

Rapport TP4 : Analyse vidéo et Tracking

Olivier LAURENT Lisa GIORDANI

Février 2022

Dans ce TP, nous allons étudier deux méthodes utilisées pour le suivi d'objet. Nous aurons pour cela à disposition plusieurs vidéos, ainsi qu'un code basique Tracking_MeanShift.py.

1 Mean Shift

Les résultats obtenus dans cette partie proviennent des fichiers python suivants :

1. Tracking_MeanShift_display.py : code qui permet d'afficher l'histogramme de teinte du modèle initial, les histogrammes marginaux, ainsi que les séquences de poids obtenus à partir de la rétroposition (Questions 1 et 2.1).
2. Tracking_MeanShift_updates.py : code qui permet de modifier la densité calculée (H : Hue, S : saturation, V : value) et qui implémente la stratégie de mise à jour de l'histogramme modèle choisie (Question 2.2).

1.1 Question 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 f_H 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 Principe de l'algorithme Mean Shift

Le principe général de l'algorithme de Mean Shift est de déterminer comment une distribution de points se déplace dans l'espace, au cours du temps. Cet algorithme est donc utilisé pour suivre le déplacement d'objets dans des vidéos. Il peut également être utilisé pour séparer l'arrière-plan statique des objets en mouvement dans une vidéo.

Avant toute chose, il faut choisir quelle valeur $g(x_i)$ on va associer à chaque point x_i de l'image pour pouvoir le caractériser (partiellement). Dans ce TP, on va utiliser le modèle TSV pour Teinte Saturation Valeur, également appelé en anglais HSV pour Hue Saturation Value). Dans cette première question, on n'utilisera que la teinte (H), mais par la suite on utilisera aussi la saturation (S) et la luminosité (V).

Premièrement, l'algorithme demande à l'utilisateur de sélectionner, sur la première image de la vidéo, l'objet qu'il souhaite suivre dans une fenêtre englobante (RoI). A partir de cette fenêtre, l'histogramme modèle est calculé : il représente la distribution locale de la grandeur g choisie. Il est obtenu grâce à la formule suivante :

$$f_g(u) = \frac{\sum_S K(x - x_i) \delta_u^{g(x_i)}}{\sum_S K(x - x_i)}$$

avec les grandeurs suivantes :

- S : surface définie par la fenêtre englobante (RoI) centrée sur x
- K : fonction kernel qui évalue la contribution de chaque point à x (fonction souvent gaussienne)
- g : fonction qui attribue une valeur à chaque pixel (par exemple, HSV)
- $\delta_u^v = 1$ si $u = v$ et $\delta_u^v = 0$ si $u \neq v$

Cet histogramme est utilisé dans la suite de l'algorithme en tant qu'histogramme de référence.

A chaque nouvelle image de la vidéo, l'algorithme Mean shift calcule une nouvelle fenêtre englobante de suivi à partir de la rétroprojection R_g de l'histogramme modèle sur les valeurs de g sur l'image actuelle :

$$R_g(x) = f_g(g(x))$$

Cette rétro-projection permet d'établir une carte de confiance, c'est-à-dire une fonction de densité de probabilité, sur la nouvelle image. La probabilité attribuée à chaque pixel de la nouvelle image est la probabilité que la valeur $g(x_i)$ du pixel soit également présente dans l'objet (défini par le ROI) de l'image précédente. En général, on retrouve un pic dans la carte de confiance, situé dans une zone proche de l'ancienne position de l'objet. La prochaine fenêtre englobante décrivant la position de l'objet se situera donc dans une zone de distribution de probabilité élevée.

Enfin, la direction du mouvement dépend de la différence entre le centre de la dernière fenêtre de suivi et le centroïde de tous les k-pixels à l'intérieur de cette fenêtre. Ainsi, l'algorithme Mean shift se déplace itérativement d'un point vers la moyenne des points de son voisinage.

1.1.2 Avantages

Dans les deux parties suivantes (2.1.2 et 2.1.3), on associera à chaque pixel d'une image la valeur de sa teinte. Ainsi, $g(x) = H(x)$

On teste l'algorithme de tracking implémenté dans le fichier Tracking_MeanShift.py sur plusieurs vidéos.

Antoine_Mug.mp4 :

L'algorithme arrive à suivre le mug correctement au début de la vidéo, c'est-à-dire lorsque le mug reste vers le centre de la caméra.



FIGURE 1 – Mean shift appliqué sur la vidéo d'un mug en mouvement

VOT-Ball.mp4 :

La fenêtre englobante de tracking suit le ballon même lorsque celui-ci se déplace assez vite.



FIGURE 2 – Mean shift appliqué sur la vidéo d'un ballon en mouvement

Pour pouvoir observer les différentes itérations de l'algorithme au cours des séquences de la vidéo, on fait afficher les histogrammes de teinte modèle et marginaux. Pour rappel, l'histogramme modèle de teinte est calculé sur la fenêtre englobante (RoI) initiale défini par l'utilisateur manuellement. Quant aux histogrammes marginaux, ce sont les histogrammes de teinte calculé sur la suite des fenêtres de suivi obtenues par l'algorithme Mean shift (à partir des rétro-projections) tout au long de la vidéo.

