TP 2: Détection du fond - Flot optique
C1 - Vision Robotique

Gallego, Natalia
AST
Université Paris Saclay
natalia.gallego@ensta-paris.fr

I. Introduction

Dans ce travail, le problème de la détection et de la classification d'objets dans une séquence d'images est abordé sous diverses perspectives, en explorant à la fois les méthodes statistiques et les algorithmes avancés d'estimation du flux optique.

Premièrement, une approche simple basée sur des statistiques locales est proposée pour classer chaque pixel d'une séquence d'images en niveaux de gris comme appartenant à l'arrière-plan ou à un véhicule. Les paramètres et seuils utilisés sont discutés, ainsi que leurs limites dans des scénarios où les variations d'intensité ne sont pas représentatives des catégories à distinguer.

Par la suite, une limitation importante de l'approche précédente est identifiée du fait de la présence de variations atypiques des niveaux de gris de certains pixels. Pour remédier à cette lacune, une méthode qui exploite la cohérence temporelle dans les niveaux de gris des pixels est introduite.

Ensuite, un algorithme $\sum \Delta$ est implémenté et évalué, utilisé pour modéliser l'arrière-plan et détecter les changements dans les séquences vidéo. Les résultats obtenus sont présentés et l'impact des paramètres de l'algorithme sur les performances de détection est analysé, ainsi que la capacité de la méthode à différencier les véhicules et leurs ombres grâce à l'intégration d'informations chromatiques.

Enfin, le problème est abordé sous l'angle de la détection de mouvement. Utilisation d'un algorithme d'estimation classique des champs de déplacement horizontal et vertical (u, v) entre deux frames consécutives. Sur la base de ces résultats, une règle de décision simple est conçue pour déterminer si un pixel appartient à un objet en mouvement.

II. Approche simple basée sur (μ, σ)

Dans cette section, une approche améliorée est mise en œuvre pour détecter les véhicules en mouvement dans une séquence d'images prises par une caméra fixe située sur une route. Un sous-ensemble d'images $N = 300$ de la séquence entière est pris en compte, converti en niveaux de gris et stocké dans un tableau tridimensionnel.

Pour chaque pixel de cette séquence, la moyenne (μ) et l'écart type (σ) sont calculés en tenant compte de toutes les images de la séquence. Un critère basé sur le seuil (α) est appliqué pour générer un masque binaire qui classe chaque pixel comme faisant partie de l'arrière-plan ou représentant un mouvement :

$$|p(i,j) - \mu(i,j)_N| > \alpha \cdot \sigma(i,j)_N \tag{1}$$

Cette approche est testée sur un sous-ensemble spécifique d'images, en l'occurrence les images 50, 100, 200 et 300, pour visualiser les régions détectées en mouvement sous forme d'un masque binaire. Contrairement à l'approche précédente, les zones de mouvement sont représentées en blanc, tandis que les zones de quiétude sont représentées en noir.

De plus, des images moyennes et d'écart type sont calculées pour différents sous-ensembles N d'images (50, 100, 200 et 300) afin d'observer l'impact de la taille de l'échantillon sur les calculs statistiques.

A. Images Moyennes et d'écart Type

Les figures suivantes présentent les images moyennes et d'écart type pour différents sous-ensembles N d'images :

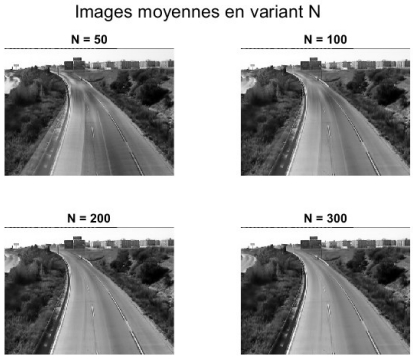

Fig. 1. Images moyennes en variant N

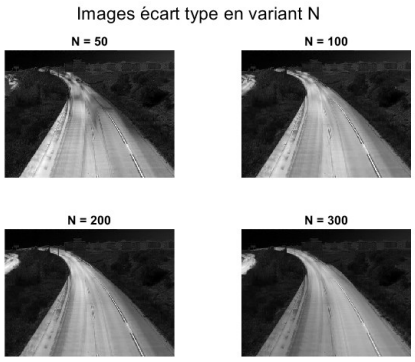

Fig. 2. Images écart type en variant N

Les images moyennes permettent de visualiser les tendances statiques de la scène, tandis que les images d'écart type illustrent les variations d'intensité des pixels sur la séquence, mettant en évidence les zones susceptibles de contenir du mouvement.

