

TD3 rapport ROB313

Analyse vidéo et Tracking

KAMINSKYI Nazar-Mykola
MAVLIUTOV Yaroslav

Janvier 2020

1 Mean Shift

1.1 Q1

1.1.1 *Le principe de l'algorithme Mean-Shift*

Le Mean Shift est une technique d'analyse non paramétrique de l'espace caractéristique pour localiser le maximum d'une fonction de densité, un algorithme de recherche du mode. Les domaines d'application incluent l'analyse de cluster dans la vision par ordinateur et le traitement d'image. C'est-à-dire que nous construisons un modèle pour un objet spécifique à partir d'un paramètre spécifique, puis nous essayons de trouver des correspondances du modèle donné avec des parties de l'image. Ceci est fait en utilisant une fonction de densité de probabilité discrète. Avec l'aide du décalage, nous trouvons juste le maximum de densité et donc très probablement l'emplacement du modèle dans l'image.

1.1.2 *Les expériences*

Passons à la partie expérimentale, utilisons le code fourni.

Les images 1, 2, 3 et 4 montrent la sélection d'un objet pour la construction du modèle (rectangle vert) et le suivi de l'objet sélectionné (rectangle bleu). Les images 5 et 6 présentent la densité marginale sur la composante H (Nous transformons l'image de RGB en HSV pour faciliter le travail). Ce sont ces images qui représentent la probabilité que le modèle coïncide avec différentes parties d'un cadre particulier de vidéo. Sur ces images, vous pouvez reconnaître les formes des images réelles.

Dans un certain nombre d'expériences, L'algorithme Mean Shift parvient à suivre les objets sélectionnés en temps réel.

Les points forts de cette approche contribuent à cela:

1. Méthode non paramétrique;
2. Facile à calculer;



Figure 1: Première image de la Figure 2: Suivre l'objet sélectionné sur vidéo "Antoine-Mug" avec l'objet la vidéo "Antoine-Mug". sélectionné.



Figure 3: Première image de la vidéo Figure 4: Suivre l'objet sélectionné sur "VOT-Ball" avec l'objet sélectionné. la vidéo "VOT-Ball".

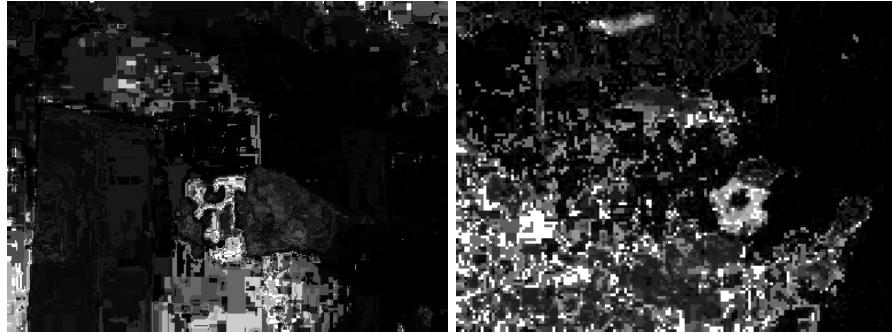


Figure 5: La densité marginale sur la composante H sur la vidéo "Antoine- la composante H sur la vidéo "VOT-Mug" avec l'objet sélectionné (La figure 1).
 Figure 6: La densité marginale sur la vidéo "Antoine- la composante H sur la vidéo "VOT-Mug" avec l'objet sélectionné (La figure 1).

Cependant, dans de nombreux cas, l'algorithme Mean Shift ne peut pas suivre correctement les objets sélectionnés. Cela peut être vu dans les images suivantes 7, 8, 9 et 10.

Cela est dû aux facteurs suivants:

1. L'algorithme est sensible à la transformation des objets;
2. L'algorithme est sensible à la mise à l'échelle des objets;
3. L'algorithme est sensible à la rotation des objets;
4. L'algorithme est sensible à la taille de la fenêtre initiale pour déterminer le modèle;
5. L'algorithme est sensible à la luminosité et au contraste des objets dans différentes parties de l'image.



Figure 7: Fausse détection d'objets: le mug.
Figure 8: Fausse détection d'objets: la voiture.

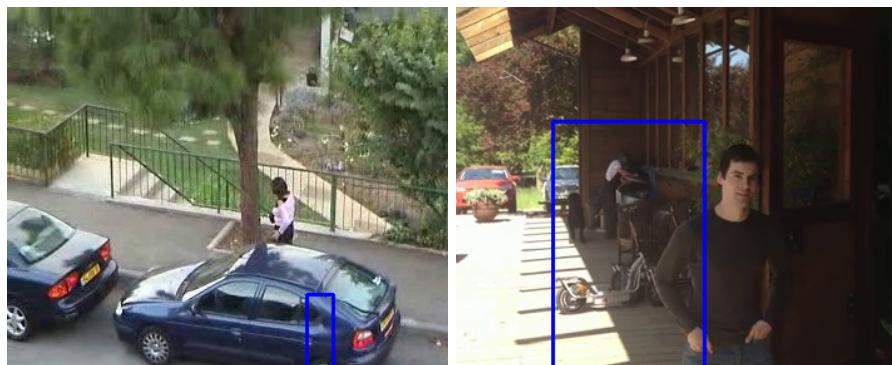


Figure 9: Fausse détection d'objets: la femme.
Figure 10: Fausse détection d'objets: l'homme.

1.2 Q2

Comme on peut le voir, l'algorithme standard Mean Shift a des inconvénients, c'est pourquoi les améliorations suivantes ont été suggérées:

1. Créer des poids pour les valeurs de densité comme une distribution gaussienne en fonction de la distance au centre du rectangle.
2. Recalculer le modèle à chaque fois si **MSE**(mean squared error) entre le modèle dans la rectangle actuelle et le modèle recherché est plus de 60000 unités.

Des expériences ont été menées pour deux cas.

1.2.1 La création de poids pour la densité

Le prototype de poids peut être vu sur la figure 11 et pour le code voir le listing 1. Il convient de noter que le calcul des poids se fait dans un rectangle avec des côtés deux fois plus grands que ceux initialement définis, nous prenons en compte la capacité du objet à se déplacer.



Figure 11: Le prototype de poids.

```

1 def f_dst_weights(frame, x,y,w,h):
2     X, Y, _ = frame.shape
3     weights = np.zeros((X, Y)) + 0.15
4
5
6     # defining a zone of curiosity
7     ww = min(x+int(1.5*w), X) - max(x-int(w/2), 0)
8     hh = min(y+int(1.5*h), Y) - max(y-int(h/2), 0)
9     template = np.indices((ww, hh))
10    template[0] += max(x-int(w/2), 0)
11    template[1] += max(y-int(h/2), 0)
12
13    # the center from which the distance will be counted
14    target = np.array([[x+int(w/2), y+int(h/2)]])
15
16    # Calculate the distance from the center to all points of
17    # interest.
18    d = distance.cdist(template.reshape(2, ww*hh).T, target, '
19        euclidean').reshape(ww, hh)
20
21    # we use the Gaussian distribution to transform the distance
22    std = 25
23    gaussian = (1/(std*((2*np.pi)**0.5)))*np.exp( -((d)**2)/(2*std*
24        std) )
25
26    # normalization
27    cv2.normalize(gaussian, gaussian, 0.15, 1, cv2.NORM_MINMAX)
28
29    # Creating weights for the density.
30    weights[max(x-int(w/2), 0):min(x+int(1.5*w), X), max(y-int(h/2),
31        0): min(y+int(1.5*h), Y)] = gaussian
32
33    # cv2.imshow('Weights', weights)
34
35    return weights

```

Listing 1: La création de poids pour la densité

Des expériences ont été menées dans lesquelles cette amélioration a été testée.

Sur les figures 12, 13, 14 et 15 vous pouvez voir les objets sélectionnés, et sur les séries d'images 16, 17, 18 et 19 vous pouvez suivre le fonctionnement

de l'algorithme Mean Shift avec des améliorations. Après les expériences, vous pouvez seulement noter une légère amélioration des résultats, mais l'algorithme perd souvent l'objet et après cela, il est très difficile de le trouver à nouveau.

