fil du temps. En effet, la couleur, l'éclairage et le contexte peuvent varier et l'histogramme initial devient alors obsolète. En recalculant l'histogramme à partir de la région d'intérêt (ROI) à chaque itération, on peut suivre l'objet même s'il se déplace ou si son apparence change.

Plus précisément, on va coder la mise à jour de l'histogramme à partir de la région d'intérêt (ROI) de l'image en utilisant la fonction *cv2.calcHist()* et *cv2.normalize()* pour normaliser les valeurs de

l'histogramme entre 0 et 255. Le nouvel histogramme est utilisé pour mettre à jour le modèle de l'objet suivi dans la boucle de suivi.

En testant notre code, on trouve le résultat affiché sur la figure 7.

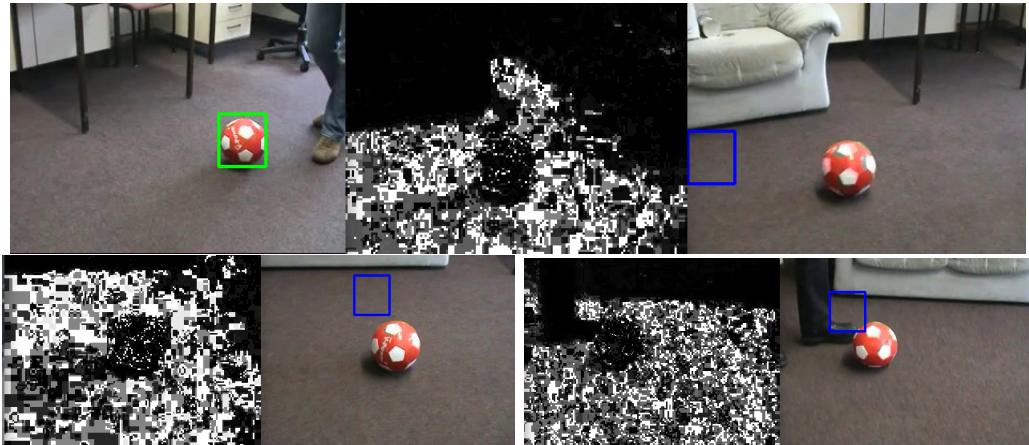


FIGURE 7 – Résultat

Même avec une mise à jour de l'histogramme pour chaque image de la séquence, le résultat n'est pas très satisfaisant. D'ailleurs, une fois qu'il y a une mauvaise détection, l'algorithme ne peut plus se corriger.

Autre amélioration : Filtrage sur l'image de retour de projection pour supprimer les pixels dont le poids est inférieur à un certain seuil

Après la rétropénétration de l'histogramme sur une image pour obtenir une carte de poids, il peut y avoir des pixels avec des valeurs de poids très faibles qui ne contribuent pas significativement à la localisation de l'objet suivi comme on l'avait remarqué précédent.

Ainsi, on va définir un seuil pour la valeur de poids minimale que chaque pixel doit avoir pour être considéré comme faisant partie de l'objet suivi. Les pixels avec une valeur de poids inférieure à ce seuil sont ignorés et remplacés par zéro dans la carte de poids, ce qui signifie qu'ils ne contribuent pas à la localisation de l'objet.

On ajoute alors la ligne suivante $dst[dst < seuil] = 0$ dans le code pour appliquer un filtrage sur la carte de poids dst. Elle modifie tous les éléments de dst dont la valeur est inférieure à seuil en les remplaçant par zéro.

On teste cette amélioration sur la séquence de vidéo VOT-Ball.mp4 . On trouve le résultat affiché sur la figure 8.