Les images moyennes calculées pour différents N (Figure 1) montrent une représentation statique de la scène, composée principalement de la route et des environnements fixes. En l'absence de véhicules, ces images se stabilisent en une version persistante de la route, avec un léger bruit résiduel qui diminue à mesure que N augmente. Cela reflète la convergence vers une moyenne plus précise lorsque davantage d'images sont utilisées.

Les images d'écart type (Figure 2) révèlent une certaine variation de l'intensité des pixels tout au long de la séquence. Ces petites variations sont principalement dues aux changements naturels de la route, tels que les ombres ou les reflets, plutôt qu'aux objets en mouvement.

B. Masque Binaire pour le Mouvement

Pour les images sélectionnées (50, 100, 200 et 300), des masques binaires sont générés en appliquant le critère décrit. Ces masques permettent d'isoler les zones de mouvement (en blanc) des zones statiques (en noir).

La méthode a été évaluée avec différentes valeurs de α.

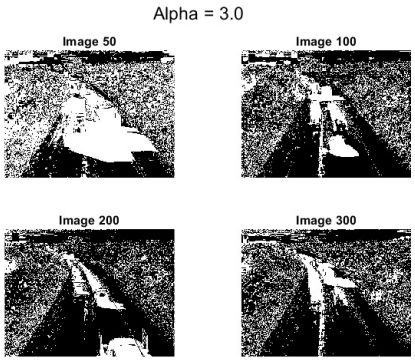

Fig. 3. Résultats avec seuil $\alpha = 0.5$

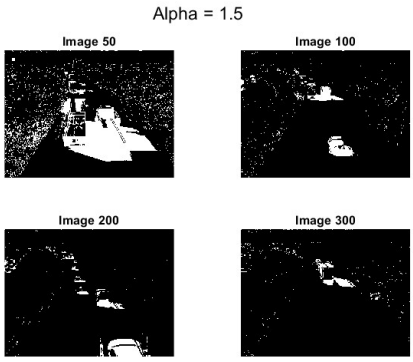

Fig. 4. Résultats avec seuil $\alpha = 1.5$

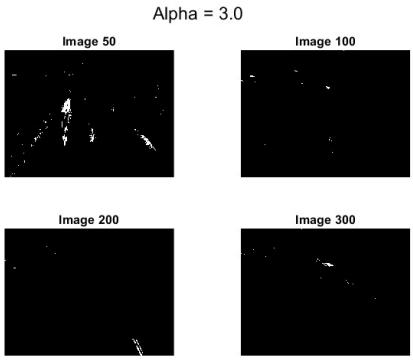

Fig. 5. Résultats avec seuil $\alpha = 3.0$

Ces masques binaires montrent comment les régions en mouvement sont capturées par l'algorithme en fonction des variations de luminance des pixels par rapport à la moyenne et à l'écart type. Cette représentation facilite l'interprétation des résultats et permet une évaluation rapide de la précision de la détection de mouvement.

Le seuil (α) définit la sensibilité du masque binaire généré. Un seuil faible ($\alpha = 0.5$) signifie que même de petites variations de l'intensité des pixels par rapport à la moyenne sont considérées comme un mouvement. Cela peut conduire à une surdétection, où les zones non pertinentes (telles que les ombres ou le bruit) sont classées comme mouvement. Cela peut être vu dans les résultats de la figure 3 où dans l'image, nous pouvons voir comment le système capture la nature autour de la route sous forme de pièces mobiles.

En revanche, un seuil élevé ($\alpha = 3.0$) réduit la sensibilité, ne permettant de détecter que des changements très significatifs. Cependant, cela peut également conduire à une sous-détection, dans laquelle certains objets en mouvement subtils ne sont pas identifiés. Dans le cas de T=80 (Figure 5) nous pouvons voir que pratiquement tous les objets en mouvement dans l'image sont ignorés et que seuls certains points présentant des changements très significatifs dans l'intensité de la couleur (voitures blanches par rapport au trottoir noir) sont identifiés comme objets en mouvement.

Dans le cas où $\alpha = 1.5$ (4), on constate une valeur du seuil plus équilibrée, ce qui permet de mieux séparer les objets environnementaux statiques des objets en mouvement. Cependant, cette méthode est encore loin d'être parfaite, car la façon dont nous pouvons visualiser le modèle est incapable de séparer ou de sélectionner de manière adéquate les bords des figures en mouvement au sein des images évaluées, et délimite seulement une ligne générale de mouvement de celles-ci.

