



UNIVERSITÉ NATIONALE DU VIETNAM  
INSTITUT FRANCOPHONE INTERNATIONAL

*Promotion : 24*  
Option : SIM (systèmes Intelligents et multimédias)  
Vision par ordinateur

---

**Segmentation des objets en mouvement en  
utilisant le flot optique**

---

*Rédigé par :*  
**MBIAYA KWUITE Franck Anael**  
**KANA NGUIMFACK Kevin**

*Enseignant :*

**Dr. Oanh Nguyen Thi**

*Année Académique 2020/2021*

# Sommaire

1	Introduction . . . . .	3
2	Estimation du flot optique dans une séquence d'images . . . . .	3
2.1	Calcul du flot optique . . . . .	4
2.2	Visualisation de la norme de chaque pixel . . . . .	4
2.3	Visualisation de la norme et de l'orientation de chaque pixel . . . . .	5
3	Segmentation des objets en mouvement . . . . .	6
3.1	Segmentation des objets se déplacent le plus vite . . . . .	6
	Utilisation du seuillage . . . . .	7
	Utilisation de Kmeans . . . . .	9
	Comparaison du seuillage et de Kmeans . . . . .	12
3.2	Segmentation des objets à différentes vitesses . . . . .	12
	Utilisation du seuillage . . . . .	12
	Utilisation de Kmeans . . . . .	15
3.3	Segmentation des objets en fonction de la norme et de l'orientation . . . . .	18
4	Conclusion . . . . .	21

# Table des figures

1	Visualisation de la norme de chaque pixel . . . . .	5
2	Visualisation de la norme et l'orientation de chaque pixel . . . . .	6
3	Seuillage : Seuil = 40, Kernel = 7x7 . . . . .	7
4	Seuillage : Seuil = 60, Kernel = 7x7 . . . . .	8
5	Seuillage : Seuil = 80, Kernel = 7x7 . . . . .	8
6	Seuillage : Seuil = 80, Kernel = 11x11 . . . . .	9
7	Kmeans : K = 2, Kernel = 7x7 . . . . .	10
8	Kmeans : K = 3, Kernel = 7x7 . . . . .	10
9	Kmeans : K = 3, Kernel = 11x11 . . . . .	11
10	Kmeans : K = 5, Kernel = 7x7 . . . . .	11
11	Seuillage : Seuil1 = 50, Seuil2 = 70, Kernel = 7x7 . . . . .	13
12	Seuillage : Seuil1 = 50, Seuil2 = 130, Kernel = 7x7 . . . . .	13
13	Seuillage : Seuil1 = 70, Seuil2 = 150, Kernel = 7x7 . . . . .	14
14	Seuillage : Seuil1 = 70, Seuil2 = 150, Kernel = 11x11 . . . . .	14
15	Seuillage : Seuil1 = 100, Seuil2 = 150, Kernel = 7x7 . . . . .	15
16	Kmeans2 : K = 3, Kernel = 7x7 . . . . .	16
17	Kmeans2 : K = 3, Kernel = 11x11 . . . . .	17
18	Kmeans2 : K = 4, Kernel = 7x7 . . . . .	17
19	Kmeans2 : K = 5, Kernel = 7x7 . . . . .	18
20	Pietons : K = 2, Kernel = 7x7 . . . . .	19
21	Trafic : K = 2, Kernel = 7x7 . . . . .	20
22	Venise : K = 3, Kernel = 7x7 . . . . .	20

## 1 Introduction

La détection d'objets mobiles dans des flux vidéo est une étape essentielle pour de nombreux algorithmes de vision par ordinateur. Cette tâche se complexifie lorsque la caméra utilisée est en mouvement. En effet, l'environnement capté par ce type de caméra apparaît en mouvement et il devient plus difficile de distinguer les objets qui effectuent réellement un mouvement de ceux qui constituent la partie statique de la scène.

Dans le travail proposé dans ce rapport, nous proposons une méthode pour segmenter les objets mobiles dans un flux vidéo. Pour faire cela, nous utilisons le flot optique. Ce travail est disponible sur [github](#).

Dans le cadre de ce travail, nous avons dans un premier temps estimer le flot optique d'une séquence d'image (vidéo). Ensuite, nous avons fait la segmentation d'objets qui se déplacent en utilisant le flot optique. Nous avons utilisé le seuillage et Kmeans pour effectuer la segmentation.

## 2 Estimation du flot optique dans une séquence d'images

Le flot optique est un champ de déplacement visuel qui permet d'expliquer des variations dans une image animée en terme de déplacement de points images. Il a de nombreuses applications dans des domaines tels que : la segmentation des images en mouvement, la compression de vidéo, la stabilisation vidéo, etc.

Le flot optique fonctionne selon 2 hypothèses :

- Les intensités de pixels d'un objet ne changent pas entre les images consécutives
- Les pixels voisins ont un mouvement similaire

Les méthodes de calcul du flot optique tentent de calculer le mouvement entre deux images qui sont pris aux instants  $t$  et  $t + \Delta(t)$  à chaque position de pixels. Ces méthodes sont appelées différentielles car elles sont basées sur des approximations de séries de Taylor locales du signal d'image, c'est-à-dire qu'elles utilisent des dérivées partielles par rapport aux coordonnées spatiales et temporelles.

Si on considère un pixel  $I(x, i, t)$  dans l'image à l'instant  $t$ , il se déplace à distance  $(\Delta x, \Delta y, \Delta t)$  dans l'image suivante à l'instant  $t + 1$ . Etant donné que ces 2 pixels sont les mêmes et que l'intensité ne change pas, nous pouvons donc écrire :

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t)$$

En supposant que le mouvement soit petit, la contrainte d'image au pixel  $I(x, y, t)$  avec la série de Taylor permet d'obtenir :

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{\partial I}{\partial x} \Delta x + \frac{\partial I}{\partial y} \Delta y + \frac{\partial I}{\partial t} \Delta t$$

En effectuant une linéarisation, on obtient :

$$\frac{\partial I}{\partial x} \Delta x + \frac{\partial I}{\partial y} \Delta y + \frac{\partial I}{\partial t} \Delta t = 0$$

En divisant chaque terme par  $\Delta t$ , on obtient :

$$\frac{\partial I}{\partial x} \frac{\Delta x}{\Delta t} + \frac{\partial I}{\partial y} \frac{\Delta y}{\Delta t} + \frac{\partial I}{\partial t} \frac{\Delta t}{\Delta t} = 0$$

Ce qui équivaut à :

$$\frac{\partial I}{\partial x} Vx + \frac{\partial I}{\partial y} Vy + \frac{\partial I}{\partial t} = 0$$

où  $Vx$  et  $Vy$  sont les composantes  $x$  et  $y$  du flot optique de  $I(x, y, t)$ ,

$\frac{\partial I}{\partial x} = Ix$ ,  $\frac{\partial I}{\partial y} = Iy$ ,  $\frac{\partial I}{\partial t} = It$  sont les dérivées de l'image à  $(x, y, t)$  dans les directions correspondantes,  
d'où

$$IxVx + IyVy = -It$$

Il s'agit d'une équation à deux inconnues et ne peut être résolue en tant que telle. Pour trouver le flot optique, un autre ensemble d'équations est nécessaire, donné par une contrainte supplémentaire. Pour la suite de notre travail, nous allons utiliser la méthode de Gunnar Farneback.