Dans le cas de "*Anoine-Mug*", en raison de la spécificité de l'arrière-plan, l'histrogramm d'une partie particulière de l'image a une très grande convergence avec l'histogramme de l'objet sélectionné, donc très souvent l'algorithme se coince à cet endroit (Frmaes: 79, 89, 130).

L'algorithme a clairement amélioré le résultat dans le cas de "*VOT-Sunshade*" et peut mieux détecter l'homme à la fois à l'ombre et au soleil qu'il ne le fait dans la configuration standard.

Dans le cas de "*VOT-Car*", grâce à une légère différence dans la palette de couleurs de la voiture, du ciel et de la route , ainsi que de fortes fluctuations de la caméra, l'algorithme ne parvient pas du tout à suivre l'objet sélectionné.



Figure 12: Objet sélectionné sur la vidéo "*Antoine-Mug*". (Mean Shift plus la modification 1)



Figure 13: Objet sélectionné sur la vidéo "*VOT-Ball*". (Mean Shift plus la modification 1)

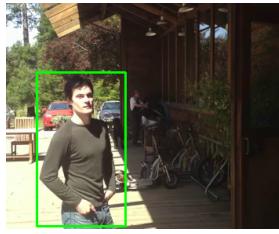


Figure 14: Objet sélectionné sur la vidéo "VOT-SunShade". (Mean Shift plus la modification 1)

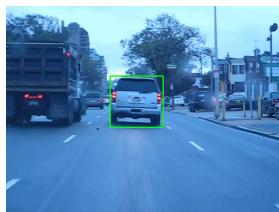


Figure 15: Objet sélectionné sur la vidéo "VOT-Car". (Mean Shift plus la modification 1)

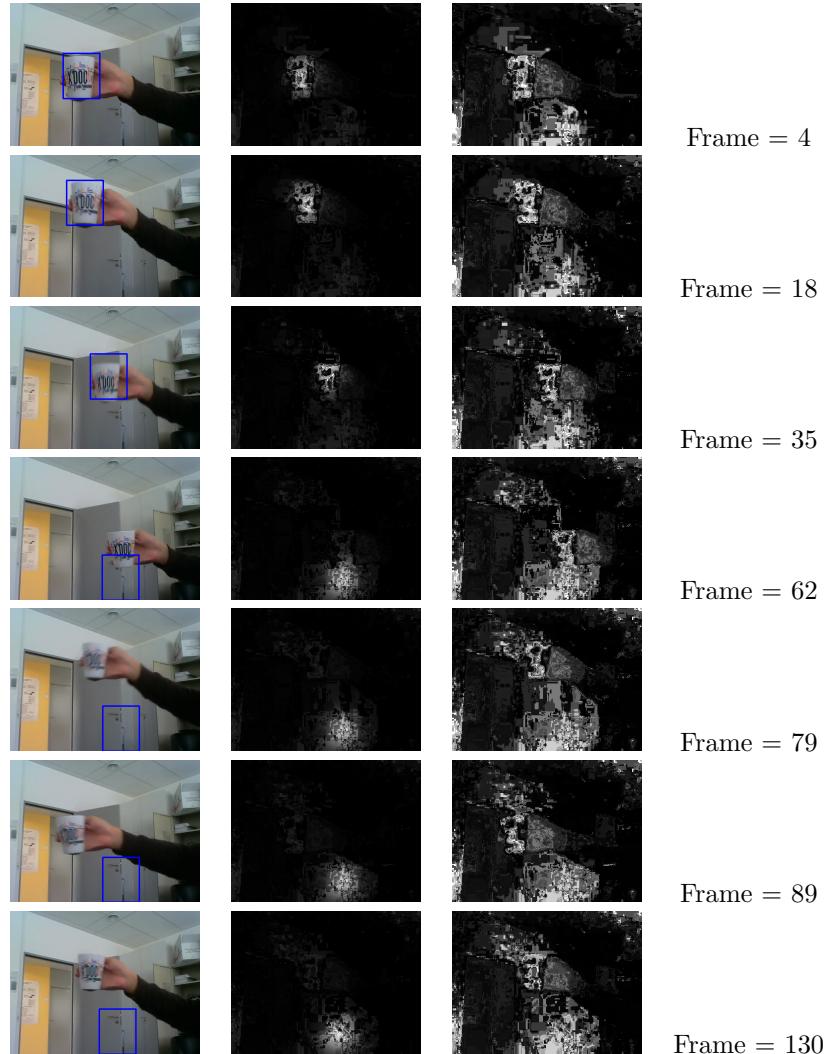


Figure 16: Série de cadres de vidéo *Antoine Mug* avec densité de distribution pondérée et originale. (Mean Shift plus la modification 1)

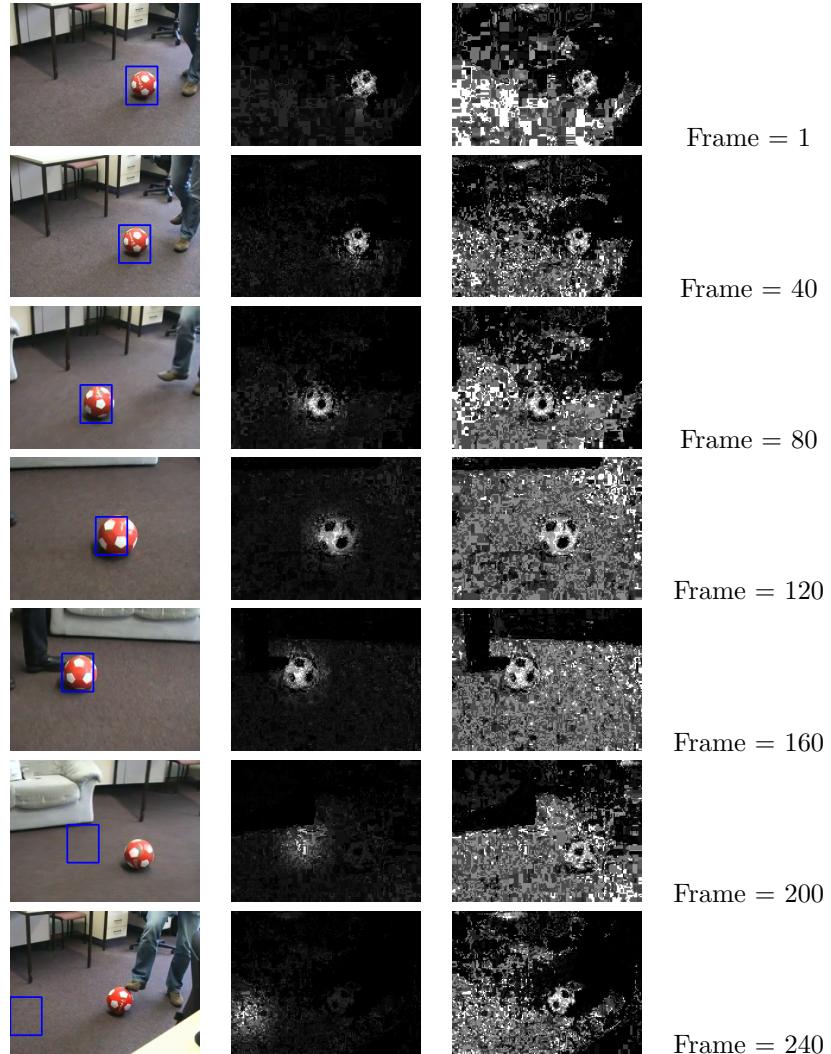


Figure 17: Série de cadres de vidéo "VOT-Ball" avec densité de distribution pondérée et originale. (Mean Shift plus la modification 1)