III. Méthode pour identifier des plateaux dans les niveaux de gris de pixels

A. En raison de la variation du niveau de gris du pixel (200,150) on peut conclure que le calcul de (μ, σ) est assez trompeur, cela est dû à :

En calculant la moyenne et l'écart type de toutes les images sans prendre en compte la variation temporelle des pixels, il est possible de ne pas capturer les changements locaux dus aux objets en mouvement ou aux variations d'éclairage. Les pixels appartenant à l'arrière-plan peuvent avoir des valeurs très variables en fonction des conditions d'éclairage, mais cela ne signifie pas nécessairement qu'il y a du mouvement. En revanche, les objets en mouvement peuvent provoquer des variations spécifiques qui ne sont pas efficacement capturées par un calcul global.

Autrement dit, si un pixel appartient à l'arrière-plan, son échelle de gris peut varier en fonction de l'éclairage, des ombres ou d'autres effets environnementaux. Cependant, lors de la moyenne de toutes les images, ces variations sont intégrées de manière non discriminatoire, ce qui peut fausser les résultats. Par conséquent, la moyenne et l'écart type ne sont pas représentatifs de la stabilité ou de la dynamique des pixels d'arrière-plan et la détection de mouvement basée sur cette approche peut donc être incorrecte.

B. Méthode

Pour prendre en compte la cohérence temporelle des pixels de fond, une méthode est proposée pour identifier les plateaux de niveaux de gris des pixels à estimer (μ, σ). Autrement dit, au lieu de simplement calculer la moyenne (μ) et l'écart type (σ) globalement sur toutes les images, ce code utilise une fenêtre temporelle centrée sur chaque image pour calculer ces statistiques localement et les plus représentatives pour chaque arrière-plan. pixel, en fonction de son comportement temporel.

Pour y parvenir, on définit pour chaque image une plage temporelle autour d'elle qui couvre plusieurs images avant et après (dans les exemples présentés cette plage a été définie comme 5 images avant et 5 images après). Dans la fenêtre temporelle, la moyenne (μ) et l'écart type (σ) de chaque pixel au fil du temps sont calculés. Cela signifie que la variabilité de chaque pixel dans le temps est prise en compte. Si un pixel présente une variance temporelle élevée (c'est-à-dire que l'écart type est grand), cela indique qu'il y a probablement un mouvement dans cette région.

Ensuite, comme auparavant, avec l'écart type calculé, il est classé si un pixel appartient à l'arrière-plan ou s'il est en mouvement. selon un seuil (α). Les résultats obtenus avec cette nouvelle méthode pour différentes valeurs du seuil α sont présentés ci-dessous.

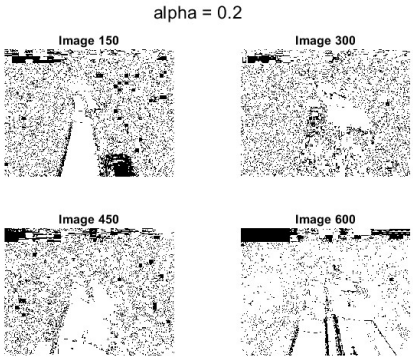

Fig. 6. Résultats avec seuil $\alpha = 0.2$

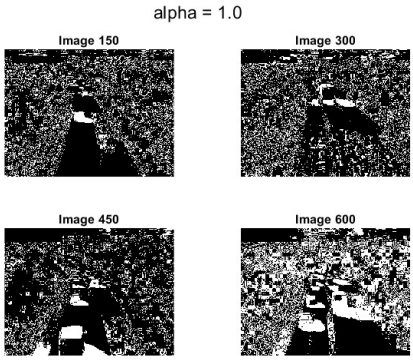

Fig. 7. Résultats avec seuil $\alpha = 1.0$

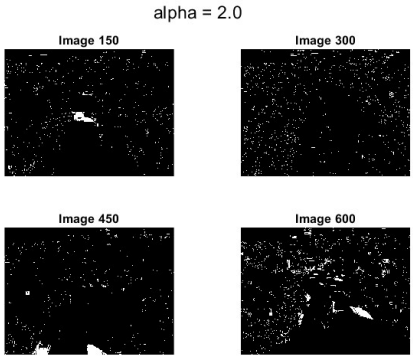

Fig. 8. Résultats avec seuil $\alpha = 2.0$

Comme précédemment, le comportement en fonction de la valeur de α varie fortement. Dans le cas d'un petit α ($= 0.2$), plus de pixels ont été détectés dans le cadre du mouvement car de petites variations dépasseront le seuil (Figure 6).

Pour les grandes valeurs de (α), telles que $\alpha = 2.0$, on constate que seuls les pixels présentant une variation significative

de l'écart type sont classés comme mouvement (7), c'est-à-dire grand blanc des voitures qui contrastent énormément avec la couleur grisâtre du trottoir. Cela a également amené, pour la plupart, les mouvements à l'intérieur de l'image à être ignorés et considérés comme des éléments statiques de la scène.