## 2.1 Calcul du flot optique

Pour calculer le flow optique, nous avons utilisé la fonction **calcOpticalFlowFarneback** disponible dans la librairie OpenCV. Cette fonction utilise les pixels à l'instant  $t$  et à l'instant  $t + 1$  et retourne un tableau contenant les vecteurs de flot optique ( $Vx, Vy$ ) pour chaque pixels.

A partir des vecteurs de flot optique ( $Vx, Vy$ ), nous calculons la norme et l'orientation du déplacement de chaque pixel en utilisant la fonction **cartToPolar** disponible dans OpenCV.

## 2.2 Visualisation de la norme de chaque pixel

Pour visualiser les résultats de la norme de chaque pixel, nous créons une image dans l'espace de couleur TSV ayant les mêmes dimensions que l'image dont nous voulons estimer le flot optique. Nous mettons les channels de couleur (T et S) de cette image à 0 et nous remplissons le channel de luminosité (V) avec la valeur de les normes précédemment calculées avec la fonction **cartToPolar**. Nous convertissons les valeurs de V pour qu'elles se situent dans l'intervalle [0, 255].

Ainsi, plus un pixel est lumineux, plus la valeur de la norme est élevée. Nous obtenons ainsi les résultats suivants :



FIGURE 1 – Visualisation de la norme de chaque pixel

### 2.3 Visualisation de la norme et de l'orientation de chaque pixel

Pour visualiser les résultats de la norme et de l'orientation de chaque pixel, nous créons une image dans l'espace de couleur TSV ayant les mêmes dimensions que l'image dont nous calculons le flot optique. Nous mettons le channel de couleur (S) de cette image à 255. Nous remplissons le channel de couleur (T) de cette image avec les valeurs de l'orientation précédemment calculées que nous convertissons en degré. Nous remplissons le channel de luminosité (V) avec les valeurs de la norme précédemment calculées. Nous convertissons les valeurs des channels T et V pour qu'elles se situent dans l'intervalle [0, 255]

Ainsi, plus un pixel est lumineux, plus la valeur de la norme est élevée. L'orientation de chaque pixel correspond à la valeur de teinte de celui-ci. Nous obtenons ainsi les résultats suivants :



FIGURE 2 – Visualisation de la norme et l'orientation de chaque pixel

### 3 Segmentation des objets en mouvement

Dans une video, si des objets déplacent à différentes vitesses, les normes du flot optique sont aussi différentes. S'ils se déplacent dans des directions différentes, les flots optiques de ces objets ont des orientations différentes. Donc, le flot optique (la norme ou l'angle ou les deux) peut aider à segmenter des objets en mouvement en appliquant une méthode de segmentation.

Dans la suite de notre travail, nous allons utiliser la méthode de seuillage et la méthode k-means pour segmenter les images et visualiser les objets en mouvement.

#### 3.1 Segmentation des objets se déplacent le plus vite

L'objectif ici est de visualiser les objets de l'image qui se déplacent le plus vite. Pour cela, nous allons utiliser la norme du flot optique de chaque pixel. Nous allons rechercher les pixels ayant la norme la plus élevée (Supérieur à un seuil) et nous allons faire la segmentation par rapport à ces pixels.

Pour améliorer les résultats que nous avons obtenus, nous allons appliquer les opérations morphologiques (ouverture, puis fermeture) sur l'image binaire obtenue après la segmentation. Nous allons également appliquer plusieurs tailles de noyau afin de visualiser les résultats.

## Utilisation du seuillage

Nous avons utilisé plusieurs valeurs de seuil (entre 0 et 255) afin de segmenter l'image et visualiser les objets qui se déplacent le plus vite. Nous avons également utilisé plusieurs noyaux pour les opérations morphologiques sur les images binaires.

Nos résultats sont représentés en 4 images :

- La première image à gauche est l'image originale
- La première image à droite est le résultat obtenu après la segmentation
- La deuxième image à droite est l'image binaire obtenue après le filtrage
- La deuxième image à gauche est l'image originale contenant les objets segmentés



FIGURE 3 – Seuillage : Seuil = 40, Kernel = 7x7



FIGURE 4 – Seuillage : Seuil = 60, Kernel = 7x7



FIGURE 5 – Seuillage : Seuil = 80, Kernel = 7x7



FIGURE 6 – Seuillage : Seuil = 80, Kernel = 11x11

Au regard des différents résultats, nous constatons que le choix du seuil et celui de la taille du noyau sont très importants. Dans le premier résultat (figure 3), les objets qui se déplacent le plus rapidement dans la vidéo sont bien détecté. Cependant, nous remarquons des erreurs de segmentation qui peuvent être dues à la mobilité de la caméra utilisée. Nous remarquons que avec un seuil de 60 (figure 4), les résultats sont bien meilleurs et nous n'avons presque pas d'erreurs de segmentation. Avec un seuil de 80 (figure 5), nous remarquons que la taille des objets segmentés diminue, ce qui signifie que les pixels qui se déplacent plus rapidement sont sélectionnés pendant la segmentation.

Comme mentionné précédemment, le choix de la taille du noyau des opérations morphologiques est tout aussi important. Nous pouvons voir que avec un seuil de 80, nous avons un meilleur résultat de segmentation avec le noyau  $7 \times 7$  (figure 5). En effet, le noyau  $11 \times 11$  (figure 6) a tendance à rogner les zones segmentées et ainsi dégrader les résultats.

### Utilisation de Kmeans

Nous avons utilisé plusieurs valeurs de K afin de faire un clustering sur les valeurs de la norme. Après le clustering, nous avons utilisé les pixels appartenant au cluster ayant la valeur de centre la plus élevée. Nous avons également utilisé plusieurs noyaux pour les opérations morphologiques sur les images binaires.