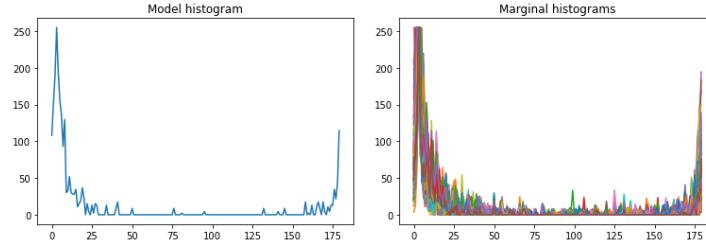


FIGURE 3 – Histogrammes de teinte modèle (gauche) et marginaux (droite)

On remarque que l'allure des histogrammes marginaux est proche de celle de l'histogramme modèle. Pour rappel, dans cette première version du code, l'histogramme modèle n'est pas mis à jour au cours de l'exécution de l'algorithme. Dans ce cas, la rétro-projection de l'histogramme modèle sur la carte de teinte des nouvelles images aura permis de suivre l'objet correctement.

1.1.3 Limitations

Nous étudions à présent les limitations de l'algorithme Mean shift en l'appliquant à plusieurs autres vidéos.

VOT-Sunshade.mp4 :

On remarque que très rapidement l'algorithme n'arrive plus à suivre l'homme lorsqu'il passe de la zone lumineuse à celle à l'ombre, et inversement.

On peut ainsi supposer que la rétro-projection de l'histogramme modèle sur la carte de teinte des nouvelles images n'a pas permis d'établir une zone de distribution de probabilité élevée à privilégier. La rétro-projection semble comporter des ambiguïtés qui ne permettent pas à l'algorithme de suivre l'homme correctement.

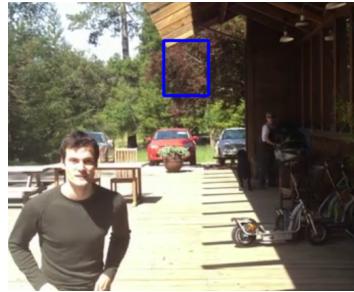


FIGURE 4 – Mean shift appliqué sur la vidéo d'un homme dont l'exposition alterne entre lumière et ombre

VOT-Woman.mp4 :

On étudie une autre vidéo où une femme marche sur un trottoir à côté duquel sont garées plusieurs voitures. Très rapidement la fenêtre englobante ne suit plus la femme, mais semble être "attirée" par les voitures.



FIGURE 5 – Mean shift appliqu   sur la vid  o d'une femme qui marche dans la rue

De plus, on peut voir que l'allure des histogrammes marginaux (ci-dessous) est assez diff  rente de celle de l'histogramme mod  le. Il y a des pics correspondant    des valeurs de teinte qu'on ne retrouve pas dans la fen  tre initiale d  finissant la femme. Ces pics ont   t   obtenus lorsque la fen  tre de suivi   tait situ  e sur les voitures. Si l'algorithme avait r  ussi    suivre correctement la femme, ces pics ne seraient pas apparus sur les histogrammes marginaux.

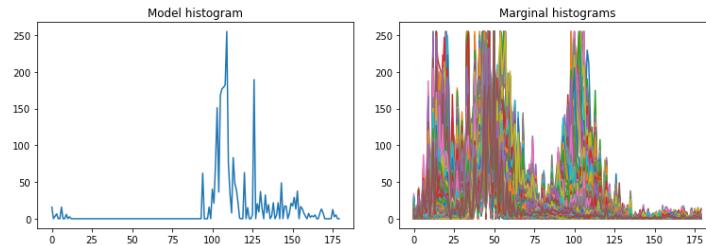


FIGURE 6 – Histogrammes de teinte mod  le (gauche) et marginaux (droite)

VOT-Car.mp4 :

Cette vid  o a   t   film  e    bord d'une voiture. On y voit plusieurs voitures circuler sur une route. On demande    l'algorithme de suivre la voiture situ  e juste avant celle dans laquelle se situe la cam  ra. L'algorithme n'arrive pas du tout    suivre cette voiture, il semble la confondre avec la route. Cela peut   tre d   au fait que la voiture soit grise tout comme la route, leur teinte est donc proche. D'ailleurs, on remarque que l'allure des histogrammes marginaux est tr  s proche de celle de l'histogramme mod  le de teinte, ce qui va dans le sens de cette hypoth  se.

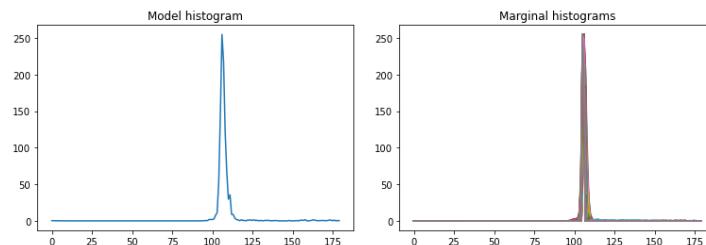


FIGURE 7 – Histogrammes de teinte mod  le (gauche) et marginaux (droite)

VOT-Basket.mp4 :

Cette vidéo filme un match de basketball. On demande à l'algorithme de suivre un joueur, mais il a de grandes difficultés à le faire. De plus, si on lui demande de suivre l'arbitre qui sort du champs quelques secondes, l'algorithme suit à la place d'autres joueurs pendant l'absence de l'arbitre, puis il n'arrive jamais à retrouver l'arbitre lorsque celui re-rentre dans le champs. On préfèrera que l'algorithme ne fasse pas apparaître de fenêtre de suivi lorsque l'objet sort du champs.