Le cas où $\alpha = 1.0$ est présenté comme la meilleure approximation de la méthode, car dans les 4 scénarios étudiés (8) il est possible de voir comment les objets en mouvement pendant toute la période sont délimités assez correctement . scène, et les objets statiques de celles-ci sont également définis.

IV. ALGORITHME $\sum \Delta$

Sur la base des informations de l'article Détection de mouvement : algorithmes rapides et robustes pour les systèmes embarqués [1], il a été décidé de tester **l'algorithme d'estimation Zipfian** présenté. Cet algorithme est conçu pour améliorer la détection de mouvement en combinant l'estimation $\sum \Delta$ (somme-différence) avec une distribution statistique **Zipf-Mandelbrot**. Cette approche vise à garantir que la fréquence de mise à jour en arrière-plan est proportionnelle à la dispersion de la distribution (variance). Les étapes effectuées au cours de l'algorithme sont expliquées ci-dessous et sont visibles sur la figure 9 :

1) Le seuil σ contrôle la périodicité de mise à jour des valeurs d'arrière-plan, ce qui signifie que les pixels sont mis à jour moins fréquemment dans les zones plus stables (où la variance est faible) et plus fréquemment dans les zones plus dynamiques.
2) Le modèle d'arrière-plan $M_t(x)$ est mis à jour en fonction de la valeur du pixel d'entrée $I_t(x)$ et du modèle d'arrière-plan précédent $M_{t-1}(x)$. Le modèle d'arrière-plan est ajusté en incrémentant ou en décrémentant sa valeur pour se rapprocher du pixel actuel. Si le pixel ne bouge pas, le motif d'arrière-plan reste constant.
3) $O_t(x)$ est calculé, qui mesure la différence absolue entre le modèle d'arrière-plan $M_t(x)$ et la valeur actuelle du pixel $I_t(x)$. Ce paramètre permet d'identifier les changements dans la scène.
4) La variance $V_t(x)$ est mise à jour toutes les trames T_V. Le paramètre N met à l'échelle la sensibilité du modèle aux différences de $O_t(x)$.
5) Enfin, il est décidé si le pixel bouge ou non en comparant $O_t(x)$ avec $V_t(x)$.

Algorithm 3: Zipfian estimation

[step #0: variance threshold computation]
find the greatest 2^p that divides $(t \mod 2^m)$
set $\sigma = 2^m/2^p$
foreach *pixel* x **do** [step #1": **conditional** M_t estimation]
  if $V_{t-1}(x) > \sigma$ **then**
    if $M_{t-1}(x) < I_t(x)$ **then** $M_t(x) \leftarrow M_{t-1}(x) + 1$
    if $M_{t-1}(x) > I_t(x)$ **then** $M_t(x) \leftarrow M_{t-1}(x) - 1$
    otherwise $M_t(x) \leftarrow M_{t-1}(x)$
  else
    $M_t(x) \leftarrow M_{t-1}(x)$
[**foreach** *pixel* x **do** [step #2: O_t computation]
  $O_t(x) = |M_t(x) - I_t(x)|$
foreach *pixel* x **do** [step #3": update V_t every T_V frames]
  if $t \mod T_V = 0$ **then**
    if $V_{t-1}(x) < N \times O_t(x)$ **then**
    $V_t(x) \leftarrow V_{t-1}(x) + 1$
    if $V_{t-1}(x) > N \times O_t(x)$ **then**
    $V_t(x) \leftarrow V_{t-1}(x) - 1$
    otherwise $V_t(x) \leftarrow V_{t-1}(x)$
    $V_t(x) \leftarrow max(min(V_t(x), V_{max}), V_{min})$
foreach *pixel* x **do** [step #4: $\hat{E}_t$ estimation]
  if $O_t(x) < V_t(x)$ **then** $\hat{E}_t(x) \leftarrow 0$ **else** $\hat{E}_t(x) \leftarrow 1$

Fig. 9. Algorithm Zipfianestimation [1]

Sur la base de l'étude de l'algorithme, il a été défini que le paramètre qui peut faire varier le plus les résultats lorsqu'ils sont modifiés est T_V, la période de mise à jour de la variance. En effet, cela a une influence directe sur la sensibilité du modèle aux changements de la scène. Sur cette base, les résultats suivants sont présentés :