Nos résultats sont représentés en 4 images :

- La première image à gauche est l'image originale
- La première image à droite est le résultat obtenu après la segmentation
- La deuxième image à droite est l'image binaire obtenue après le filtrage
- La deuxième image à gauche est l'image originale contenant les objets segmentés



FIGURE 7 – Kmeans : K = 2, Kernel = 7x7

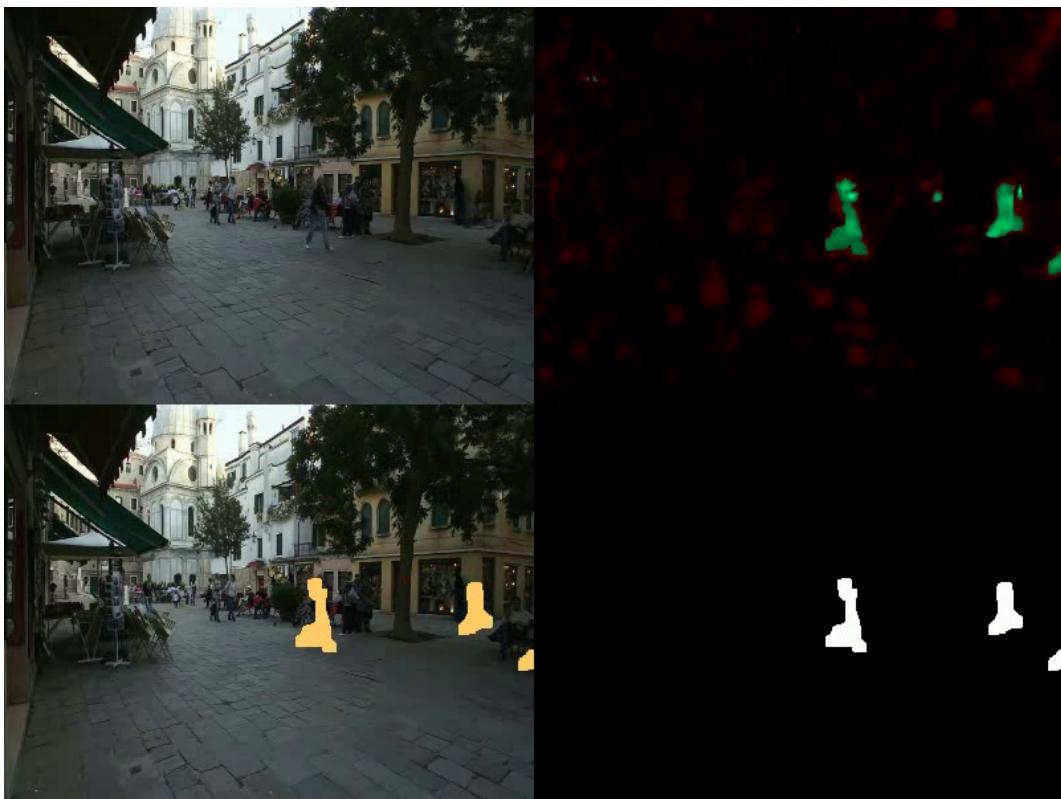


FIGURE 8 – Kmeans : K = 3, Kernel = 7x7



FIGURE 9 – Kmeans : K = 3, Kernel = 11x11



FIGURE 10 – Kmeans : K = 5, Kernel = 7x7

Nous pouvons remarquer que la valeur de K a une influence importante pour la segmentation. En effet, nous remarquons que nous avons un meilleur résultat pour la valeur de K = 2 et K = 3. Avec la valeur K = 2 (figure 7), nous pouvons remarquer que nous avons beaucoup plus de pixels sélectionnés

pendant la segmentation, alors que avec la valeur de  $K = 3$  (figure 8), le résultat est beaucoup plus fin. Cependant, avec la valeur de  $K = 5$  (figure 10), nous remarquons que le fait d'avoir trop de classe diminue le nombre de pixels sélectionnés pour les plus grandes valeurs de la norme. Le résultat est alors moins bon.

Nous pouvons également voir que la valeur du noyau utilisé pour les opérations morphologiques a une grande importance pendant la segmentation. En effet, nous remarquons que avec le noyau  $7 \times 7$  (figure 8), nous avons de meilleurs résultats qu'avec le noyau  $11 \times 11$  (figure 9). Ceci est du au fait que un noyau de grande taille a tendance à rogner beaucoup plus l'image qu'un noyau de petite taille.

### Comparaison du seuillage et de Kmeans

Nous remarquons de façon générale que nous obtenons de meilleurs résultats avec la méthode Kmeans. En effet, Kmeans regroupe de façon plus efficace les pixels qui ont des valeurs de norme très proches que le seuillage.

## 3.2 Segmentation des objets à différentes vitesses

L'objectif ici est de visualiser les objets de l'images qui se déplacent en fonction de leur vitesse de déplacement. Pour cela, nous allons utiliser la norme du flot optique de chaque pixel. Nous allons rechercher les pixels ayant la norme appartenant à un intervalle et nous allons faire la segmentation par rapport à ces pixels.

Pour améliorer les résultats que nous avons obtenus, nous allons appliquer les opérations morphologiques (ouverture, puis fermeture) sur l'image binaire obtenue après la segmentation. Nous allons appliquer plusieurs taille de noyau afin de visualiser les résultats.

A la différence des travaux précédent, nous avons travailler ici dans l'espace de couleur  $La^*b^*$  afin de visualiser les résultats.

### Utilisation du seuillage

Nous avons utilisé deux valeurs de seuil (Seuil\_1 et Seuil\_2) afin de visualiser les objets qui se déplacent très rapidement et ceux qui se déplacent moyennement rapidement. Nous avons fait varier ces valeurs de seuil afin d'observer leurs influences sur les résultats. Nous avons également utilisé plusieurs valeurs du noyau pour les opérations morphologiques sur les images binaires.