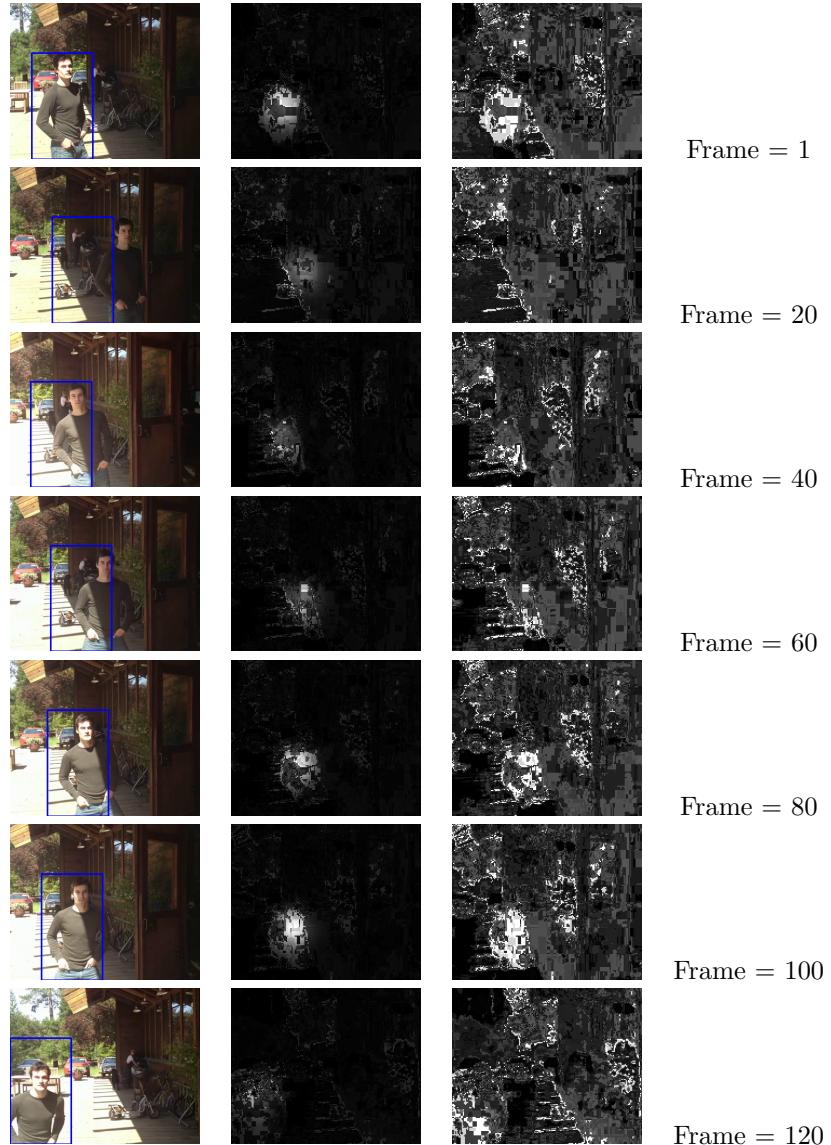


Figure 18: Série de cadres de vidéo "VOT-SunShade" avec densité de distribution pondérée et originale. (Mean Shift plus la modification 1)

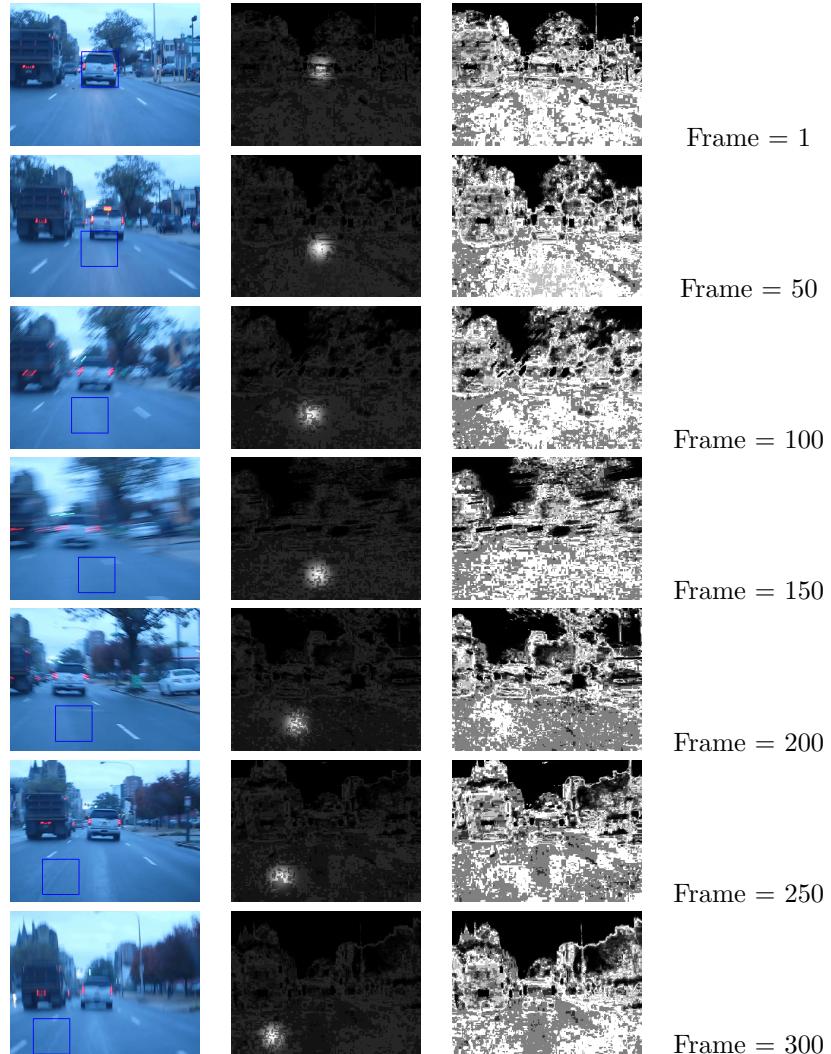


Figure 19: Série de cadres de vidéo "VOT-Car" avec densité de distribution pondérée et originale. (Mean Shift plus la modification 1)

1.2.2 Mise à jour de l'histogramme de modèle

Pour modifier l'histogramme du modèle, une stratégie simple basée sur des erreurs quadratiques moyennes (**MSE**) a été proposée. Si l'erreur dépasse le **seuil de 60000**, vous devez remplacer l'histogramme du modèle par l'histogramme du modèle dans le rectangle actuel (voir le listing 2).

```

1 model_difference = np.sum( (roi_hist - roi_hist_current)**2 )/w*h
2     print('Test {} : {}'.format(w*h, model_difference))
3

```

```

4
5     if model_difference > 60000 :
6         roi_hist = roi_hist_current

```

Listing 2: Mise à jour de l'histogramme modèle

Cette modification n'a pas amélioré le résultat de l'algorithme. À partir des cadres de différentes vidéos, il est clair qu'avec le changement de l'histogramme du modèle, la densité de distribution change également. Ce dernier change de sorte que l'objet sélectionné ne peut plus être trouvé sur lui.



Figure 20: Objet sélectionné sur la vidéo "Antoine-Mug". (Mean Shift plus la modification 1 et 2)



Figure 21: Objet sélectionné sur la vidéo "VOT-Ball". (Mean Shift plus la modification 1 et 2)

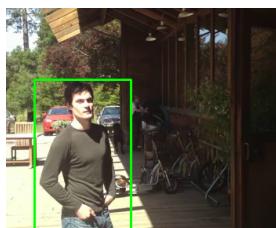


Figure 22: Objet sélectionné sur la vidéo "VOT-SunShade". (Mean Shift plus la modification 1 et 2)

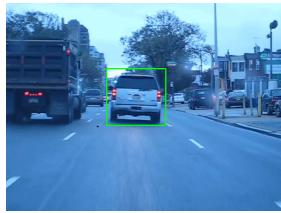


Figure 23: Objet sélectionné sur la vidéo "VOT-Car". (Mean Shift plus la modification 1 et 2)