Remarques générales :

L'utilisation de l'algorithme Mean shift pour le suivi d'objet présente d'autres inconvénients.

La taille de la fenêtre de suivi est fixe et ne s'adapte donc pas au changement de profondeur de la position de l'objet vis-à-vis de la caméra (perspective).

De plus, il est nécessaire de faire appel à un utilisteur pour sélectionner la région où se situe l'objet dans la première image, puisque l'histogramme modèle est calculé à partir de cette information. On pourrait imaginer combiner cet algorithme avec un algorithme permettant de détecter et classifier les objets d'une image, pour éviter cette action manuelle qui peut également être source d'imprécision.

1.2 Question 2

Analyser plus finement le résultat en affichant la séquence des poids à partir de la rétropénétration R_H de l'histogramme f_H de teinte, définie par $R_H(x; y) = f_H(H(x; y))$. 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 Analyse des résultats

A présent, on va analyser plus finement les résultats obtenus dans la partie précédente en affichant une carte des poids de la rétro-projection utilisée dans l'algorithme Mean shift. Pour ce faire, on binarise les valeurs de la retro-projection pour afficher les poids sous la forme d'une carte en noir et blanc.

VOT-Ball.mp4 :

Sur cette vidéo, le ballon est d'une couleur (rouge) différente du reste des éléments de la vidéo. La teinte des pixels permet donc de différencier assez correctement le ballon dans les images. C'est ce que l'on remarque également sur la carte des poids ci-dessus. Cela explique le fait que l'algorithme arrive plutôt bien à suivre le ballon.



FIGURE 8 – Mean shift appliqué sur la vidéo d'un ballon en mouvement (gauche), séquence de poids utilisée par l'algorithme (droite) en utilisant la teinte des pixels

VOT-Woman.mp4 :

Comme dit à la question 1, lorsqu'on utilise la teinte des pixels, l'algorithme n'arrive pas à suivre la femme et semble être attiré par les voitures. On peut voir que la silhouette de la femme n'apparaît pas dans la carte des poids, alors qu'on arrive à distinguer une voiture.



FIGURE 9 – Mean shift appliqu   sur une vid  o d'une femme marchant dans la rue (gauche), s  quence de poids utilis  e par l'algorithme (droite) en utilisant la teinte des pixels

En prenant comme grandeur associ  e    chaque pixel la saturation, l'algorithme n'arrive pas    mieux suivre la femme puisque cette fois-ci il a des difficult  es    la distinguer de l'arri  re plan (arbres et barri  re) de l'image.

Lorsqu'on utilise la luminosit  , au d  but de la vid  o l'algorithme arrive    suivre la femme, mais assez rapidement il la perd. N  anmoins, on voit que la silhouette de la femme appara  t l  g  rement et que l'arri  re-plan et la voiture n'ont pas des poids aussi lev  s que dans les deux cas pr  c  dents. L'utilisation de la luminosit   semble   tre la meilleure solution entre H, S et V, m  me si le suivi n'est pas complet.

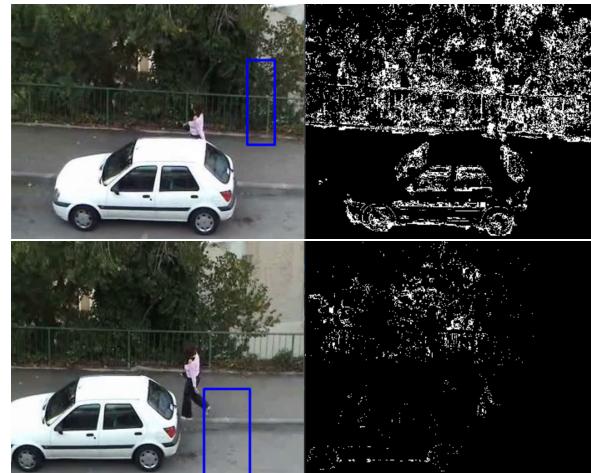


FIGURE 10 – Mean shift appliqu   sur une vid  o d'une femme marchant dans la rue en utilisant la saturation (haut) / luminosit   (bas) des pixels

Antoine_Mug.mp4 :

On étudie les résultats obtenus en utilisant la teinte, la saturation et la luminosité des pixels. On remarque qu'on distingue mieux le mug sur la carte des poids de la rétro-projection lorsqu'on utilise la teinte. Dans ce cas, il peut arriver que la fenêtre de suivi reste bloquée dans le bas de l'image et on remarque que cette zone correspond bien à une distribution de probabilité élevée.