Nos résultats sont représentés en 4 images :

- La première image à gauche est l'image originale
- La première image à droite est le résultat obtenu après la segmentation
  - La zone en rouge représente les objets qui se déplace très rapidement
  - La zone en jaune représente les objets qui se déplacent moyennement rapidement
- La deuxième image à droite est la somme des images binaires obtenues après la segmentation
  - La zone en blanche représente les objets qui se déplace très rapidement
  - La zone en gris représente les objets qui se déplacent moyennement rapidement
- La deuxième image à gauche est l'image originale contenant les objets segmentés
  - La zone en rouge représente les objets qui se déplace très rapidement
  - La zone en bleu représente les objets qui se déplacent moyennement rapidement



FIGURE 11 – Seuillage : Seuil1 = 50, Seuil2 = 70, Kernel = 7x7

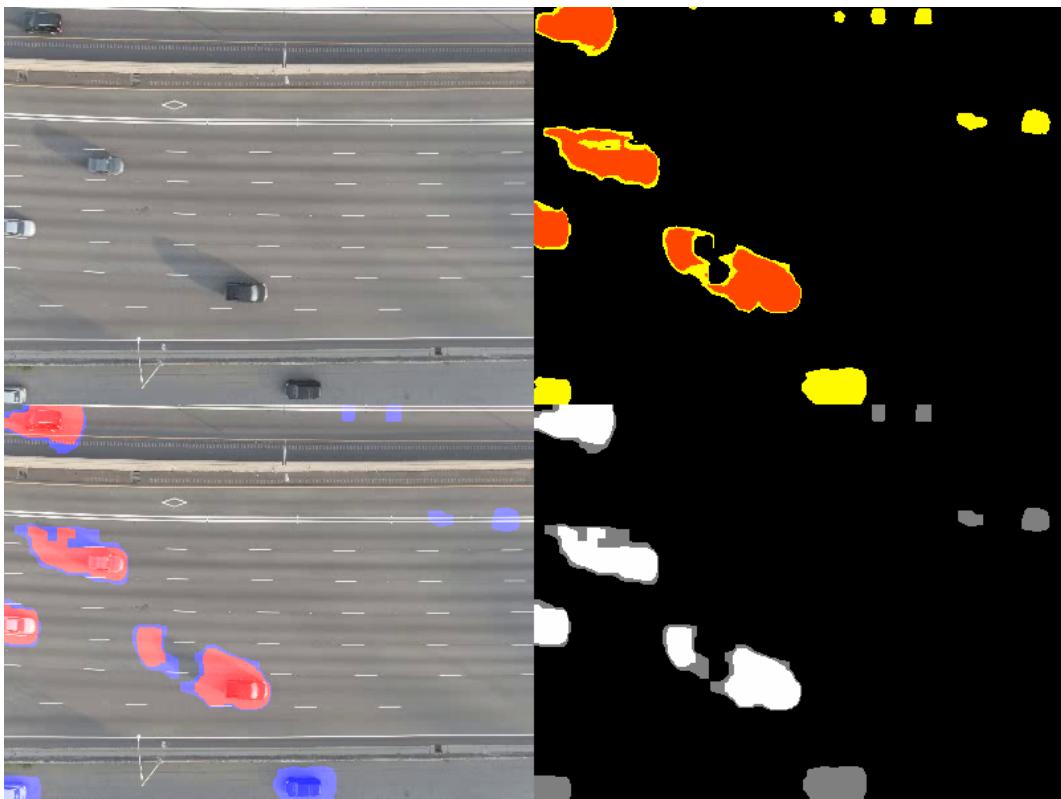


FIGURE 12 – Seuillage : Seuil1 = 50, Seuil2 = 130, Kernel = 7x7

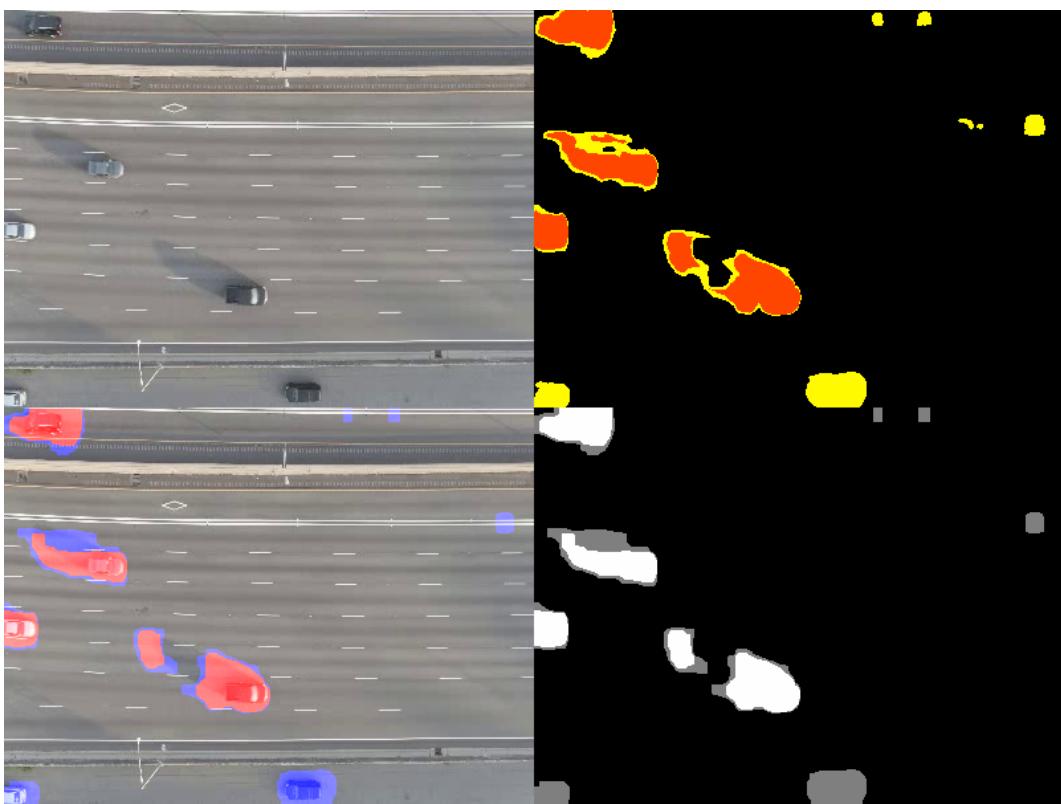


FIGURE 13 – Seuillage : Seuil1 = 70, Seuil2 = 150, Kernel = 7x7

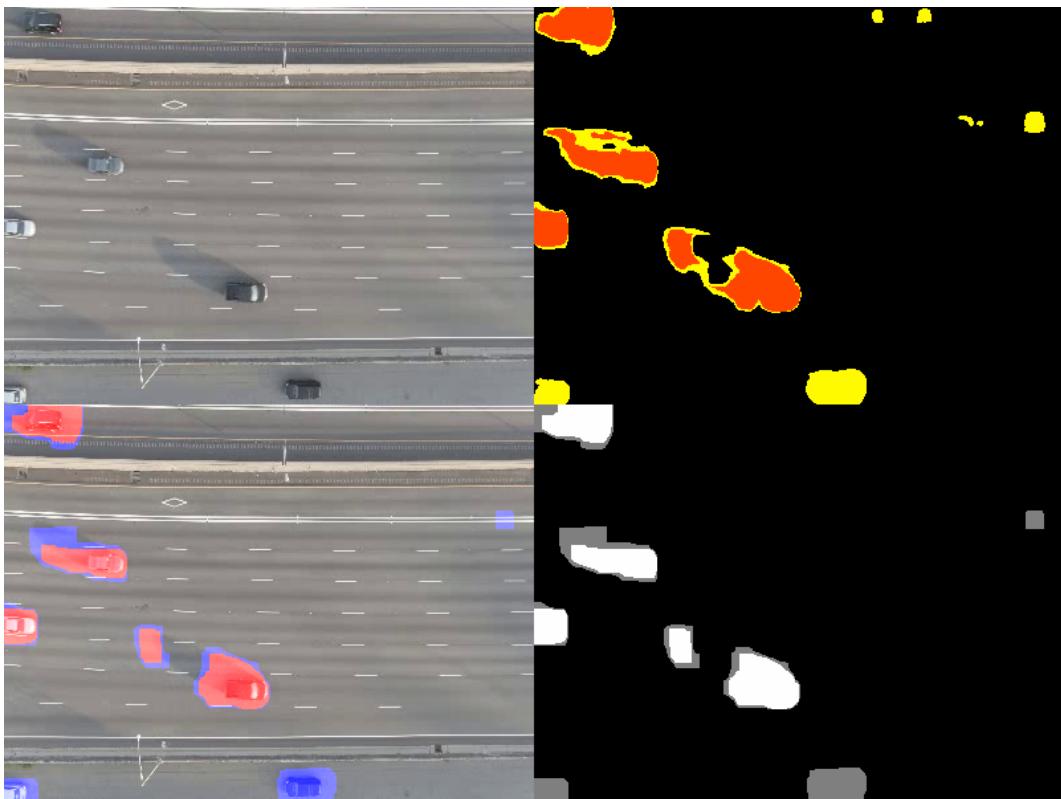


FIGURE 14 – Seuillage : Seuil1 = 70, Seuil2 = 150, Kernel = 11x11



FIGURE 15 – Seuillage : Seuil1 = 100, Seuil2 = 150, Kernel = 7x7

Nous pouvons remarquer que le choix des valeurs de seuil est important pour la segmentation. Pour la figure 11, nous avons choisi deux valeurs de seuil relativement basses. Malgré la différence de 20 entre les deux seuils, nous remarquons que tous les objets qui bougent dans l'image appartiennent à la même classe. Ceci est due au fait que tous les objets se déplacent à une vitesse plus grande que le choix du seuil le plus élevé.

Par contre pour le choix des seuils (50, 130) (figure 12) et (70, 150) (figure 13), nous remarquons que nous identifions deux catégories d'objets dans l'image. Toutefois, Le résultat obtenu avec le choix de seuils (70, 150) contient moins d'erreurs de segmentation que les seuil (50, 130).