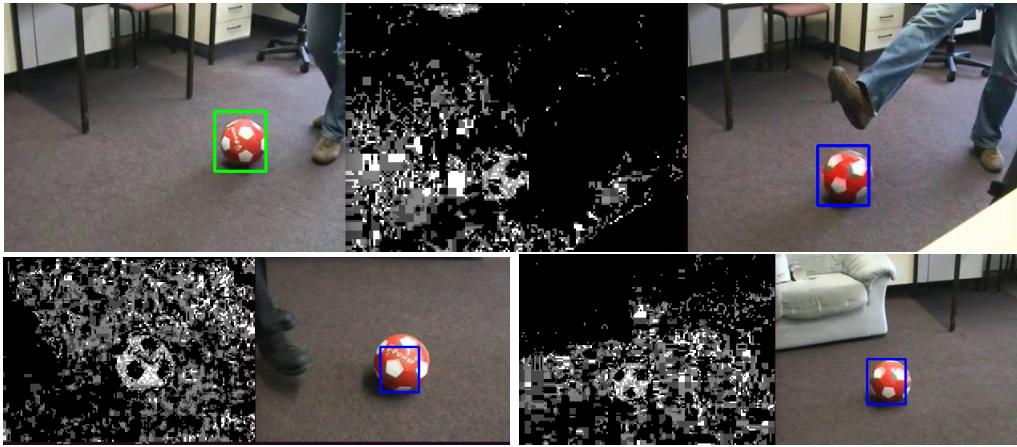


FIGURE 8 – Résultat pour un seuil égal à 60.

==== > On remarque qu'on a eu des résultats beaucoup plus intéressants.

Ce filtrage est utile car il élimine le bruit dans la carte de poids, qui peut provenir de l'éclairage, du mouvement de l'appareil photo ou d'autres sources, et qui pourrait interférer avec la localisation précise de l'objet suivi. En supprimant les pixels avec des poids très faibles, le filtre permet également de simplifier l'image de retour de projection pour faciliter l'identification de la position de l'objet suivi. Il est important de noter que la taille et le centrage du support sur l'objet sont cruciaux.

Autres pistes d'amélioration

Pour améliorer le suivi, on pourrait explorer plusieurs pistes :

- Utiliser plusieurs histogrammes $f(H)$, $f(S)$ et/ou $f(V)$ pour modéliser séparément les différentes composantes de couleur. On peut alors combiner ces histogrammes de différentes manières pour obtenir un poids de rétro-projection plus discriminant.
- Utiliser des techniques plus avancées de suivi, comme l'algorithme de CamShift qui est une extension de Mean Shift pour les histogrammes pondérés. Cela permet de mieux prendre en compte la forme et l'orientation de l'objet à suivre. On peut aussi utiliser des descripteurs de forme ou de texture pour améliorer la robustesse du suivi.

III TRANSFORMÉE DE HOUGH

La transformée de Hough est une technique utilisée pour détecter des formes géométriques dans une image, notamment des lignes, des cercles et des ellipses. Contrairement à la méthode de Mean Shift, la transformation de Hough ne permet pas de segmenter une image en régions homogènes, mais elle permet de détecter des formes spécifiques dans l'image.

Dans cette partie, nous allons mettre en œuvre la transformée de Hough. Dans la question 3, nous allons utiliser la norme du gradient pour définir l'indice de la transformée de Hough, en ne choisissant que les pixels ayant une petite norme de gradient. Ensuite, dans la question 4, nous allons construire un modèle de l'objet sous la forme d'une table R et calculer la transformée de Hough pour chaque trame de la vidéo. Nous suivrons la valeur maximale correspondante.

III.1 Question 3 :

Cette question concerne le calcul de l'orientation locale et du module du gradient des pixels d'une image. Le gradient représente la variation de l'intensité lumineuse des pixels d'une image, et l'orientation locale du gradient indique la direction dans laquelle cette variation est la plus forte.