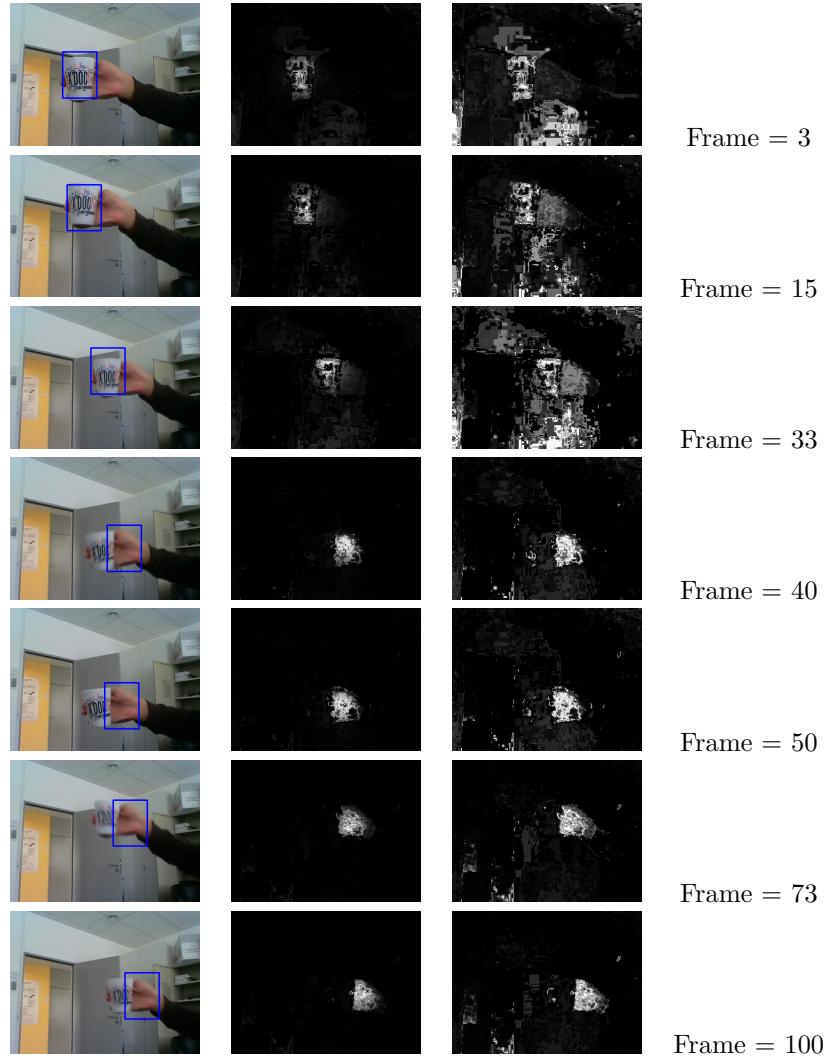


Figure 24: Série de cadres de vidéo "Antoine-Mug" avec densité de distribution pondérée et originale. (Mean Shift plus la modification 1 et 2 - rectangle bleu)

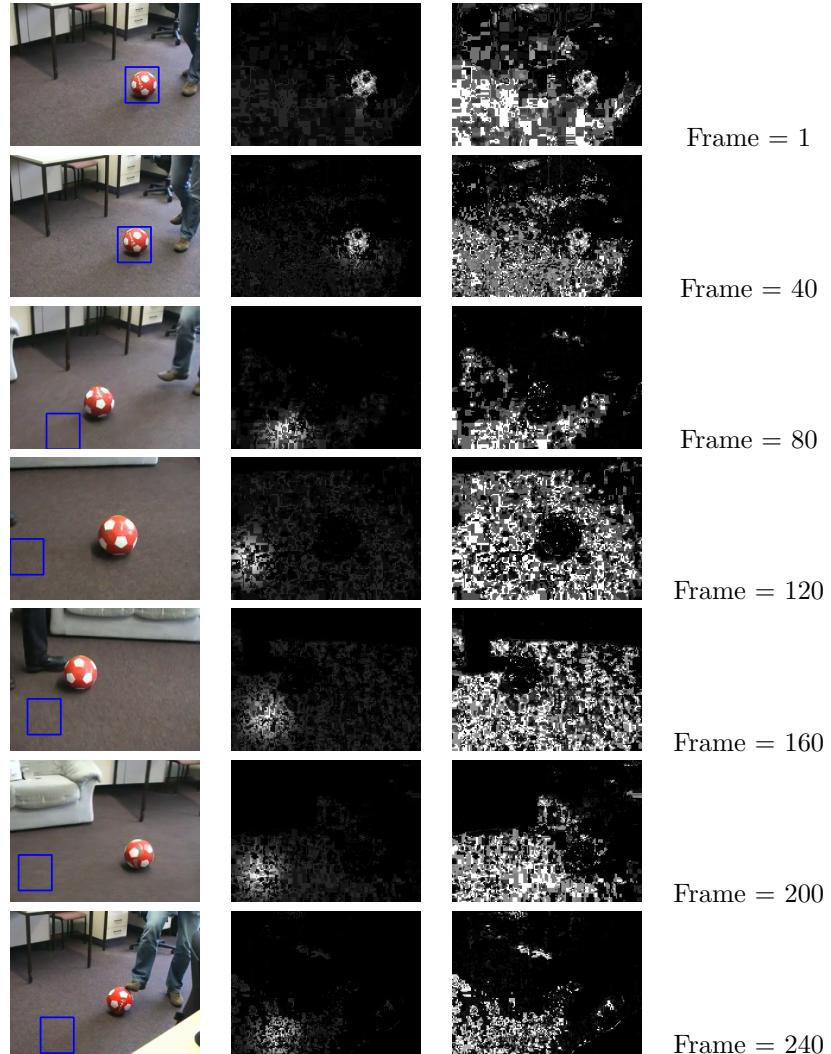


Figure 25: Série de cadres de vidéo "VOT-Ball" avec densité de distribution pondérée et originale. (Mean Shift plus la modification 1 et 2 - rectangle bleu)

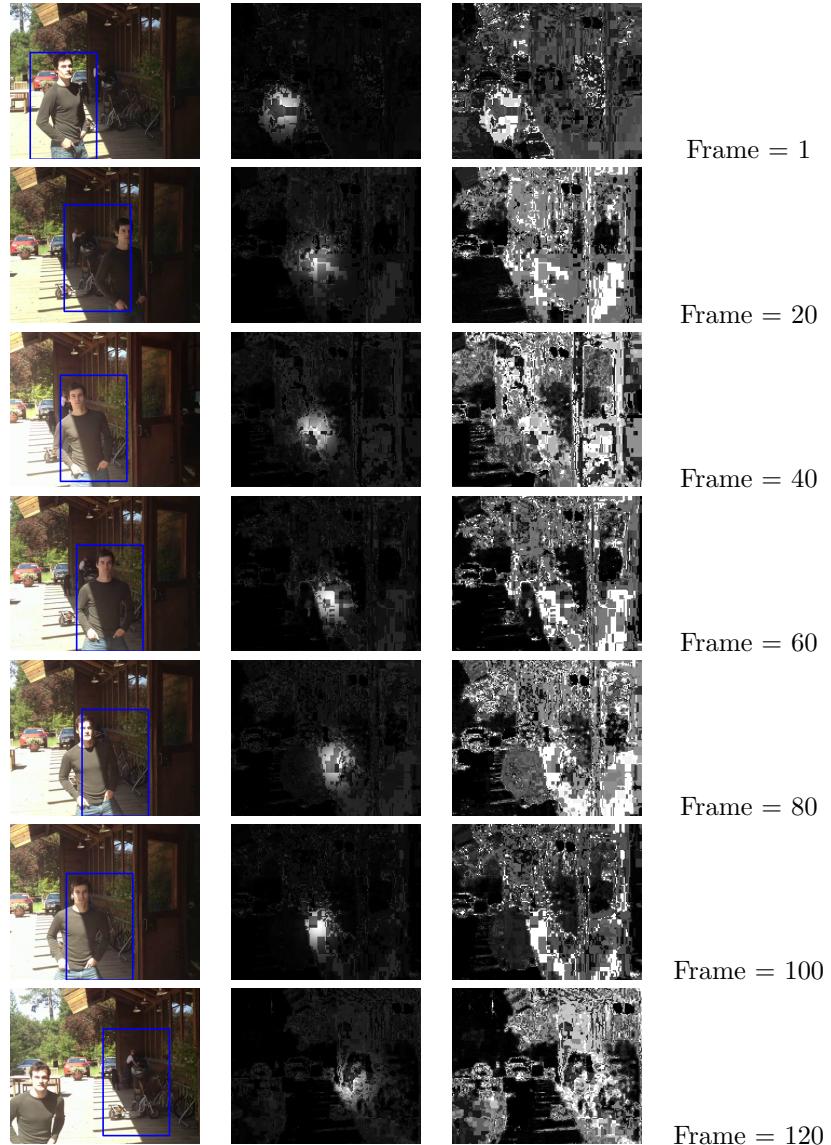


Figure 26: Série de cadres de vidéo "VOT-SunShade" avec densité de distribution pondérée et originale. (Mean Shift plus la modification 1 et 2 - rectangle bleu)