Nous remarquons aussi que lorsque les deux valeurs de seuil sont trop élevées, nous perdons certains objets dans l'image (figure 15).

Nous remarquons également que le choix de la taille du noyau est important pendant la phase des opérations morphologiques pour raffiner les résultats. En effet, avec les seuils (70, 150), nous avons utilisé les valeur de noyau  $7 \times 7$  (figure 13) et  $11 \times 11$  (figure 14). Nous remarquons que le noyau  $11 \times 11$  donne un meilleur résultat parce qu'il réduit le nombre faux positif dans l'image.

Pour terminer, nous remarquons que les objets se déplacent avec leurs ombres dans l'image.

### Utilisation de Kmeans

Nous avons utilisé Kmeans afin de visualiser les objets qui se déplacent très rapidement et ceux qui se déplacent moyennement rapidement. Nous avons fait varier ces valeurs de K (avec  $K > 2$ ) afin d'observer leurs influences sur les résultats. Nous avons également utilisé plusieurs valeurs du noyau pour les opérations morphologiques sur les images binaires.

Dans notre implémentation, nous récupérons les objets dont la vitesse correspond aux deux clusters les plus élevés.

Nos résultats sont représentés en 4 images :

- La première image à gauche est l'image originale
- La première image à droite est le résultat obtenu après la segmentation
  - La zone en rouge représente les objets qui se déplace très rapidement
  - La zone en jaune représente les objets qui se déplacent moyennement rapidement
- La deuxième image à droite est la somme des images binaires obtenues après la segmentation
  - La zone en blanche représente les objets qui se déplace très rapidement
  - La zone en gris représente les objets qui se déplacent moyennement rapidement
- La deuxième image à gauche est l'image originale contenant les objets segmentés
  - La zone en rouge représente les objets qui se déplace très rapidement
  - La zone en bleu représente les objets qui se déplacent moyennement rapidement