Pour calculer l'orientation locale et le module du gradient d'une image, on peut utiliser la fonction cv2.Sobel() de la bibliothèque OpenCV. Cette fonction calcule le gradient de l'image dans les directions x et y en utilisant le noyau Sobel.

```
#calcul of gradient
grad_x = cv2.Sobel(frame_gray,cv2.CV_32F,1,0)
grad_y = cv2.Sobel(frame_gray,cv2.CV_32F,0,1)
gradient = np.hypot(grad_x, grad_y)/256
img = cv2.cvtColor(gradient, cv2.COLOR_GRAY2BGR)
seuil=0.3
img[np.where(((img[:,:,:]<seuil).all(axis=2))] = [0,0,255]

cv2.imshow('Module du gradient',img)
# Calcul of orientations
orientation = np.arctan2(grad_x, grad_y)
# Normalisation f the orientations between 0 and 1
orientation = (orientation + np.pi) / (2 * np.pi)
cv2.imshow('Orientation du gradient', orientation)
# Calculate gradient norm
norm = np.sqrt(grad_x**2 + grad_y**2)
# Normalise the norm between 0 and 1
cv2.imshow('Norme du gradient', norm / norm.max())
```

FIGURE 9 – Résultat

Dans ce code, le module du gradient est également calculé pour chaque pixel de chaque image. Ce module représente l'amplitude du changement maximal d'intensité de la couleur. Le seuil est ensuite appliqué au module du gradient pour masquer les pixels dont l'orientation n'est pas significative. En d'autres termes, les pixels dont le module du gradient est inférieur à cette valeur sont masqués en leur attribuant une valeur de 0.

On a défini le seuil avec la variable seuil. La valeur par défaut pour le seuil est 0.3.

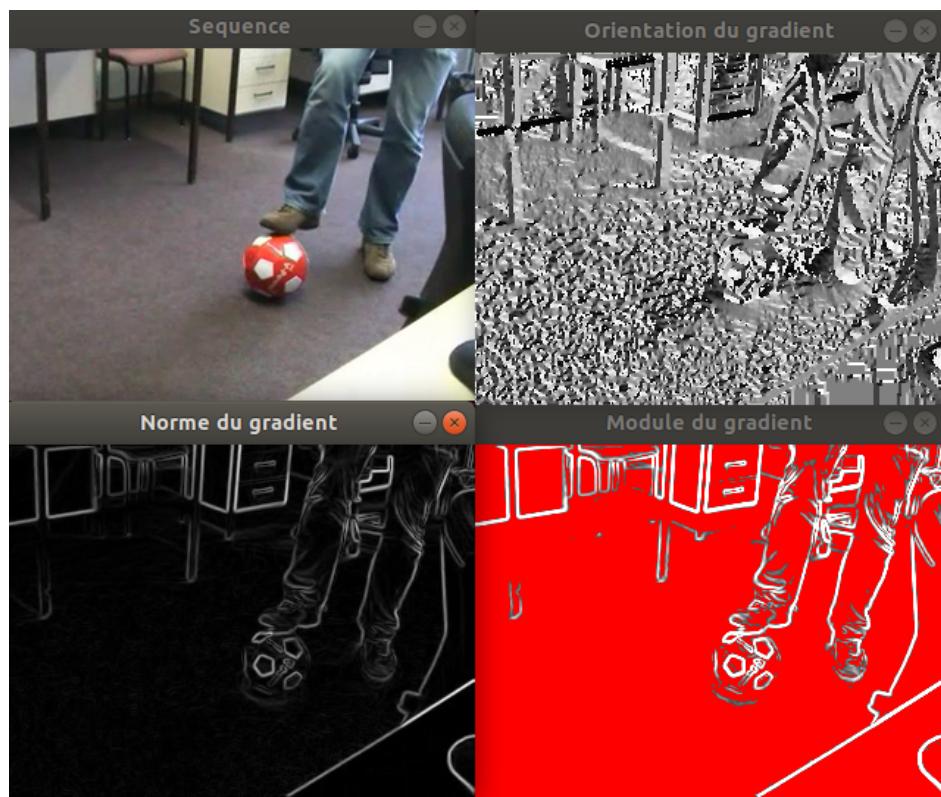


FIGURE 10 – Séquence des orientations avec Seuil = 0.3



FIGURE 11 – Séquence des orientations avec Seuil = 0.75

III.2 Question 4 :

