INF6804 – Vision par ordinateur

Hiver 2019

TP1 – Segmentation vidéo

Groupe 01

1743175 – Bilal Itani

1740543 – Mehdi Kadi

Soumis à :

Hui-Lee Ooi

Soufiane Lamghari

1 février 2019

Table des matières

1. Présentation des deux approches à étudier

Dans le cadre de ce travail, nous avons eu la chance d'explorer deux techniques d'extraction de régions d'intérêts dans une séquence vidéo. Pour résoudre les problèmes de détection de changement, il existe deux approches communément utilisées. La première catégorie d'algorithme est celle permettant de faire une segmentation avant-plan/arrière-plan. La deuxième catégorie d'algorithme est celle par classification. Une technique de chacune de ses catégories a été mise en pratique.

La première technique, soit celle de la segmentation avant-plan/arrière-plan (MOG2) disponible dans la librairie d'OpenCV [2], est une approche améliorée de l'approche d'amalgames de distribution Gaussienne (Gaussian Mixture - GMM). Cette approche suit les méthodes traditionnelles de soustraction de l'avant-plan à l'arrière-plan, en construisant un modèle de distribution probabiliste pour chaque pixel de l'image. Nous discuterons en profondeur cette approche dans la section 1.1 du présent document.

La deuxième technique, soit celle par classification, est une approche plus moderne, utilisant un réseau de convolution de neurone. L'idée est de prendre une image et de la subdiviser en une grille où chaque cellule permet de détecter un objet appartenant à une classe prédéfinie avec un seuil (taux de confiance). Cette approche utilise un réseau de convolution en cascade qui permet d'extraire de convolution en convolution les régions importantes de l'image. L'approche de YOLO sera discutée en détail dans la section 1.2 du présent document.

1.1 Méthode de segmentation avant-plan/arrière-plan (MOG2)

Tel que mentionné précédemment, la méthode MOG2, implémentée dans la librairie d'OpenCV est une amélioration de la technique de soustraction d'arrière-plan par amalgames de distribution Gaussienne. Cette dernière est une méthode paramétrique vu que l'arrière-plan est modélisé par une loi normale pour laquelle les paramètres sont estimés. La méthode Gaussienne consiste à modéliser chaque pixel d'une trame sous une forme Gaussienne afin de tenir en considération du bruit et associe des poids à chaque pixel représentant le délai qu'une couleur demeure inchangé dans l'arrière-plan. Ce dernier se définit donc par les pixels ayant le moins de variation de couleur et demeurant le plus inchangés pour une période de temps.

Dans la libraire d'OpenCV, il existe une méthode nommée MOG. Il s'agit de la méthode standard de soustraction d'arrière-plan. Elle associe k distributions pour chaque pixel. Ainsi, nous avons un nombre fixe de distributions pour tous les pixels confondus d'une trame vidéo. Par rapport à la méthode MOG, la méthode MOG2 consiste en une amélioration, car elle sélectionne pour chaque pixel un nombre k de distributions pouvant varier d'un pixel à un

autre d'une même trame. Cette méthode permet donc de représenter plus fidèlement, dans certains cas, les changements de scènes, notamment les variations d'illumination, par un plus grand nombre de distributions.

D'autant plus, la méthode MOG2 permet à chaque lecture d'une trame subséquente, une mise à jour de l'image de référence afin de tenir en considération des changements d'intensités. Par rapport, aux méthodes non-paramétriques qui estiment la densité de probabilité par des histogrammes, la méthode d'amalgames de distribution Gaussienne assigne une loi de probabilité à chaque pixel. Une limitation de cette méthode est d'ailleurs due à l'utilisation élevée de mémoire vu qu'il faut stocker un nombre *k* de distributions pour chaque pixel [1].