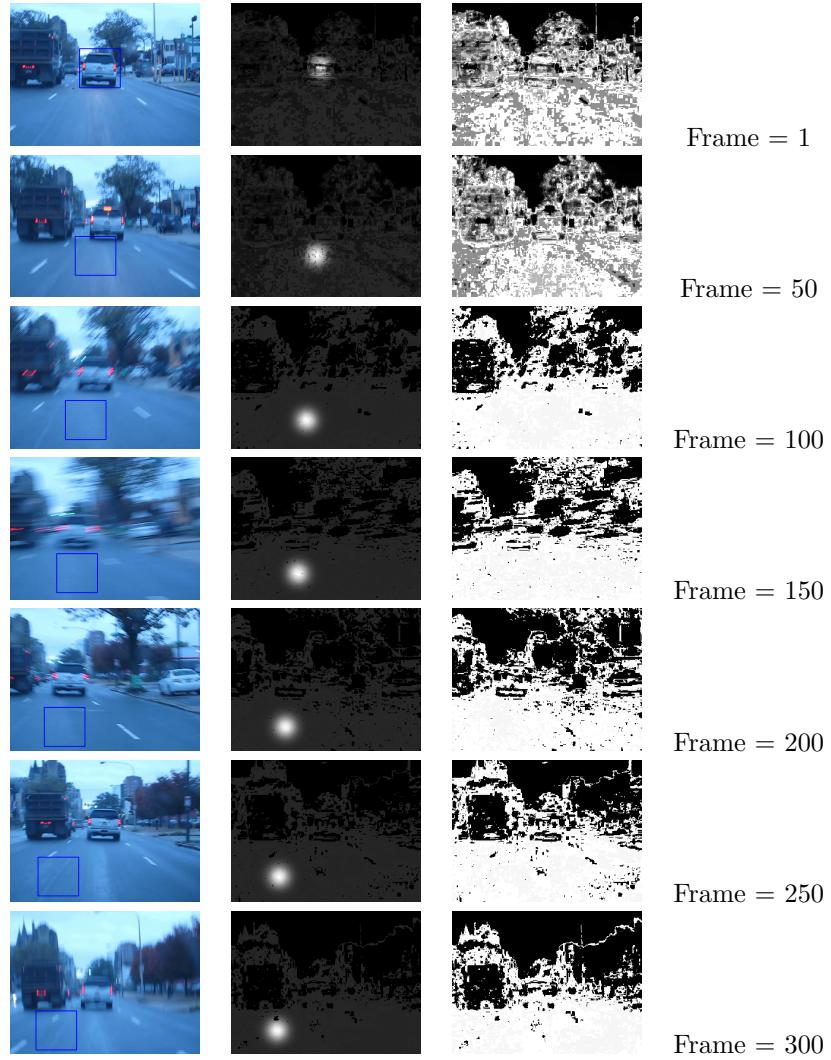


Figure 27: Série de cadres de vidéo "VOT-Car" avec densité de distribution pondérée et originale. (Mean Shift plus la modification 1 et 2 - rectangle bleu)

2 Transformée de Hough

2.1 Q3

Les fonctions suivantes ont été créées pour calculer l'orientation du gradient et créer un masque qui basé sur la valeur du module de gradient: *get_gradient_magnitude* et *get_gradient_orientation* (voir le listing 3).

```
1 def get_gradient_magnitude(frame):
```

```

2   dx = cv2.Sobel(frame, cv2.CV_64F, 1, 0, ksize=3)
3   dy = cv2.Sobel(frame, cv2.CV_64F, 0, 1, ksize=3)
4   # Compute the magnitude of the gradient
5   return np.hypot(dx,dy).astype('uint8')
6
7
8 def get_gradient_orientation(frame):
9     dx = cv2.Sobel(frame, cv2.CV_64F, 1, 0, ksize=3)
10    dy = cv2.Sobel(frame, cv2.CV_64F, 0, 1, ksize=3)
11    # Compute the orientation
12    return (np.arctan2(dy,dx) * 180 / np.pi)

```

Listing 3: L'oriertation et le module du gradient

Ensuite, un seuil de gradient a été trouvé pour chaque vidéo pour créer un masque (voir la figure 28). On aurait pu choisir un seuil, mais on aurait eu les pires résultats de l'algorithme de suivi des objets avec de transformée de Hough.

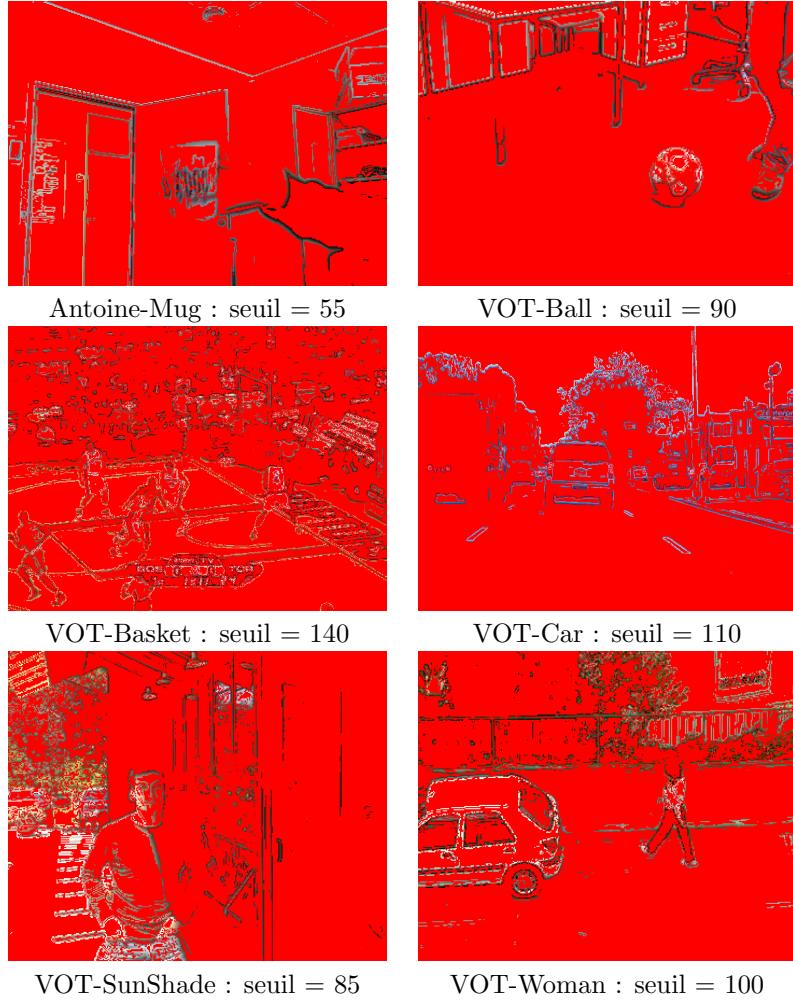


Figure 28: La séquence des orientations où les pixels masqués apparaissent en rouge

2.2 Q4

Deux fonctions ont été créées pour suivre les objets à l'aide de la transformation généralisée de Hough. Le *build_r_table* pour la construction de la R-table (construction d'un modèle d'objet), le *transform_hough* pour la procédure de vote (trouver une nouvelle position de l'objet dans l'image). Remarque: grâce à l'utilisation de la bibliothèque **numpy**, il a été possible de réduire le temps de calcul lors de l'utilisation de matrices.

```

1 def build_r_table(obj):
2     X,Y = obj.shape
3     gradient_magnitude = get_gradient_magnitude(obj)