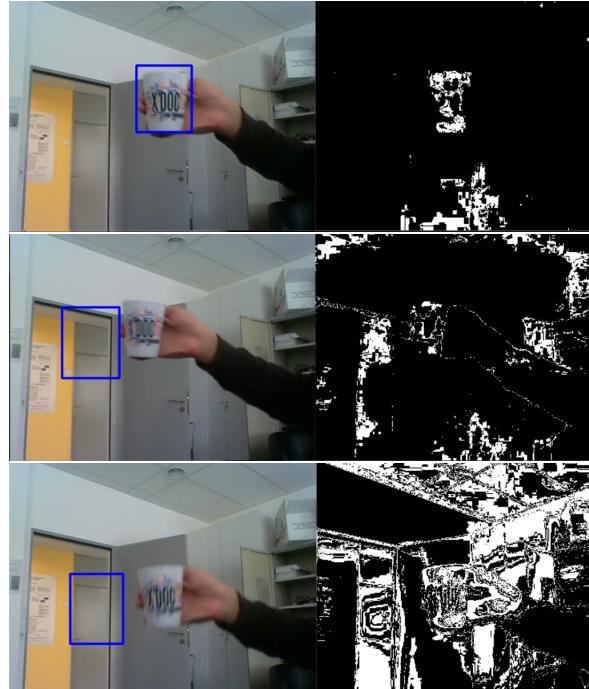


FIGURE 11 – Mean shift appliqué d'une femme marchant dans la rue en utilisant la teinte (haut) / saturation (milieu) / luminosité (bas) des pixels

VOT-Basket.mp4 :

En utilisant la luminosité des pixels, l'algorithme parvient assez bien à suivre un joueur car il arrive à le distinguer des autres éléments (sol, public...).



FIGURE 12 – Mean shift appliqué sur une vidéo d'un match de basket en utilisant la luminosité des pixels

VOT-Car.mp4 :

Lorsqu'on utilise la teinte, l'algorithme n'arrive pas à distinguer la voiture de la route. Cela confirme nos observations de la question 1. En utilisant la saturation, l'algorithme confond également la voiture avec la route mais de façon moins importante. Quant à l'utilisation de la luminosité, elle permet de distinguer la voiture de la route, mais cette fois-ci ce sont les arbres qui se confondent avec la voiture. Dans aucun de ces trois cas, l'algorithme ne parvient à suivre la voiture.

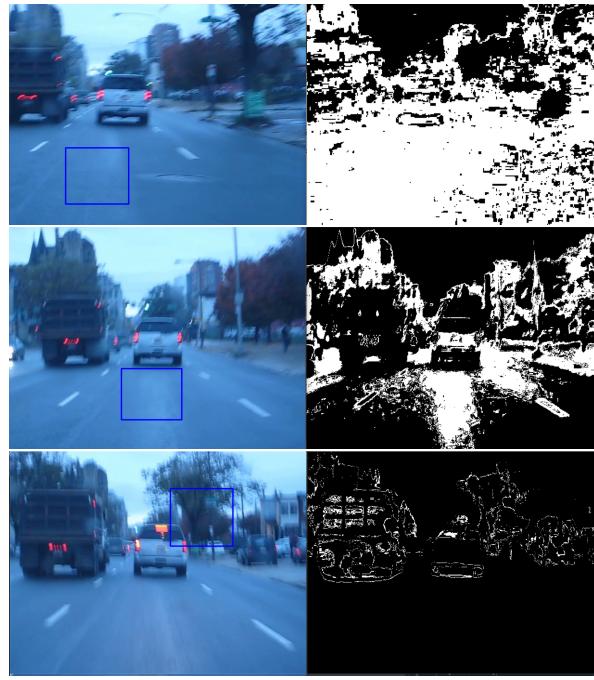


FIGURE 13 – Mean shift appliqu  sur une vid o de circulation routi re en utilisant la teinte (haut) / saturation (milieu) / luminos t  (bas) des pixels

VOT-Sunshade.mp4 :

Dans les deux premiers cas (teinte et saturation), la silhouette de l'homme se confond avec l'arri re-plan. Ce n'est pas le cas lorsqu'on utilise la luminos t , cela semble  tre la meilleure solution. N anmoins, cette solution n'est pas satisfaisante pour autant, parce que l'algorithme n'arrive pas   suivre l'homme.



FIGURE 14 – Mean shift appliqu  sur une vid o d'un homme passant d'une zone lumineuse   ombrag e e en utilisant la teinte (haut) / saturation (milieu) / luminos t  (bas) des pixels

1.2.2 Améliorations

Comme vu précédemment, on peut améliorer les performances de l'algorithme en modifiant la valeur associée aux pixels $g(x)$. On peut choisir parmi la teinte (H), la saturation (S) ou la luminosité (V). Pour ce faire, il faut premièrement observer les caractéristiques de la vidéo sur laquelle on souhaite effectuer le suivi (changement d'exposition, luminosité, contraste...). Ensuite, on peut faire afficher les poids de la rétro-projection utilisée par l'algorithme pour choisir la grandeur qui permettra de repérer au mieux l'objet à suivre parmi les autres objets et l'arrière-plan.

D'autre part, on peut mettre en place une stratégie de mise à jour de l'histogramme modèle au cours de la vidéo.

Jusqu'à présent, l'histogramme modèle était celui défini à l'initialisation par l'utilisateur et n'était jamais modifié. Cela peut poser des problèmes lorsque un objet tourne par rapport à la caméra, par exemple. Dans ce cas, il pourrait être utile d'utiliser l'histogramme marginal de la dernière fenêtre de suivi en tant qu'histogramme modèle, puisqu'il a plus de chance d'être similaire à celui de la prochaine image.