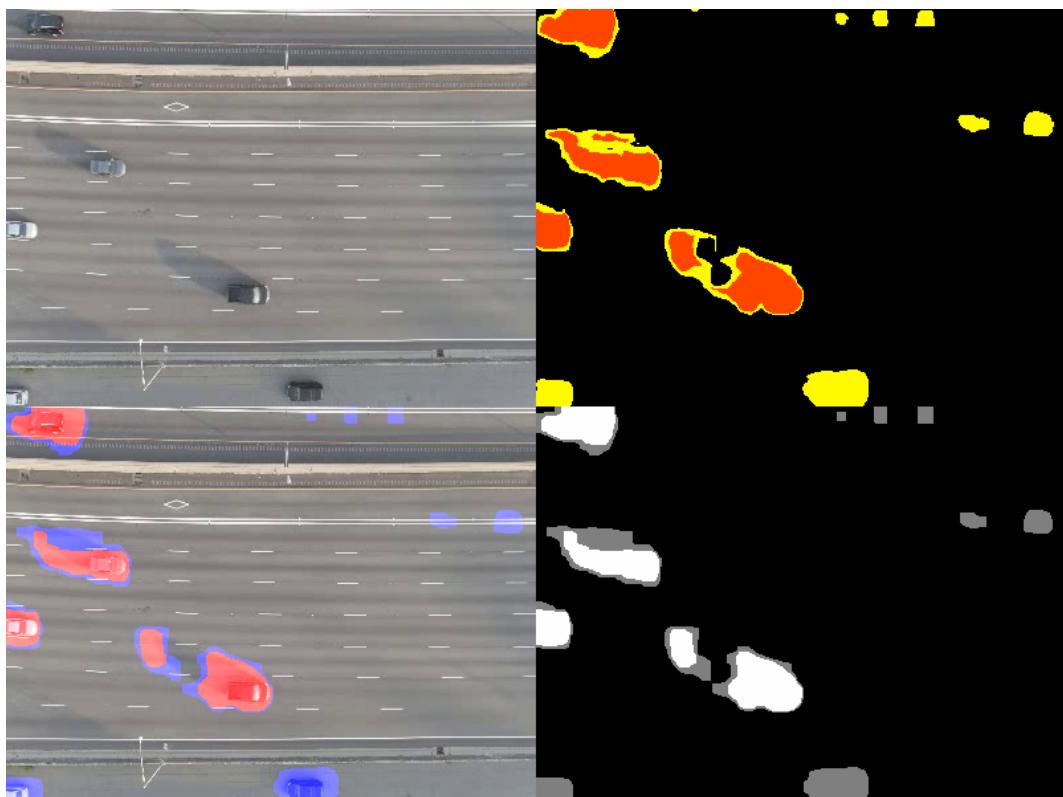


FIGURE 16 – Kmeans2 : K = 3, Kernel = 7x7

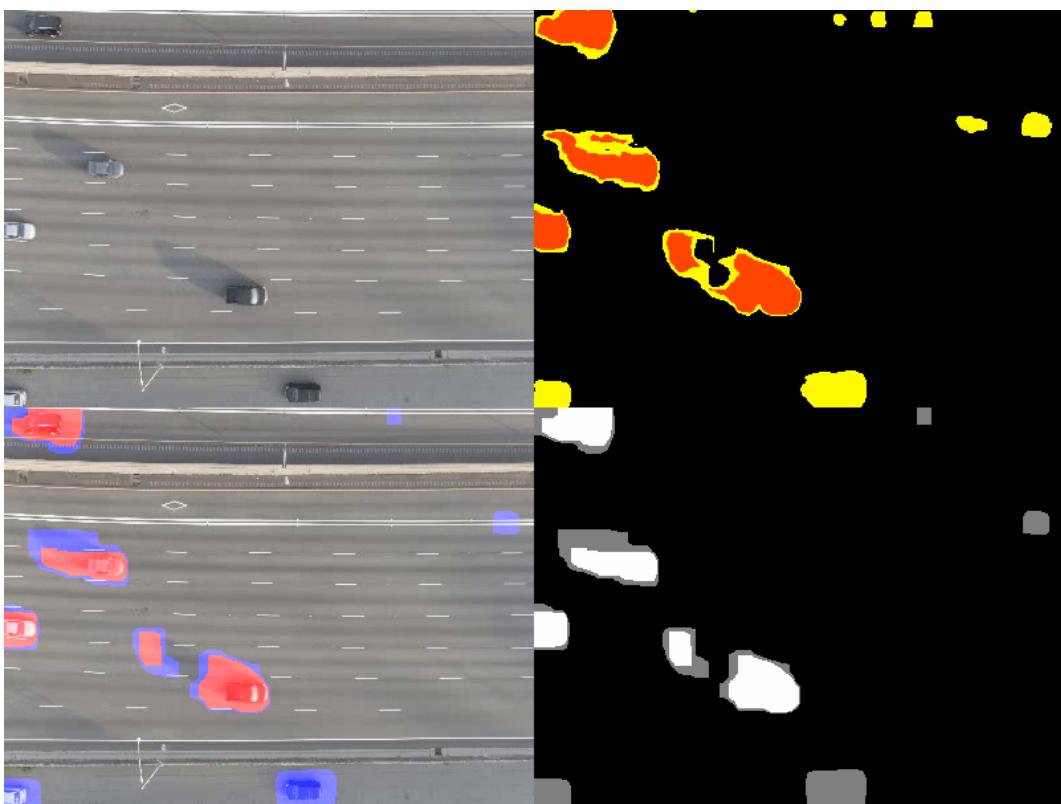


FIGURE 17 – Kmeans2 :  $K = 3$ , Kernel = 11x11

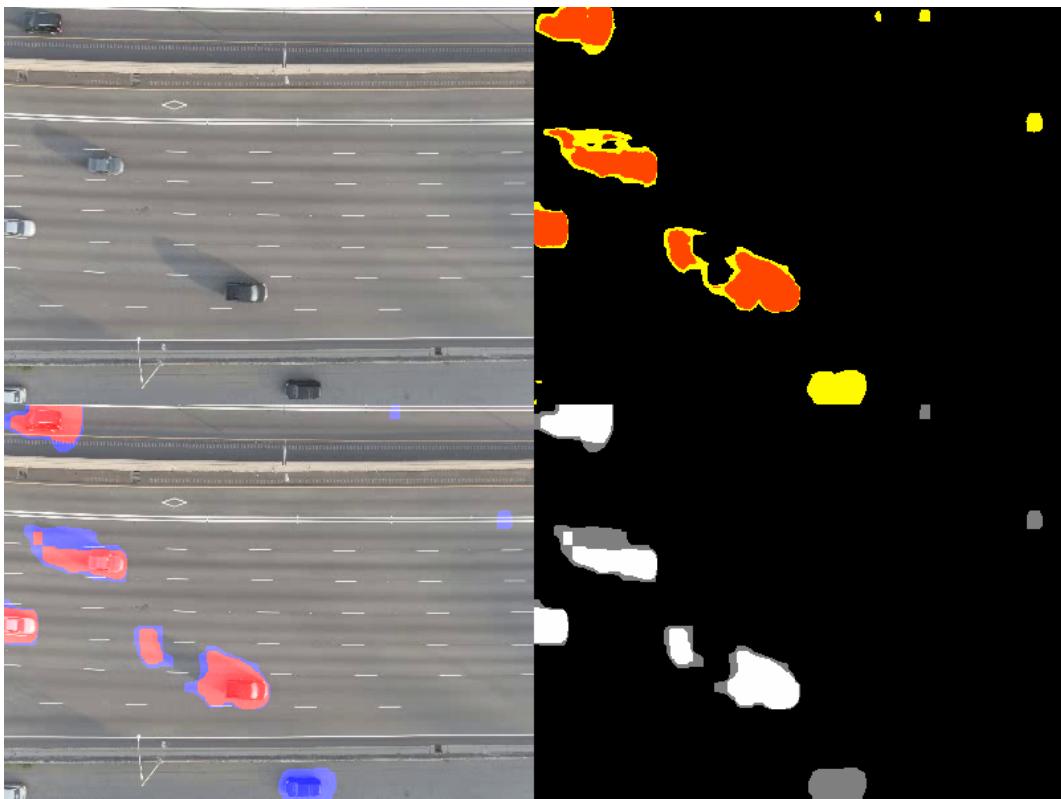
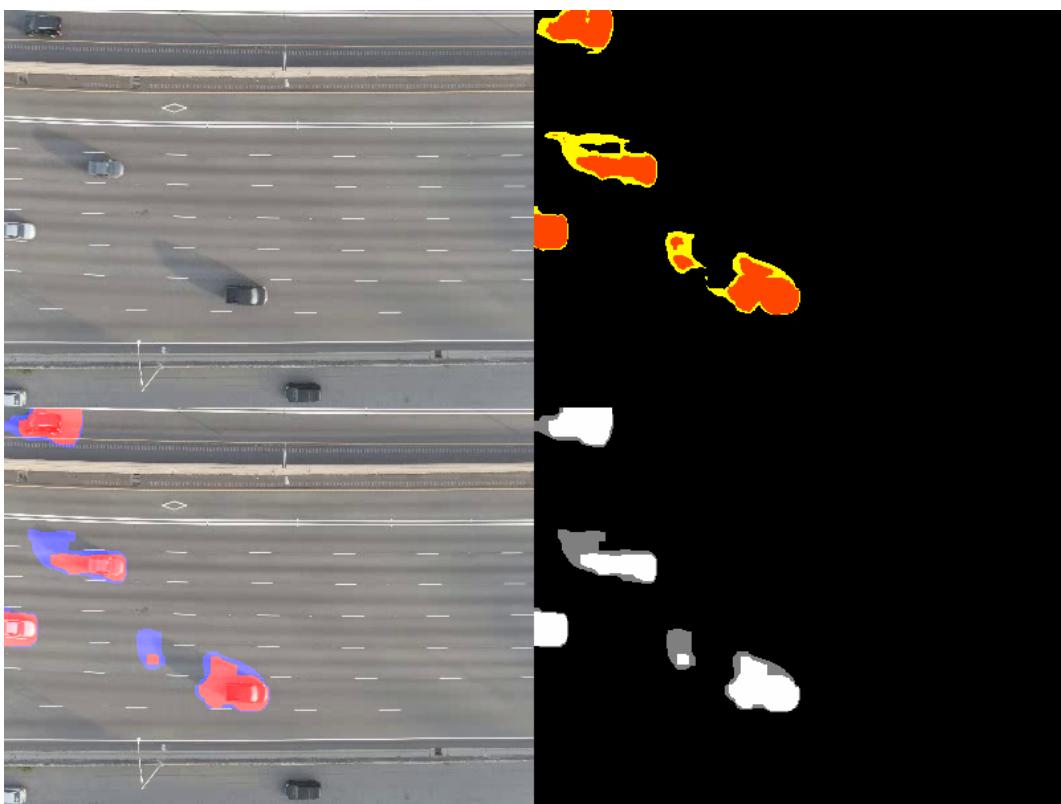