```

```

4     _ , filtered = cv2.threshold(gradient_magnitude, seuil, 255, cv2.
5         THRESH_BINARY)
6     # cv2.imshow('r_table', filtered)
7     orientation = get_gradient_orientation(filtered)
8     orientation[filtered == 0] = -255
9     unique_orientation = np.unique(orientation)
10
11    r_table = dict()
12    center = np.array([[int(X/2) ,int(Y/2)]])
13
14    for teta in unique_orientation:
15        if teta == -255:
16            continue
17        r_table[teta] = center - np.argwhere(orientation == teta)
18
19    return r_table
20
21 def transform_hough(image, r_table, x,y,w,h):
22
23    X, Y = image.shape
24    gradient_magnitude = get_gradient_magnitude(image)
25    _ , filtered = cv2.threshold(gradient_magnitude, seuil, 255, cv2.
26        THRESH_BINARY)
27    # cv2.imshow('get_gradient_magnitude', maskmask)
28    orientation = get_gradient_orientation(filtered)
29    orientation[filtered == 0] = -255
30
31    vote = np.zeros(image.shape)
32
33    for teta in r_table:
34        tmp = np.argwhere(orientation == teta)
35        if tmp.shape[0] == 0 :
36            continue
37        for r in r_table[teta]:
38
39            ind_for_vote = tmp + r
40            ind_for_vote = ind_for_vote[ (ind_for_vote[:,0] < X) & (
41                ind_for_vote[:,0] > 0) &(ind_for_vote[:,1] < Y) & (ind_for_vote
42                [:,1] > 0) ]
43            vote[ind_for_vote[:,0], ind_for_vote[:,1]] += 1
44
45    # extra points for matches in the current rectangle
46    vote[max(x-w, 0):min(x+2*w, X), max(y - h, 0):min(y+2*h, Y)] +=
47        200
48
49    centers = np.argwhere(vote == np.amax(vote))
50    center = centers.mean(axis = 0).astype('int')
51    # center = centers[0]
52    return center[0], center[1]

```

Listing 4: La construction de R-Table et la transformée de Hough

Sur les figures 29, 30, 31, 32, 33, 34, 35 et 36 vous pouvez voir les objets sélectionnés et les modèles pour R-Table, et sur les séries d'images 37, 38, 39 et 40 vous pouvez suivre le fonctionnement de l'algorithme Hough.

Après les expériences, vous pouvez noter une amélioration des résultats

par rapport à l'algorithme Mean Shift mais la Transformée de Hough est plus coûteux en termes de calcul par rapport à l'algorithme Mean Shift.

L'algorithme ne détecte pas toujours correctement les objets, bien qu'en cas de perte, il puisse facilement les trouver.

La version de base de la Transformée de Hough est très sensible à la qualité des images dans la vidéo (nombre d'images par seconde). Si l'image est floue, l'algorithme ne peut pas trouver correctement l'objet, ce qui est visible sur les figures 37 (frames: 96), 39 (frames: 100, 150 ou 300).

En outre, la version de base de la Transformée de Hough est sensible aux rotations des objets sélectionnés, ce qui est visible sur les figures 38 (frames: 80, 160), cependant l'algorithme Mean Shift détecte correctement l'objet.

En outre, la version de base de la Transformée de Hough est sensible à la transformation des objets sélectionnés (mise à l'échelle, mouvements humains), ce qui est visible sur les figures 40.

Malgré les inconvénients de la transformation de Hough, l'algorithme a montré un assez bon résultat et fait un bon travail avec le suivi des objets. Il est à noter que le seuil du masque de dégradé de la section précédente a été appliqué à chaque vidéo.



Figure 29: Objet sélectionné sur la Figure 30: Le modèle pour construire vidéo "Antoine-Mug".(Transformée de une R-table (La vidéo "Antoine-Mug" - Transformée de Hough)

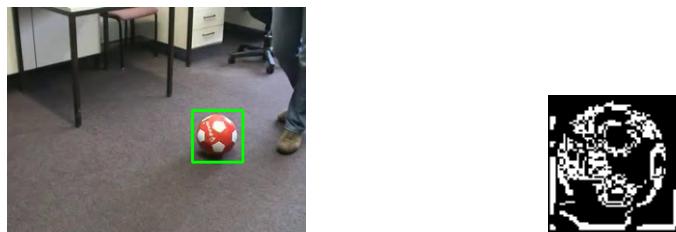


Figure 31: Objet sélectionné sur Figure 32: Le modèle pour construire la vidéo "VOT-Ball".(Transformée de une R-table (La vidéo "VOT-Ball" - Transformée de Hough)

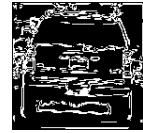


Figure 33: Objet sélectionné sur Figure 34: Le modèle pour construire la vidéo "VOT-Car".(Transformée de une R-table (La vidéo "VOT-Car" - Hough) Transformée de Hough)



Figure 35: Objet sélectionné sur la Figure 36: Le modèle pour construire une R-table (La vidéo "VOT-Woman" - Transformée de Hough) Transformée de Hough)

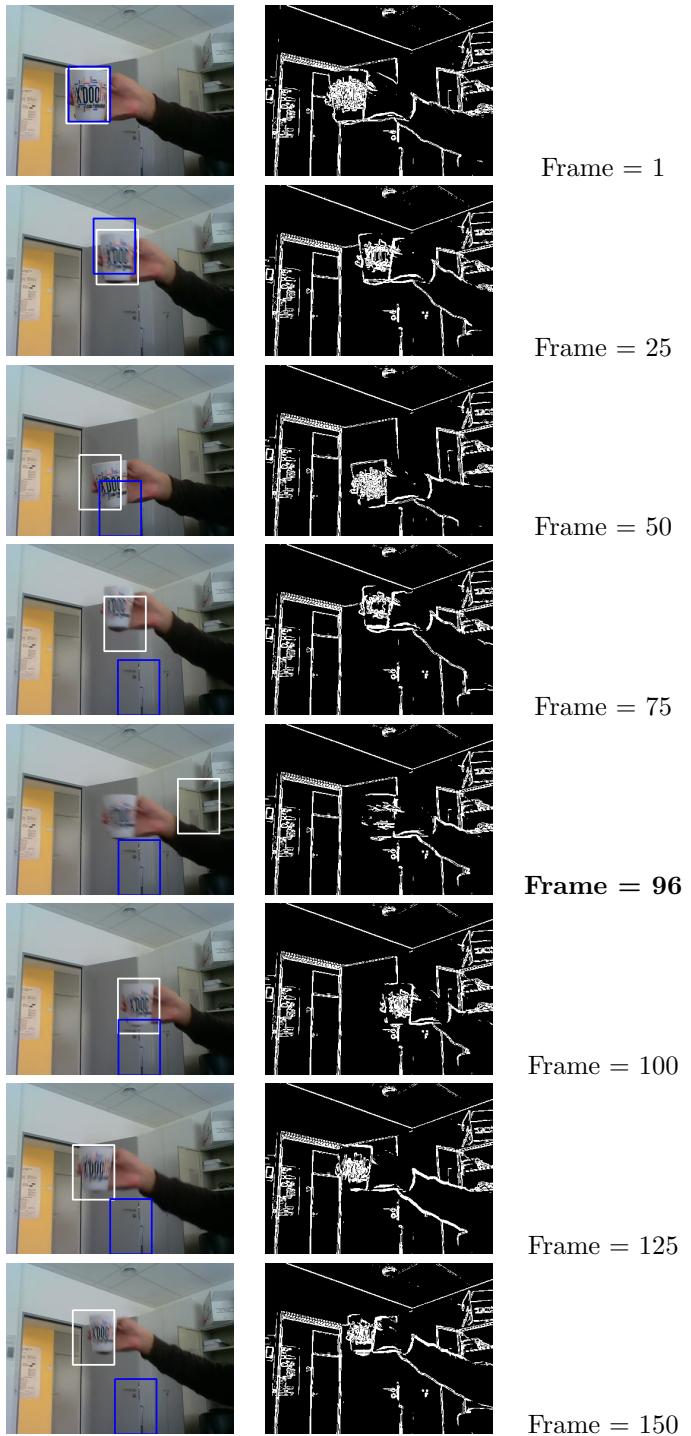


Figure 37: Série de cadres de vidéo "Antoine-Mug" avec un masque de gradient. (Transformée de Hough - rectangle blanc)