On réalise ici un suivi d'objet sur deux des vidéos fournies (*Antoine_Mug* et *VOT-Ball*). On se base sur la construction d'une R-table réalisée à partir des informations de gradient des images considérées. Plus précisément, notre algorithme se présente comme suit :

- L'utilisateur définit (avec le code fourni) une région d'intérêt dans l'image initiale, celle délimitant l'objet que l'on souhaite suivre.
- Similairement à ce qui a été fait à la question précédente, on calcule l'orientation et le module du gradient en ses différentes coordonnées. L'orientation est ramenée entre 0 et 1 puis "discrétisée" sur un certain nombre de valeurs (8 ici, ceci sera utile pour la suite). Le module est quant à lui divisé par 256.
- On réalise à partir des informations précédentes la **R-table**, structure qui nous permet de caractériser l'objet que l'on souhaite suivre dans la scène. Elle est construite à partir d'un dictionnaire : en chaque point de la zone d'intérêt, si le gradient possède un module assez important (seuil utilisé ici : 0.75), une clé est donnée par l'orientation de celui-ci. Les valeurs associées à cette clé sont des vecteurs (tuples) reliant le centre de la zone d'intérêt aux points associés à cette orientation de gradient.
- La structure étant établie, on calcule en chaque nouvelle image les mêmes informations de gradient en tout point et l'on ajoute s'il y a correspondance de l'orientation du poids (toujours avec une condition sur la valeur du module) sur l'image d'estimation (H) au point courant, déplacé du vecteur associé. On comprend qu'une discréttisation des valeurs des orientations permet une correspondance pour les différents pixels (il semble qu'une valeur "continue" soit trop précise pour tomber dessus un certain nombre de fois).

Le centre de l'objet suivi est alors estimé comme le point de valeur maximale de notre image H .

On montre en figure 12, 13 et 14 les résultats obtenus sur les séquences vidéos mentionnées. Ces dernières présentent des difficultés différentes pour le suivi d'objet (respectivement un ballon et un mug). Dans la première séquence, on initialise notre zone d'intérêt autour du ballon. On observe ainsi au début de l'estimation un pic très net, la structure qui vient d'être initialisée est bien reconnue. Plus tard, au moment où le ballon commence à entrer en rotation, l'estimateur décroche par moments. Cela est probablement dû au fait que la rotation entraîne de notre point de vue un changement de motif en surface du ballon, ce qui limite la possibilité de détection. A un autre instant encore (dernière image), on voit que la confiance sur l'estimation reste faible mais, sans doute parce que le fond plus uniforme empêche les confusions, l'estimation devient meilleure.

La seconde séquence considérée présente un mug en déplacement dans le cadre, avec différentes vitesses. Cette fois-ci une difficulté dans suivi semble provenir du flou engendré par un déplacement trop rapide de l'objet (on voit bien ce phénomène en figure 14).