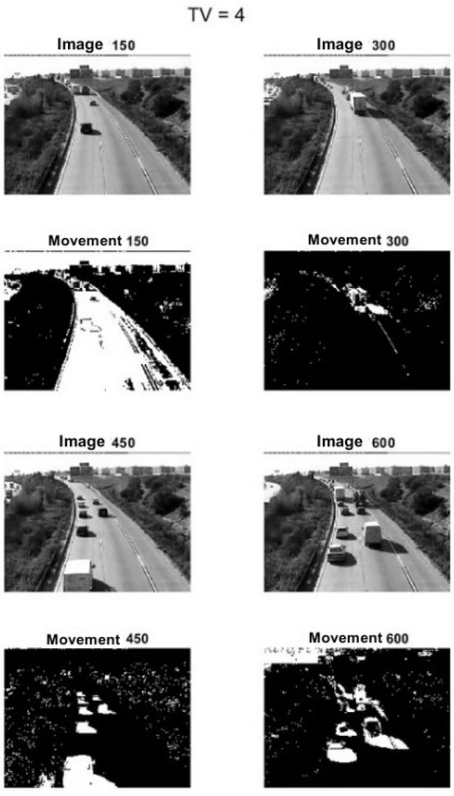

Fig. 10. Résultats avec seuil T = 4

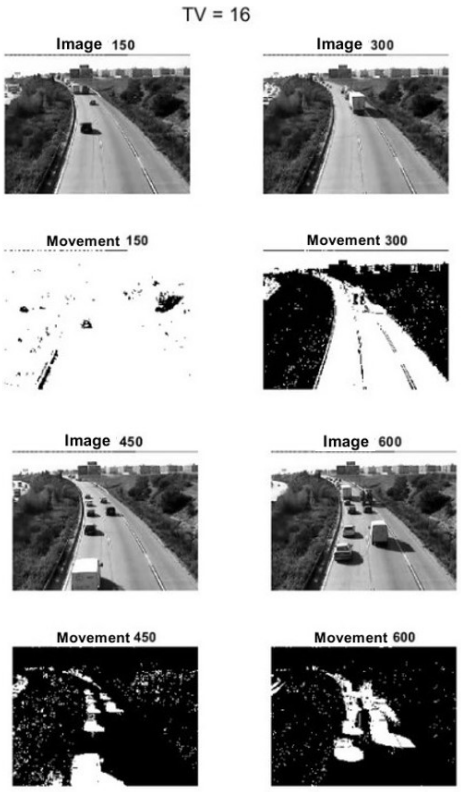

Fig. 11. Résultats avec seuil T = 16

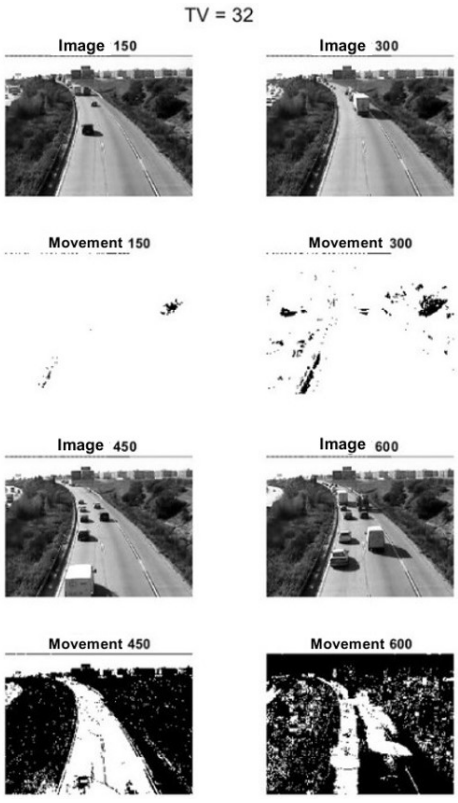

Fig. 12. Résultats avec seuil T = 32

Remarque : pour l'étude, les paramètres statiques ont été définis avec les valeurs suivantes :

- $N = 10$
- $V_{min} = 0$
- $V_{max} = 255$
- $m = 4$
- $\rho = 1$

Si T_V est petit ($T_V = 4$), l'algorithme répond plus rapidement aux changements de scène, comme on peut le voir sur la figure 10, où à partir de l'image 100 on voit une bonne séparation des objets dynamique et statique par rapport aux deux autres scénarios. Cependant, cette valeur rend le résultat plus sensible au bruit et peut produire davantage de faux positifs (pixels en mouvement mal détectés).

Dans le cas où T_V est grand ($T_V = 32$), le modèle est plus stable, mais il peut être trop lent pour s'adapter à des changements rapides, ce qui, comme on le voit sur la figure 12, provoque le 600 les images ne suffisent pas à définir complètement la séparation entre les objets en mouvement et les objets statiques. Contrairement au cas précédent, on constate que l'algorithme a encore du mal à définir correctement le fond de l'image.