On implémente alors cette stratégie selon laquelle l'histogramme modèle est mis à jour à chaque nouvelle image de la vidéo par le dernier histogramme marginal. Un autre problème se pose alors dans le cas où l'algorithme commet une erreur et suit un autre objet. En effet, il continuera à suivre ce nouvel objet jusqu'à la fin de la vidéo sans jamais rectifier son erreur. Ce comportement a par exemple été visible dans la vidéo VOT-Mug.mp4 où l'algorithme se met subitement à suivre la main au lieu de suivre le mug après un mouvement plus rapide du mug, et ce jusqu'à la fin de la vidéo.

On décide donc de mettre en oeuvre une stratégie différente de mise à jour de l'histogramme modèle. Lorsque la moyenne des poids de la rétro-projection dans la fenêtre de suivi actuelle est supérieure à un certain seuil, on met à jour l'histogramme modèle avec l'histogramme marginal actuel. En effet, si la moyenne des poids est élevée dans la fenêtre, on considère que la probabilité que l'objet se situe dans cette fenêtre est également élevée. On met donc à jour l'histogramme modèle pour tenir compte des éventuels changements qui pourraient avoir lieu sur l'objet (rotation, éloignement...).

La difficulté réside à présent dans le choix du seuil que l'on appellera dst_ref . Ce seuil est initialisé avec la valeur, arrondie à l'unité, de la moyenne de la rétro-projection dans la fenêtre de suivi initiale (tracée par l'utilisateur). Ensuite, à chaque fois que ce seuil est dépassé, il est lui aussi mis à jour par la valeur de la moyenne de la rétro-projection dans la fenêtre de suivi actuelle, arrondie à l'unité. Cela permettra au seuil de s'adapter au nouvel histogramme modèle.

Afin d'éviter que l'algorithme ait complètement la trace de l'objet en ayant mis à jour l'histogramme modèle sur le mauvais objet, on conserve le premier histogramme modèle dans la variable $model_hist_init$ et son seuil associé dst_ref_init . Si la moyenne de la rétro-projection sur la fenêtre de suivi actuelle est inférieure à $dst_ref_init/2$, on considère que l'algorithme s'est trop éloigné de l'histogramme de l'objet initial, alors on ré-initialise l'histogramme modèle avec l'histogramme modèle initial $model_hist_init$.

En appliquant cette dernière stratégie de mise à jour de l'histogramme modèle, on obtient de meilleurs résultats. Par exemple, lorsque l'algorithme perd la trace du mug dans la vidéo Antoine_Mug.mp4, il arrive à la retrouver au bout de quelques secondes. Cependant, cela ne fonctionne pas à chaque exécution de l'algorithme. De plus, dans d'autres vidéos comme VOT-Sunshade.mp4 l'algorithme décrit comme tel n'arrive toujours pas à suivre l'homme. Cela est dû au fait que l'algorithme peut difficilement être amélioré lorsque la carte des poids de la rétro-projection en permet pas de distinguer l'objet à suivre du reste des objets et de l'arrière-plan. Une alternative envisageable pourrait être de donner à l'algorithme Mean Shift une carte de rétro-projection binarisée pour savoir si cela a une influence sur les performances de l'algorithme.

2 Transformée de Hough

Dans cette partie, nous utiliserons les codes suivants :

- *Tracking_Hough_display.py* : affiche pour chaque trame l'orientation et le module du gradient, ainsi que les pixels non votants (question 3).
- *Tracking_Hough_compute.py* : calcule la transformée de Hough et localise son maximum (question 4).
- *Tracking_Hough_update.py* : utilise différentes méthodes pour calculer le maximum de la transformée de Hough et met à jour la R-Table (question 5).

2.1 Question 3

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. L'objectif de cette question est de définir l'index de la Transformée de Hough (l'orientation), ainsi que l'ensemble des pixels votants, i.e. ceux dont l'orientation est significative.

2.1.1 R-Table

La méthode de suivi basée sur les transformées de Hough nécessite la construction d'une R-Table. Cette table stocke les votes (vecteurs) des pixels votants classés (en index) en fonction de l'orientation du gradient au niveau du pixel. Pour remplir une telle table, nous avons donc besoin de définir quels pixels sont votants et également l'index (orientation du gradient) de la table.

Dans la figure ci-dessous, les pixels votants sont les seuls pixels représentés (par des points noirs). La sous-figure de droite représente les différents index de cette R-Table.