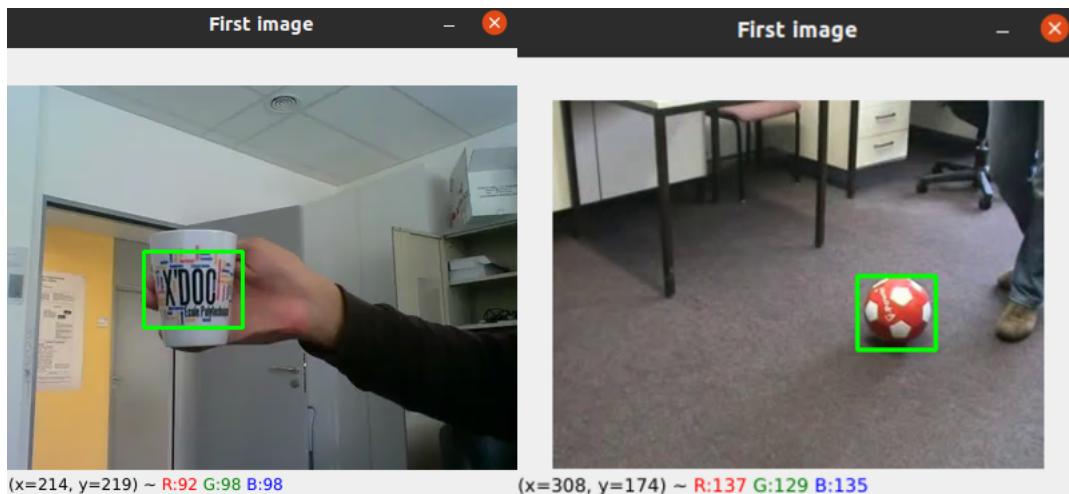


FIGURE 12 – Détermination sur l'image initiale des zones d'intérêt pour les séquences *Antoine_Mug* et *VOT-Ball*