Le résultat d'une valeur moyenne entre les deux cas précédents ($T_V = 16$) a également été étudié (figure 11). On voit là que, contrairement au cas où $T_V = 32$, le système parvient avec les 600 images à réaliser une séparation correcte du fond et des objets en mouvement, cependant ceci est encore moins précis que le résultat obtenu avec $T_V = 4$.

Grâce à ce qui précède, nous pouvons conclure que dans les scènes dynamiques, un faible T_V nous permet de nous adapter rapidement aux changements, tels que les objets en mouvement rapide ou les variations d'éclairage.

V. Approche simple basée sur (μ, σ) dans les images couleur

Il a été décidé de tester une approche avec les images couleur pour tenter d'éliminer la détection des ombres des voitures circulant sur la route. La solution est définie à l'aide des canaux de couleur **HSV**, qui permettent de séparer l'image en Teinte, Saturation et Luminosité.

Lorsque nous travaillons dans l'espace **HSV**, nous cherchons à atténuer le problème des ombres, car les canaux H et S sont plus invariants aux changements d'éclairage. Le canal H décrit la couleur pure, tandis que S indique la saturation ou la pureté de la couleur ; En revanche, le canal V (valeur), qui représente la luminosité, n'est pas utilisé pour calculer l'écart type. Par conséquent, cette approche peut réduire considérablement les erreurs de détection causées par les ombres.

Comme dans les exercices précédents, une fenêtre temporelle et un seuil sont définis pour classifier le mouvement. Par la suite, les images de la séquence sont converties de **RGB** en **HSV**, ce qui permet de travailler avec une représentation plus robuste face aux changements d'éclairage.

La moyenne et l'écart type des canaux **V** (Luminosité) et **S** (saturation) sont calculés. L'écart type de ces deux canaux

est utilisé pour détecter le mouvement. Cela nous permet d'identifier les pixels où le changement est significatif. Si le décalage dépasse les seuils défini, ce pixel est classé comme faisant partie du mouvement.

Les résultats obtenus avec cette méthode pour différentes valeurs du seuil T sont présentés ci-dessous.

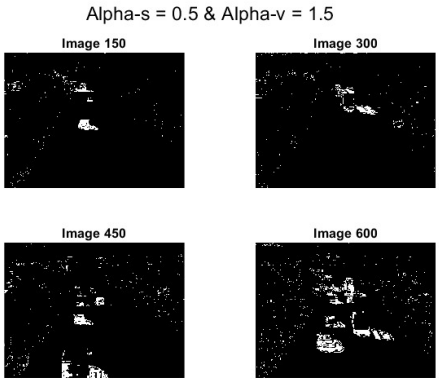

Fig. 13. Résultats avec seuil alpha˙s = 0.5 et alpha˙v = 1.5

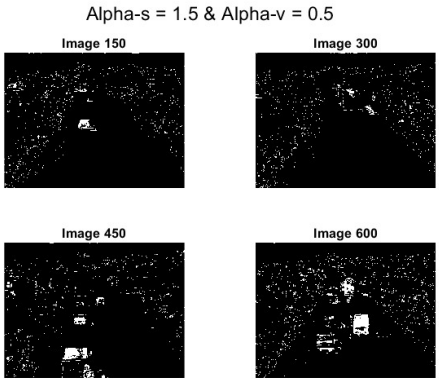

Fig. 14. Résultats avec seuil alpha˙s = 1.5 et alpha˙v = 0.5

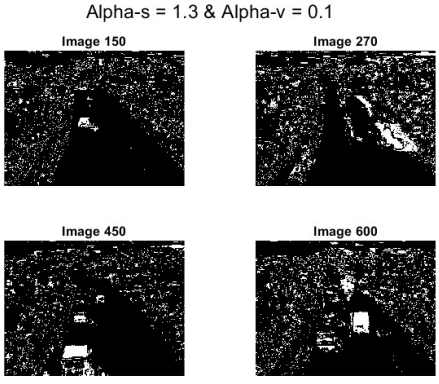

Fig. 15. Résultats avec seuil alpha˙s = 1.3 et alpha˙v = 0.1

Dans le premier cas, nous évaluons que le scénario $alpha_s$ (seuil pour le canal de saturation) est faible (0,5) et $alpha_v$ (seuil pour le canal de luminosité) est élevé (1,5) (Figure 13). Une faible valeur de $alpha_s$ devrait permettre de détecter le mouvement dans les régions à faible saturation, telles que les ombres ou les zones aux couleurs ternes. Dans le même temps, un seuil élevé dans le canal de luminosité filtre les pixels à faible luminosité, en se concentrant uniquement sur les zones plus claires. Cependant, comme on peut le constater, ce réglage ne suffit pas à éliminer les ombres, et ignore au contraire les zones de mouvement dans les régions plus sombres, comme les voitures dans les zones d'ombre.