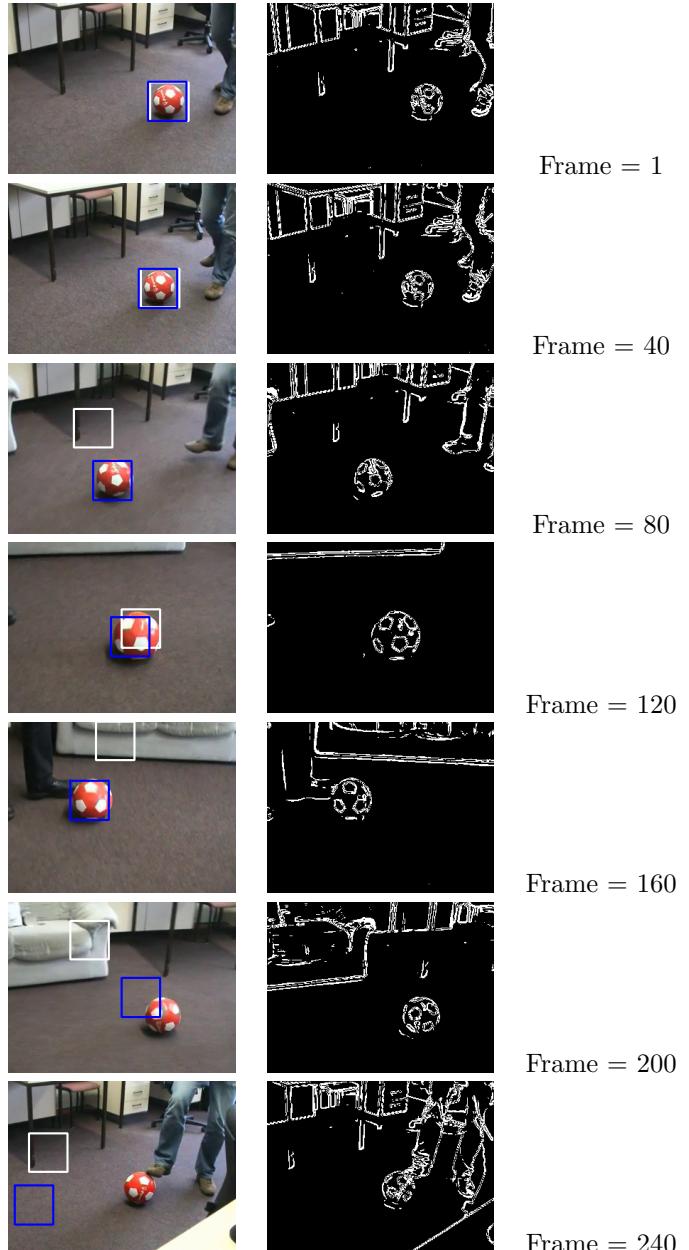


Figure 38: Série de cadres de vidéo "VOT-Ball" avec un masque de gradient. (Transformée de Hough - rectangle blanc)

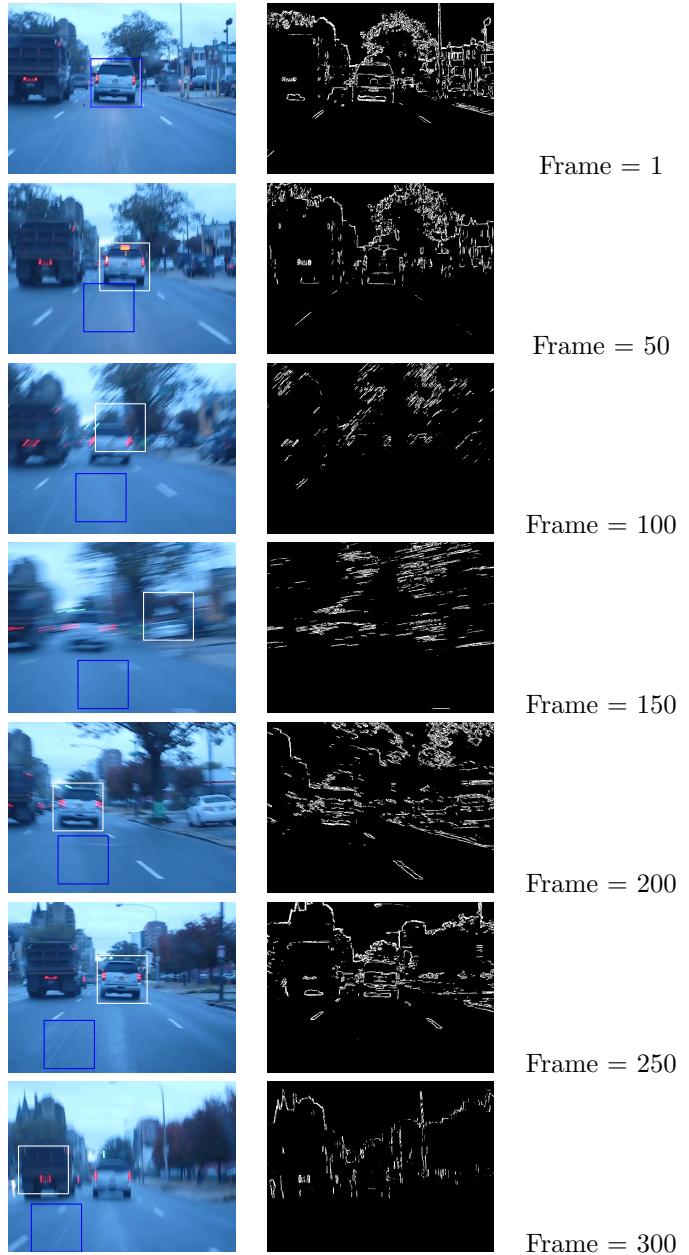


Figure 39: Série de cadres de vidéo "VOT-Car" avec un masque de gradient. (Transformée de Hough - rectangle blanc)



Figure 40: Série de cadres de vidéo "VOT-Woman" avec avec un masque de gradient. (Transformée de Hough - rectangle blanc)

2.3 Q5

On a remplacé le calcul du maximum par l'application du Mean Shift sur la transformée de Hough.

Le résultat est dans une certaine mesure améliorée, maintenant l'algorithme même si ne peut pas trouver correctement l'objet ne met pas en évidence les zones de l'image qui sont très loin de l'objet (Voir les résultats sur les figures 41, 42 et 43).