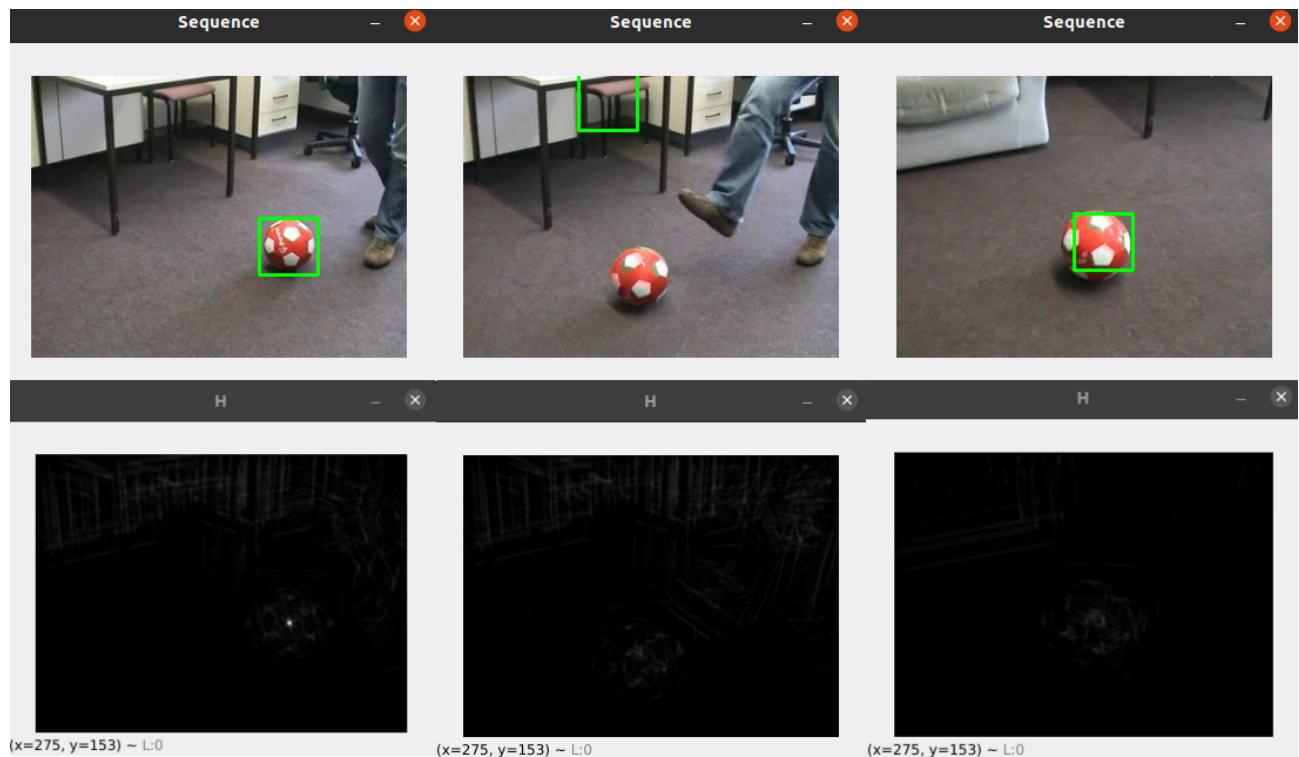


FIGURE 13 – Estimation en trois instants de la position du ballon (en haut) avec l'image des poids accumulés pour l'estimation (en bas) pour la séquence *VOT-Ball*

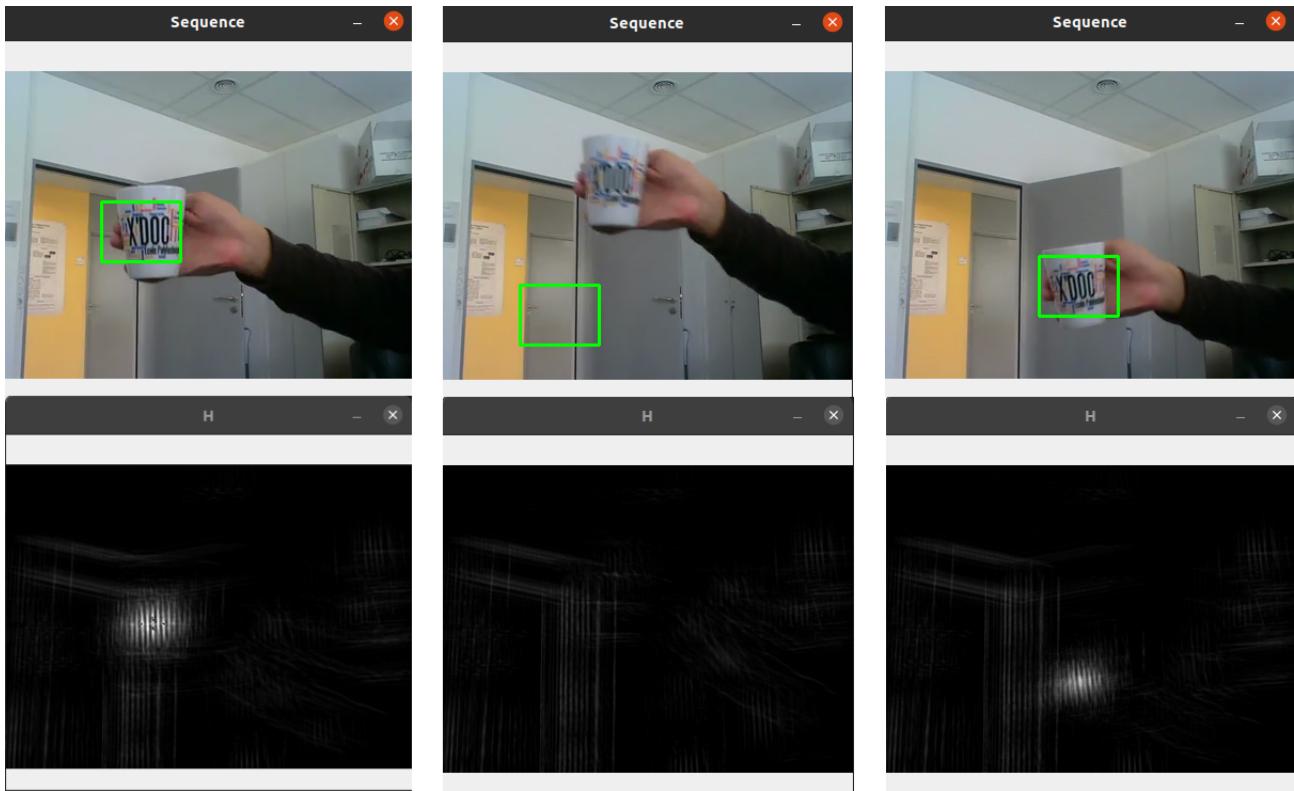


FIGURE 14 – Estimation en trois instants de la position du mug (en haut) avec l'image des poids accumulés pour l'estimation (en bas) pour la séquence *Antoine_Mug*