En revanche, nous évaluons le scénario lorsque la valeur de $alpha_s$ est élevée (1,5) et celle de $alpha_v$ est faible (0,5) (Figure 14). Cela signifie que les zones de faible saturation, telles que les ombres, ont tendance à être supprimées plus facilement. D'un autre côté, un seuil faible dans le canal de luminosité permet de détecter les changements même dans les zones sombres, comme les ombres. Comme nous pouvons le voir, cette configuration améliore les performances de l'algorithme, en éliminant certains restes dans le mouvement ; Cependant, étant moins sensibles aux changements de saturation, les zones de mouvement à saturation modérée, telles que les objets aux surfaces grises, sont également ignorées dans certains cas.

En essayant d'améliorer les performances de l'algorithme et en voyant que la configuration d'une valeur élevée de $alpha_s$ et faible de $alpha_v$ semble être une bonne approximation pour éliminer les restes, nous avons essayé d'utiliser d'autres valeurs pour l'étude ($alpha_s = 1,3$ et $alpha_v = 0,1$). Ceux-ci, bien qu'ils améliorent les performances en termes de reconnaissance des voitures en mouvement, introduisent également de fausses zones de mouvement en arrière-plan des images (Figure 15).

VI. Algorithme Horn-Schunck

La méthode Horn-Schunck [2] est un algorithme classique pour estimer le flux optique entre deux images consécutives. Le flux optique décrit le mouvement apparent des intensités de pixels entre deux images, qui correspond au mouvement des objets de la scène ou de l'observateur. Cette méthode suppose une régularité dans le champ d'écoulement et minimise une fonction énergétique composée de deux termes : un terme de données et un terme de régularité.

Pour déterminer les dérivées spatiales et temporelles E_x, E_y, et E_t, le code utilise les approximations suivantes :

$$\begin{aligned}
E_x &\approx \frac{1}{4}[E_{i,j+1,k} - E_{i,j,k} + E_{i+1,j+1,k} - E_{i+1,j,k} \\
&\quad + E_{i,j+1,k+1} - E_{i,j,k+1} + E_{i+1,j+1,k+1} - E_{i+1,j,k+1}], \\
E_y &\approx \frac{1}{4}[E_{i+1,j,k} - E_{i,j,k} + E_{i+1,j+1,k} - E_{i,j+1,k} \\
&\quad + E_{i+1,j,k+1} - E_{i,j,k+1} + E_{i+1,j+1,k+1} - E_{i,j+1,k+1}], \\
E_t &\approx \frac{1}{4}[E_{i,j,k+1} - E_{i,j,k} + E_{i+1,j,k+1} - E_{i+1,j,k} \\
&\quad + E_{i,j+1,k+1} - E_{i,j+1,k} + E_{i+1,j+1,k+1} - E_{i+1,j+1,k}],
\end{aligned}$$

Ces dérivées permettent de mesurer comment l'intensité lumineuse change dans les directions x, y et au cours du temps.

L'approche de Horn-Schunck inclut un terme de régularisation basé sur le Laplacien pour lisser les champs de vitesse u et v. Les moyennes locales sont calculées en convoluant les champs u et v avec un noyau donné :

$$\text{Noyau} = \frac{1}{12}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 0 & 2 \\ 1 & 2 & 1 \end{bmatrix}.$$

Cela produit des champs moyennés $\bar{u}$ et $\bar{v}$, qui servent à calculer les nouvelles valeurs de u et v à chaque itération.

Solution itérative

L'algorithme Horn-Schunck résout les équations suivantes de manière itérative :

$$u^{n+1} = \bar{u} - \frac{E_x(E_x\bar{u}^n + E_y\bar{v}^n + E_t)}{\alpha^2 + E_x^2 + E_y^2},$$
$$v^{n+1} = \bar{v} - \frac{E_y(E_x\bar{u}^n + E_y\bar{v}^n + E_t)}{\alpha^2 + E_x^2 + E_y^2}.$$

Ici, α est un paramètre de régularisation qui contrôle l'importance du lissage. Ces équations garantissent que les champs de vitesse respectent les contraintes d'intensité tout en restant lisses.

Au cours de l'expérimentation, deux paramètres importants du système ont été modifiés, à savoir :

- Le paramètre α : Qui contrôle la fluidité du flux optique estimé. C'est un terme de régularisation qui pénalise les changements brusques du champ de flux optique entre pixels voisins. Autrement dit, cela favorise une estimation plus fluide et plus continue du flux optique, ce qui a un impact direct sur la façon dont le mouvement est estimé dans certaines zones de la vidéo.
- Le nombre d'itérations : cela détermine combien de fois le processus de mise à jour des champs de flux optique dans l'algorithme de Horn-Schunck est répété. En augmentant le nombre d'itérations, l'algorithme a plus de possibilités d'affiner son estimation du flux optique.