Figure 41: Objet sélectionné sur la Figure 42: Le modèle pour construire une R-table (La vidéo "Antoine-Mug" - Transformée de Hough + Mean Shift)

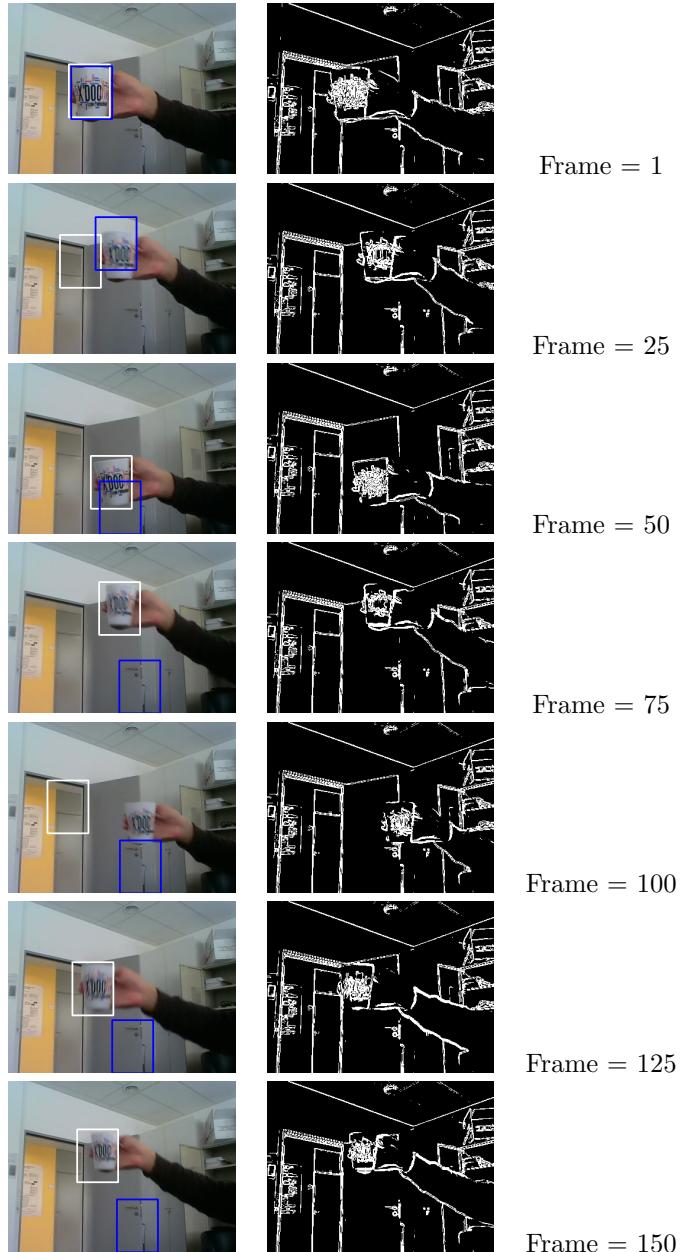


Figure 43: Série de cadres de vidéo "Antoine-Mug" avec avec un masque de gradient. (Transformée de Hough + Mean Shift - rectangle blanc; Mean Shift standart - rectangle bleu)

Vous pouvez utiliser la transformation suivante pour suivre la déformation

d'un objet (échelle et rotation):

$$x_c = x + s_x(x' \cos(\theta) + y' \sin(\theta))$$

$$y_c = y + s_y(-x' \sin(\theta) + y' \cos(\theta))$$

où s_x et s_y sont les coefficients d'échelle, θ est l'angle de rotation, x', y' - les valeurs sont enregistrées dans le R table.

Ensuite, pour chaque combinaison d'angle et d'échelle, vous devez trouver le nombre de voix. Combinaison des coefficients avec le plus grand nombre de votes détermine la forme actuelle de l'objet. Après cela, vous devez recalculer le R Table en fonction des nouvelles dimensions de l'objet.

Cependant, trouver la bonne combinaison les coefficients est difficile à calculer, ce qui ralentit l'algorithme.

Malheureusement, nous n'avons pas réussi à mettre en œuvre ces transformations. Nous avons seulement effectué la transformation de zoom (voir les figures 44, 45 et 46). À partir du cadre 50, on voit que le rectangle blanc change de taille en fonction des dimensions de la tasse.



Figure 44: Objet sélectionné sur la Figure 45: Le modèle pour construire vidéo "Antoine-Mug".(Transformée de une R-table (La vidéo "Antoine-Mug" - Transformée de Hough + la transformation)

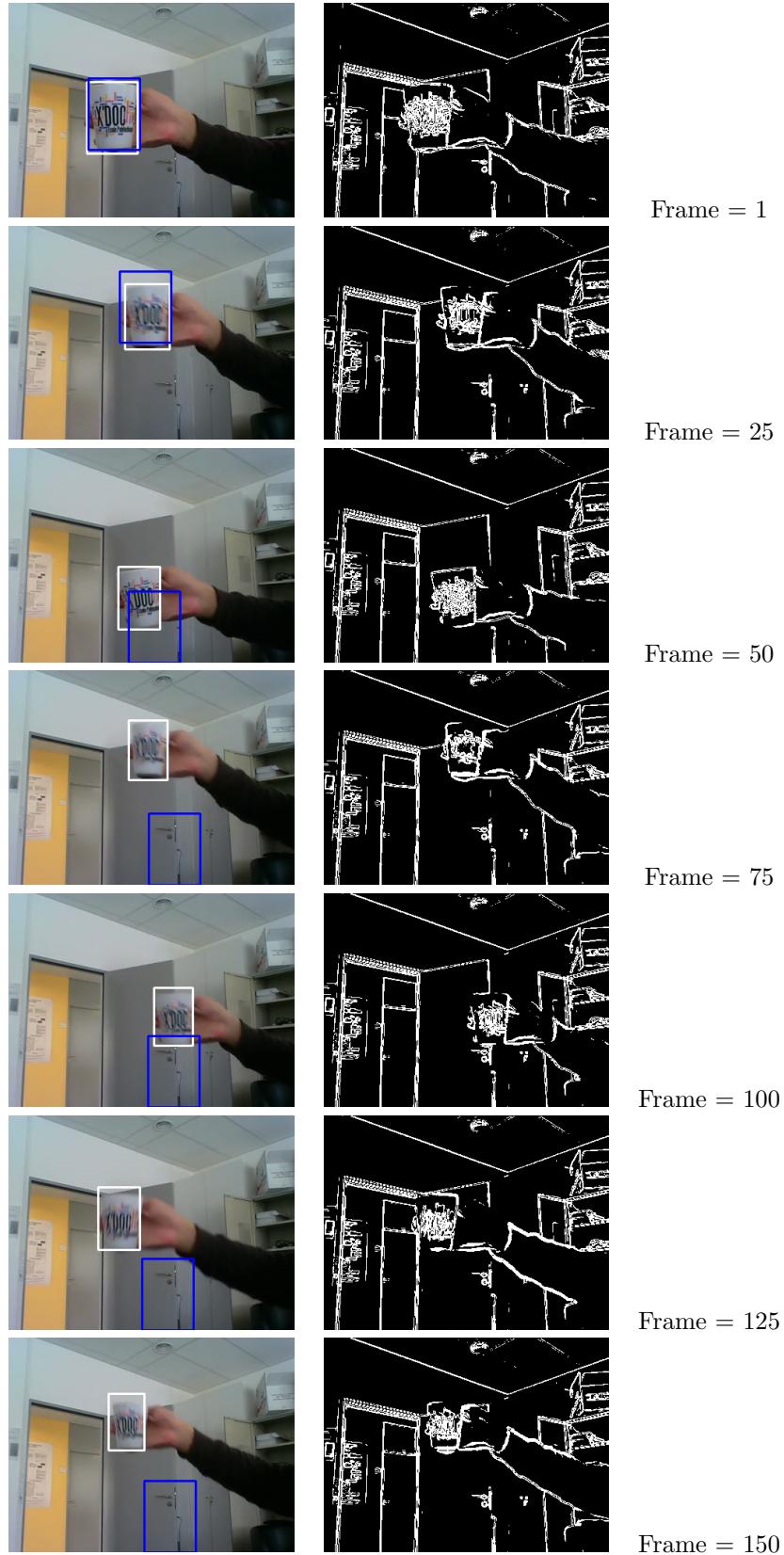


Figure 46: Série de cadres de vidéo "Antoine-Mug" avec un masque de gradient. (Transformée de Hough + la transformation - rectangle blanc; Mean Shift standart - rectangle bleu)

3 Conclusion

Ainsi, dans ce travail, deux algorithmes (Mean Shift et Transformée de Hough) ont été examinés pour l'analyse vidéo et le suivi des objets. Au cours des expériences, les avantages et les inconvénients des deux approches ont été identifiés. Le principal avantage de Mean Shift est sa vitesse de fonctionnement. Le principal avantage de Transformée de Hough est sa précision et sa capacité à prendre en compte la déformation des objets. Pas le dernier rôle a la qualité de la vidéo (nombre d'images par seconde). Grâce à l'utilisation de la bibliothèque numpy on a réussi à accélérer l'algorithme Transformée de Hough. La possibilité de combiner les deux approches a été examinée.