FIGURE 18 – Kmeans2 :  $K = 4$ , Kernel = 7x7

FIGURE 19 – Kmeans2 :  $K = 5$ , Kernel =  $7 \times 7$ 

Nous pouvons remarquer que le choix des valeurs de  $K$  est important pour la segmentation. Nous remarquons que lorsque  $K$  est trop élevé ( $K = 4$  et  $K = 5$ ), nous perdons certains objets (figures 18 et 19).

Par contre pour le choix des  $K = 3$  (figure 16), nous remarquons que nous identifions deux catégories d'objets dans l'image.

Toutefois, nous pouvons afficher tous les objets dont le cluster est différent de celui ayant la plus petite valeur de centre. Ainsi, nous pourrions peut-être visualiser tous les objets avec des valeurs de  $K$  plus élevées.

Nous remarquons également que le choix de la taille du noyau est important pendant la phase des opérations morphologiques pour raffiner les résultats. En effet, avec  $K = 3$ , nous avons utilisé les valeur de noyau  $7 \times 7$  (figure 16) et  $11 \times 11$  (figure 17). Nous remarquons que le noyau  $11 \times 11$  donne un meilleur résultat parce qu'il réduit le nombre faux positif dans l'image.

Pour terminer, nous remarquons que les objets se déplacent avec leurs ombres dans l'image.

### 3.3 Segmentation des objets en fonction de la norme et de l'orientation

L'objectif ici est de visualiser les objets de l'images qui se déplacent en fonction de leur vitesse et de leur direction de déplacement. Pour cela, nous allons utiliser la norme et l'orientation du flot optique de chaque pixel. Nous allons rechercher les pixels qui se déplacent le plus rapidement dans la vidéo et nous allons faire la segmentation en fonction de leur orientation. Nous allons utiliser Kmeans pour sélectionner les pixels qui se déplacent le plus vite et nous allons utiliser un seuillage pour la segmentation en fonction de l'orientation.

Pour la segmentation en fonction de l'orientation, nous allons utiliser quatre intervalle d'angle (en degré) :

- Entre 0 dégré et 90 dégré : Les pixels ont la couleur Rouge
- Entre 90 dégré et 180 dégré : Les pixels ont la couleur Verte
- Entre 180 dégré et 270 dégré : Les pixels ont la couleur Bleu
- Entre 270 dégré et 360 dégré : Les pixels ont la couleur Jaune

Pour améliorer les résultats que nous avons obtenus, nous allons appliquer les opérations morphologiques (ouverture, puis fermeture) sur l'image binaire obtenue après la segmentation par rapport à la norme. Nous allons utiliser une taille de noyau égale à  $7 \times 7$ .

Nous travaillons ici dans l'espace de couleur HSV pour afficher les résultats de segmentation. Nous avons testé cette segmentation sur trois vidéos différentes. Nos résultats sont représentés en 4 images :

- La première image à gauche est l'image originale
- La première image à droite est le résultat obtenu après la segmentation (norme + orientation)
- La deuxième image à droite est la somme des images binaires obtenues après la segmentation (norme)
- La deuxième image à gauche est l'image originale contenant les objets segmentés (norme + orientation)