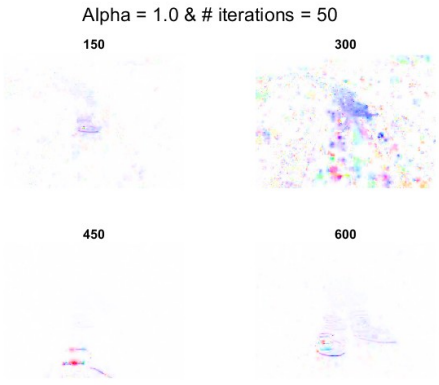

Fig. 16. Résultats avec seuil alpha = 1.0 et # d'itérations = 50

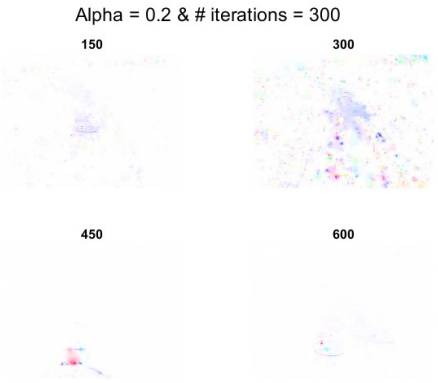

Fig. 17. Résultats avec seuil alpha = 0.2 et # d'itérations = 300

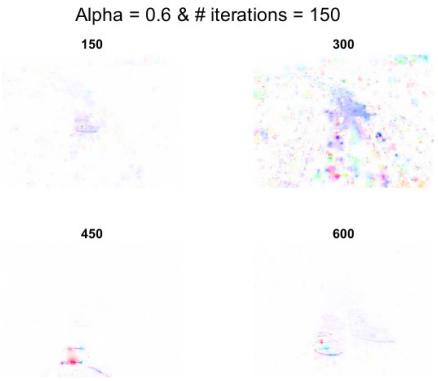

Fig. 18. Résultats avec seuil alpha = 0.6 et # d'itérations = 150

Dans une première expérience, la valeur de α est restée élevée ($\alpha = 1$), tandis que le nombre d'itérations est resté faible ($\#itérations = 50$). Cela provoque un fort lissage en raison du α, ce qui rend le flux optique plus fluide et plus continu. Cependant, comme nous le voyons, cela entraîne une perte de détails dans l'image. En revanche, une faible valeur du nombre d'itérations peut rendre les résultats moins précis, du fait de sa non-convergence. Il convient de noter qu'en raison de ce paramètre, ce scénario présentait le temps de traitement le plus court. Les résultats peuvent être vus dans la figure 16.

Une valeur α faible a alors été définie ($\alpha = 0,2$), tandis que le nombre d'itérations a été considérablement augmenté ($\#itérations = 300$). Cela a pour conséquence qu'en raison de α, le lissage du flux optique est faible, donc l'algorithme s'ajustera davantage aux changements locaux de l'image, en voyant les détails les plus fins. Cependant, comme on le voit, cela provoque l'introduction de bruit dans l'image. D'un autre côté, une valeur élevée du nombre d'itérations donne à l'algorithme plus de temps pour converger vers une solution plus précise. Cependant, ce changement augmente de façon exponentielle le temps d'exécution de l'algorithme. Les résultats peuvent être vus sur la figure 17.

Enfin, nous voulions revoir ce qui se passait lorsque les valeurs de α et $desitérations$ étaient définies à mi-chemin, c'est pourquoi nous avons décidé que $\alpha = 0,6$ et $\#itérations = 150$. Cela permet de contrôler le lissage du flux optique et de maintenir l'agrégation du bruit de l'image sous contrôle. De plus, le temps de convergence de l'algorithme lui permet d'atteindre une réponse moyennement précise. Les résultats de cette configuration peuvent être consultés dans 18.

VII. GitHub

Vous pouvez vous référer à la référence suivante pour voir les codes de nœud du projet : GitHub/C1

References

[1] Lionel Lacassagne, Antoine Manzanera, Antoine Dupret. Motion detection: Fast and robust algorithms for embedded systems. En: *International Conference on Image Processing (ICIP), Nov 2009, Le Caire, Egypt*. IEEE, 2009. Disponible en: https://hal.science/hal-01130889/document. 10.1109/ICIP.2009.5413946.

[2] Berthold K.P. Horn, Brian G. Rhunck. Determining Optical Flow. En: *Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, U.S.A.*, 1981. Disponible en: https://people.csail.mit.edu/bkph/papers/Optical_Flow_OPT.pdf.