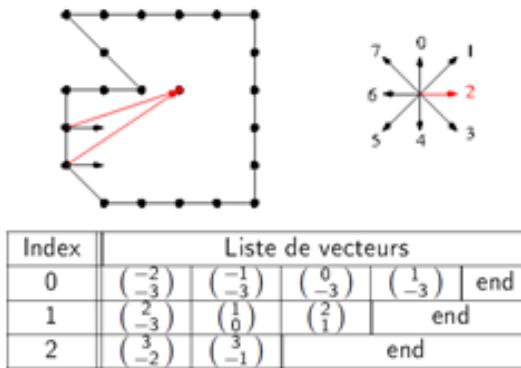


FIGURE 15 – Exemple de R-table

2.1.2 Index de la transformée de Hough

On commence par calculer pour chaque trame l'orientation du gradient. On calcule l'argument du gradient des pixels de l'image. Pour cela, on utilise les dérivées de Sobel selon l'axe x et selon l'axe y. L'opérateur de Sobel est un opérateur de différenciation discrète qui combine le lissage gaussien et la différenciation. Il calcule une approximation du gradient de la fonction d'intensité d'une image. Ensuite, on place les deux dérivées en argument de la fonction *arctan*. On obtient des angles compris entre $-\pi$ et π . On transforme ces angles en radians en degrés et on approche chaque angle à la dizaine près. Chaque pixel a donc à présent comme orientation de gradient une valeur divisible par 10 et comprise entre -170 et 180. La R-Table possède donc un index de taille 36.

Pour afficher les orientations du gradient sur une image en noir et blanc, on normalise temporairement les orientations en 0 et 1.

2.1.3 Pixels votants

On calcule également le module du gradient. Pour ce faire, on utilise également les dérivées de Sobel. Ensuite, on calcule l'hypoténuse d'un triangle rectangle dont les côtés sont de même longueur que les dérivées. L'hypoténuse est le support du gradient.

Ensuite, on définit un seuil pour le module du gradient au dessus duquel les pixels sont considérés votants. La norme du gradient pouvant varier entre 0 et 255 (clippé), on choisit un $seuil = 100$. On fait afficher en rouge les pixels dont l'orientation n'est pas assez significative pour être votants.

A l'aide du code *Tracking_Hough_display.py*, on obtient les résultats suivants sur la vidéo *Antoine_Mug.mp4* :

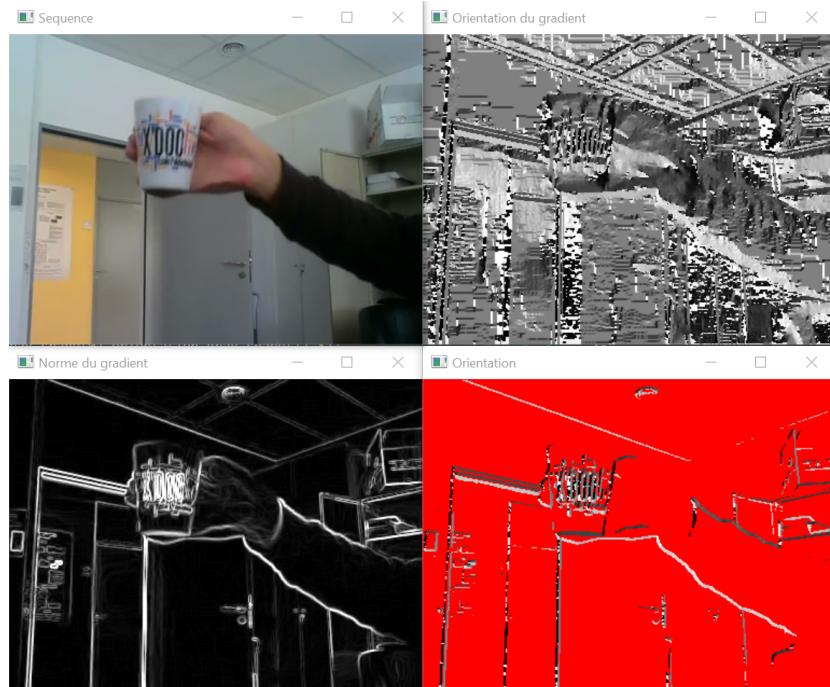


FIGURE 16 – Calcul de l'index de vote (orientation du gradient), avec sélection des pixels votants (norme du gradient)

2.2 Question 4

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. Illustrer vos réponses en montrant des exemples de transformées de Hough et des détections correspondantes.

2.2.1 Calcul de la transformée de Hough

La principe de la transformées de Hough est d'indiquer par une liste de votes des hypothèses de localisation. Ainsi, plus il y a de votes pour une localisation, plus elle est probable.

On note H la transformée de Hough. On utilise une R-Table pour organiser les votes afin de calculer H . Il faut donc commencer par en construire une en suivant la description faite précédemment.

Ensuite, on peut calculer la transformée de Hough en suivant l'algorithme ci-dessous.

Initial: $H(x) = 0$ partout. Pour tout point x de l'image, soit $\lambda(x)$ la dérivée quantifiée. Pour toute occurrence j de la R-Table associée à $\lambda(x)$, faire : $H(x + \vec{\delta}_{\lambda(x)}^j) += \omega_{\lambda(x)}^j$	R-Table pondérées : $\{i, \{\vec{\delta}_i^j, \omega_i^j\}_j\}_i$
--	---

FIGURE 17 – Algorithme de calcul de la transformée de Hough (H) avec $\lambda(x)$ l'orientation de x

Ici, on considère que chaque pixel a le même poids dans le vote. Donc, $\omega_i^j = 1$.

2.2.2 Localisation du maximum de la transformée de Hough

Le maxima de H est le meilleur objet candidat parce que c'est celui ayant reçu le plus de votes. Après l'avoir localisé, on associe sa position au centre de la prochaine fenêtre de suivi.