Ainsi, l'algorithme permet une détection structurelle de l'objet, mais l'on voit que son altération au cours de la séquence vidéo rend la tâche difficile. En revanche, par son exploration globale de l'image, l'algorithme devrait présenter l'avantage d'être capable de répondre à des déplacements brusques voir très discontinus de l'objet suivi.

IV SYNTHÈSE

IV.1 Question 5 :

- Combinaison des méthodes MeanShift et Transformée de Hough :

Il est possible de combiner les deux méthodes pour améliorer la précision du tracking. En utilisant MeanShift pour localiser une région d'intérêt initiale et la Transformée de Hough pour déterminer l'orientation, il est possible d'obtenir une estimation plus précise de la position de l'objet. La combinaison de ces deux méthodes peut également être utilisée pour détecter les changements d'échelle de l'objet en utilisant la Transformée de Hough pour déterminer l'échelle.

- Mise à jour du modèle pour améliorer la robustesse aux changements d'aspects et aux occultations :

Pour rendre le tracking plus robuste aux changements d'aspects et aux occultations, il est possible de mettre à jour le modèle utilisé pour la détection de l'objet en utilisant des techniques d'apprentissage profond. Ces techniques permettent d'apprendre des représentations de l'objet plus robustes aux variations d'aspects et aux occlusions. Par exemple, l'utilisation de réseaux de neurones convolutifs (CNN) pour extraire des caractéristiques discriminantes de l'objet peut aider à améliorer la robustesse du modèle.

- Exploitation des caractéristiques apprises par un réseau profond en association avec les méthodes MeanShift et Transformée de Hough :

Les caractéristiques apprises par un réseau profond peuvent être utilisées en association avec les méthodes MeanShift et Transformée de Hough pour améliorer la précision du tracking. Par exemple, les caractéristiques extraites par un réseau de neurones convolutifs peuvent être utilisées pour initialiser la région d'intérêt de l'objet dans la méthode MeanShift. De plus, la Transformée de Hough peut être utilisée pour déterminer l'orientation de l'objet en utilisant les caractéristiques extraites par le réseau de neurones convolutifs. Cela peut aider à améliorer la précision du tracking dans des situations difficiles, telles que les changements d'aspects et les occlusions.

RÉFÉRENCES

- [1] Mean-shift Tracking , R.Collins, CSE, PSU CSE598G Spring 2006,
<https://www.cse.psu.edu/~rtc12/CSE598G/introMeanShift.pdf>
- [2] On Detection of Multiple Object Instances using Hough Transforms,
<https://www.robots.ox.ac.uk/~vgg/publications/2010/Barinova10/barinova10.pdf>
- [3] Camshift,
[https://fr.wikipedia.org/wiki/Camshift#:~:text=Le%20Camshift%20\(Continuously%20Adaptive%20Mean,dernier%20est%20arriv%C3%A9%20%C3%A0%20convergence](https://fr.wikipedia.org/wiki/Camshift#:~:text=Le%20Camshift%20(Continuously%20Adaptive%20Mean,dernier%20est%20arriv%C3%A9%20%C3%A0%20convergence)
- [4] Unveiling the Power of Deep Tracking , Goutam Bhat1 , Joakim Johnander, Martin Danelljan Fahad Shahbaz Khan and Michael Felsberg
https://openaccess.thecvf.com/content_ECCV_2018/papers/Goutam_Bhat_Unveiling_the_Power_ECCV_2018_paper.pdf