FIGURE 20 – Piétons : K = 2, Kernel = 7x7

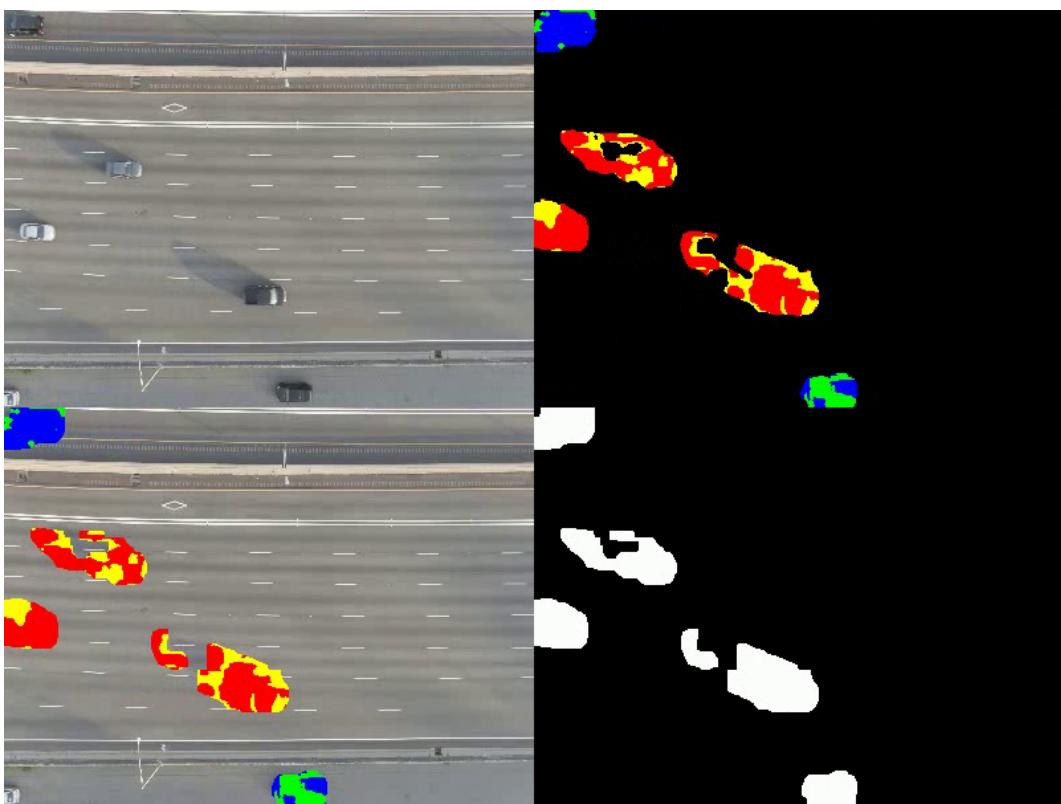


FIGURE 21 – Trafic : K = 2, Kernel = 7x7

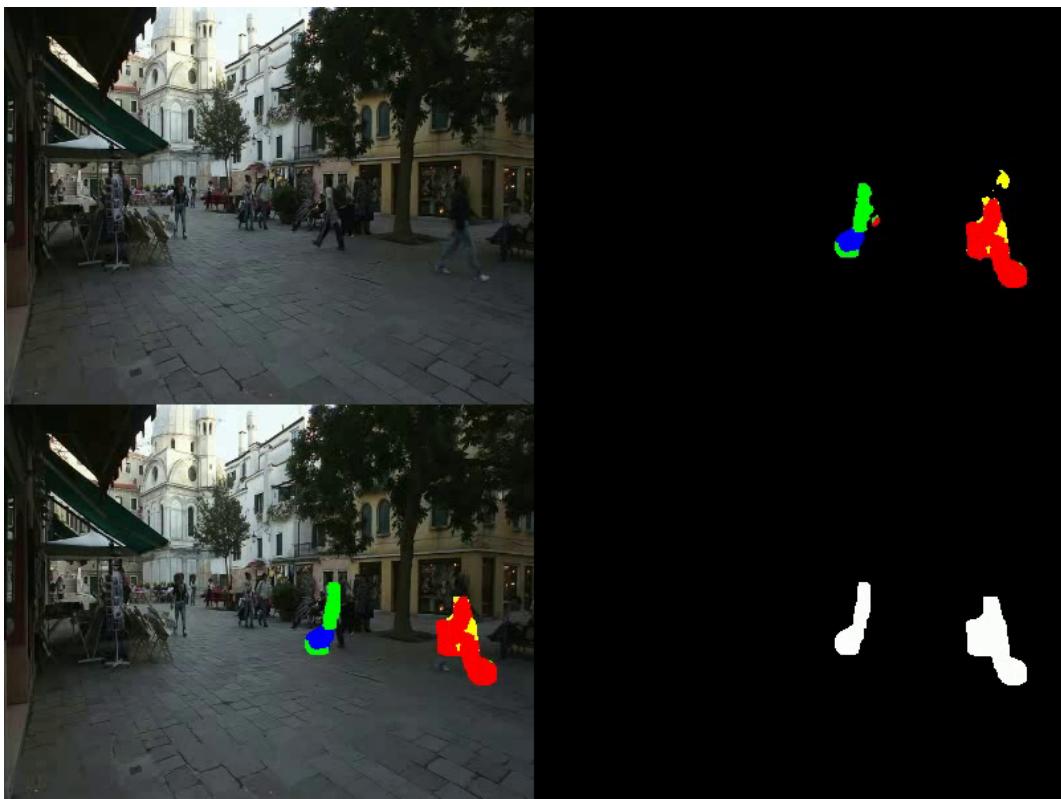


FIGURE 22 – Venise : K = 3, Kernel = 7x7

Nous remarquons que dans les différentes vidéos, les direction sont deux à deux confondues pour les différents objets. Nous remarquons Toutefois que l'ajout de l'information sur l'orientation du déplacement

des objets nous permet de les différencier. Il serait intéressant de diviser les angles en intervalles plus petit pour raffiner le résultat de segmentation.

## 4 Conclusion

Parvenu au terme de notre travail où il était question de faire la segmentation des objets en mouvement en utilisant le flot optique, nous pouvons dire nous avons atteint nos objectifs. nous avons obtenu des résultats assez intéressants, même si nous pensons qu'il est possible de les améliorer.

Après les nombreux tests que nous avons effectués en utilisant la méthode de seuillage, nous avons remarqué que le choix des valeurs de seuil est important pour la segmentation. Toutefois, il est important de préciser que le choix des valeurs de seuil dépend de la vidéo que nous segmentons. Il n'existe donc pas de choix de seuil qui soit universel.

Nous avons également remarqué que pour la méthode avec Kmeans, le choix de la valeur de K est importante. Le choix de cette valeur dépend du nombre de classe d'objet que nous voulons segmenter dans la vidéo.

Nous avons également utilisé les opérations morphologiques pour améliorer les résultats de la segmentation. Ces opérations utilisent un noyau dont le choix de la taille permet également d'affiner les résultats. Lorsque le noyau est trop petit, ces opérations n'influent pas sur le résultat et lorsque le noyau est trop grand, ces opérations ont tendance à rendre le résultat moins intéressant. Il est alors important de trouver une taille de noyau qui permettent d'améliorer les résultats. Ce choix dépend de la vidéo sur laquelle nous faisons la segmentation.

Ce rapport présente quelques uns des résultats que nous avons obtenus. Notre travail complet disponible [ici](#) donne accès à tous les résultats que nous avons obtenus en fonction de la méthode utilisée.