Il est possible que plusieurs localisations sur l'image aient comme valeur $H(x) = H_{max}$. On choisit alors de considérer comme localisation correspondant au maximum, le barycentre de ces localisations. On aurait pu également choisir de manière aléatoire une localisation parmi celles qui correspondent à un maximum de H , mais cette solution nous a paru moins robuste que la précédente.

2.2.3 Analyse des résultats

A l'aide du code *Tracking_Hough_compute.py*, on obtient les résultats suivants sur la vidéo *Antoine_Mug.mp4* :

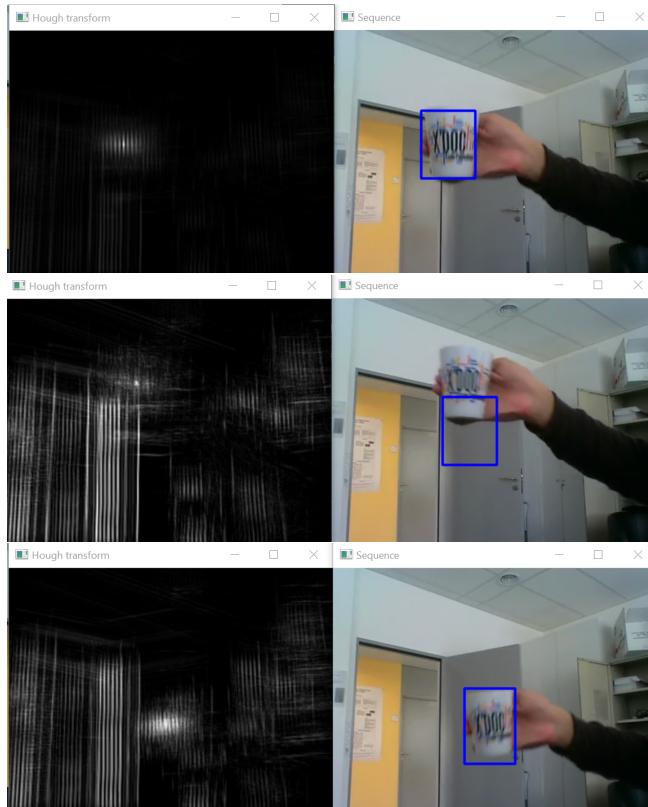


FIGURE 18 – Tracking par transformée de Hough (temps croissant de haut en bas)

Au début de la vidéo, l'algorithme parvient à suivre correctement le mug comme on peut le voir sur les images en haut de la figure 18. On remarque d'ailleurs que la position de la fenêtre de tracking et celle du mug correspondent bien au maximum de la transformée de Hough.

Ensuite, l'algorithme commence à perdre la trace du mug, ce qui est visible sur les images du milieu de la figure 18. Cela peut être expliqué par le fait que plusieurs localisations semblent être égales au maximum de H . On peut effectivement voir sur l'image de gauche que plusieurs zones sont très lumineuses.

Néanmoins, on remarque que l'algorithme arrive à retrouver la trace du mug et à le suivre de nouveau dans la suite de la vidéo, comme en témoigne les images du bas de la figure 18. D'ailleurs, la localisation du mug correspond bien au maximum de H qui semble être plus facilement distinguable des autres valeurs élevées qu'auparavant.

D'autre part, le temps d'exécution de l'algorithme est très long. Cela est causé par le nombre important de pixels votants (aux alentours d'un million) pour chaque trame. Nous pourrions accélérer la vitesse d'exécution de l'algorithme en augmentant le seuil définissant quels pixels peuvent voter. On peut également réduire la taille de l'index de la R-Table qui est actuellement de 36 en approximant l'orientation du gradient de manière plus grossière.

2.3 Question 5

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.

2.3.1 Calcul du maximum de H avec Mean Shift

Jusqu'à présent, on calculait la nouvelle fenêtre de suivi en utilisant le barycentre des localisations correspondant au maximum de H . Cette fois-ci, on utilise l'algorithme Mean Shift appliqué à la transformée de Hough H afin de calculer la prochaine fenêtre de suivi.

A l'aide du code *Tracking_Hough_update.py*, on obtient les résultats suivants sur la vidéo *Antoine_Mug.mp4* :

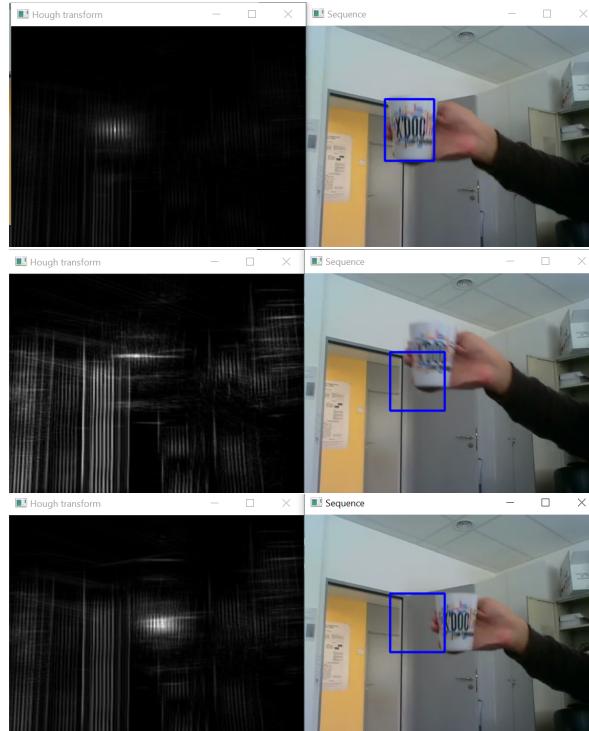


FIGURE 19 – Tracking par transformée de Hough (temps croissant de haut en bas) en calculant le maximum de H avec Mean Shift

Au début de la vidéo, l'algorithme parvient à suivre correctement le mug comme c'était le cas dans la version précédente (calcul classique du maximum de H). Au même moment qu'auparavant, l'algorithme perd la trace du mug.

Cependant, cette fois-ci l'algorithme utilisant Mean Shift pour calculer la nouvelle fenêtre de suivi, ne parvient jamais à suivre à nouveau le mug une fois sa trace perdue.

D'autre part, on remarque que le temps d'exécution de l'algorithme est encore plus long qu'auparavant, ce qui le rend assez peu agréable à utiliser en pratique.

2.3.2 Mise à jour de la R-Table

Jusqu'à présent la R-Table n'était calculée qu'une seule fois au début de l'algorithme. Elle était calculée à partir des informations données par l'utilisateur (RoI tracée manuellement). Le fait de ne jamais mettre à jour la R-Table peut poser un problème si l'objet change d'angle par rapport à la caméra laissant apparaître une nouvelle face différente de la précédente. Cela peut également poser un problème si l'objet se déforme au cours de la vidéo. C'est pourquoi il peut être préférable de mettre à jour la R-Table au cours de la vidéo. Nous choisissons d'effectuer un nouveau calcul de la R-Table à chaque nouvelle trame.

Nous appliquons cette mise à jour de la R-Table tout en calculons de manière classique le maximum de H . Puis, nous appliquons cette mise à jour combinée à l'algorithme Mean Shift pour le calcul de la fenêtre de suivi. Nous obtenons des résultats très similaires présentés ci-dessous :

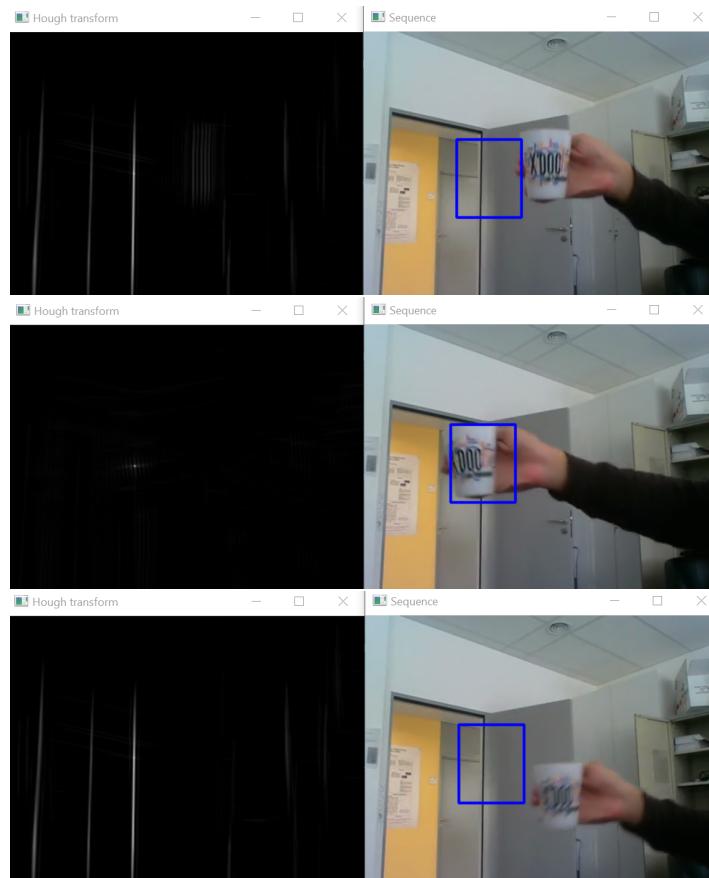


FIGURE 20 – Tracking par transformée de Hough (temps croissant de haut en bas) avec mise à jour de la R-Table

Au début de la vidéo, l'algorithme perd assez vite la trace du mug, car il semble être attiré par une autre zone (partie du cadre de la porte). On observe d'ailleurs que la fenêtre de suivi est située sur cette droite de maxima.

Ensuite, le mug passe devant cette zone (cf images du milieu), on peut donc espérer que l'algorithme retrouve la trace du mug, puisque la transformée de Hough se rapproche de son état initial avec un seul maximum situé au niveau du mug. Cependant, ce n'est pas le cas comme on peut le voir sur les images du bas de la figure 21. La mise à jour de la R-Table n'a pas permis à l'algorithme de retrouver la trace du mug et de donc de le suivre correctement.

L'algorithme le plus satisfaisant semble donc être celui qui utilise la méthode classique pour calculer le maximum de H , et sans mise à jour systématique de la R-Table à chaque nouvelle trame. Nous pourrions mettre en oeuvre une stratégie plus fine pour mettre à jour la R-Table uniquement lorsque le moment semble opportun.