1.2 Méthode de détection par classification

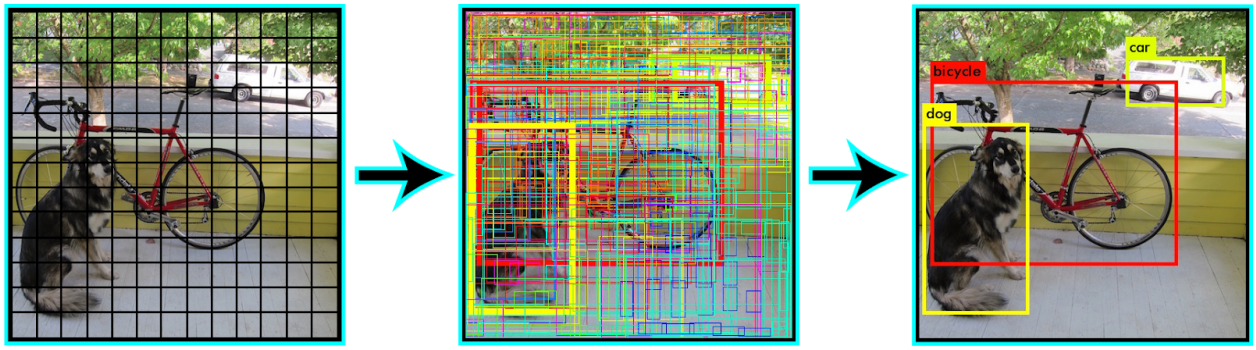

Fig. 1 : Visualisation de YOLO

La méthode qu'on utilise, soit YOLO, divise d'abord l'image en une grille. Chaque cellule de la grille peut détecter au plus un objet. Dans chacune des cellules de la grille, l'algorithme va prédire un nombre B de boîtes englobantes. Chacune de ces boîtes englobantes à un score de confiance. Peu importe le nombre de boîtes englobante dans la cellule, seulement un seul objet est détecté. Ainsi, si deux objets sont très proches, la méthode n'en détectera qu'un seul. De plus, une probabilité que de l'objet dans la cellule appartienne à une certaine classe est calculée pour chaque classe d'objets. Le score de confiance pour chaque objet détecté est la multiplication du score de confiance de la boîte englobante par la probabilité conditionnelle que l'objet appartienne à la classe. C'est avec ce score de confiance que YOLO détermine si un objet appartient à une certaine classe. Dans ce cas, une boîte englobante sera affichée autour de l'objet classifié, indiquant sa classe et son score de confiance. Chaque boîte englobante contient cinq paramètres, soit X, Y, W, H et le score de confiance. Les valeurs X et Y représentent le coin en haut à gauche, W représente la longueur de la boîte alors que H représente sa hauteur.

En résumé, l'idée générale de YOLO est de construire un tenseur (7,7,30). Pour ce faire, il utilise un réseau de neurones profond afin de réduire la dimension de l'espace à 7 x 7, avec un 1024 canaux. Une régression linéaire est ensuite appliquée pour générer une boîte de prédiction de taille 7 x 7 x 2 (e.g figure 1, image du centre). La

prédiction est faite à l'aide d'un seuil paramétrable. Par défaut, YOLO va préserver les prédictions ayant une valeur supérieure à 0.25 (e.g figure 1, image de droite).

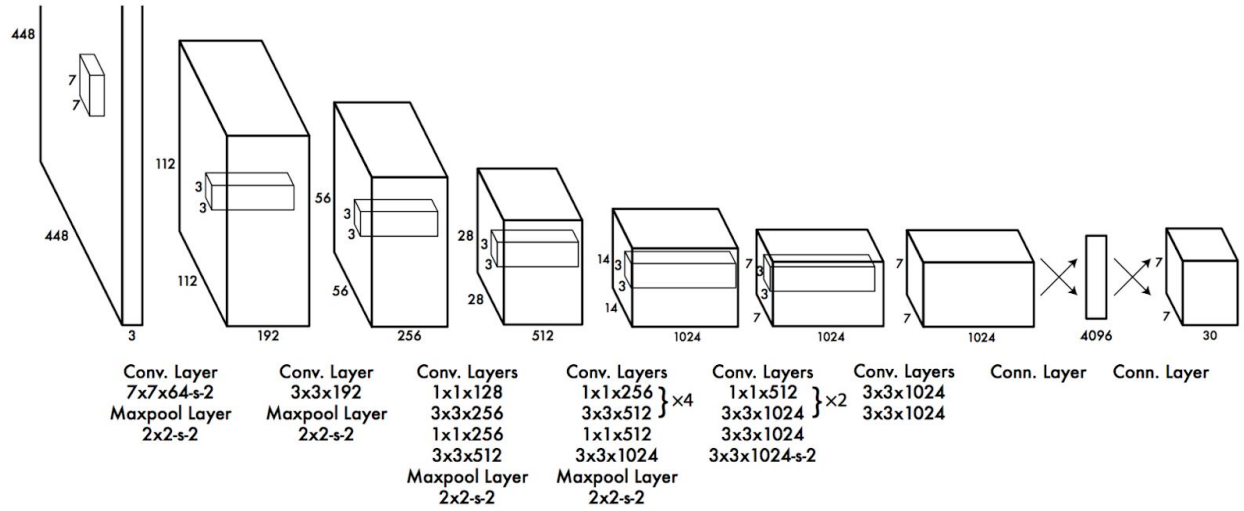

Fig.2 Réseau de convolution de neurone de YOLO

La figure 1, ci-haut, présente le réseau de neurones de YOLO. Il est possible de constater que le réseau transforme bien l'espace, à l'aide de convolution en cascade, en taille 7 x 7 x 1024, puis à l'aide d'une opération de régression linéaire, on obtient bien un tenseur de taille 7 x 7 x 30.

Lorsqu'on utilise un réseau de convolution de neurone, il faut l'entraîner sur des données d'entraînement afin d'apprendre les poids pour la détection des différentes classes. Cette étape prend énormément de temps, surtout lorsqu'on n'a pas les ressources nécessaires pour faire l'entrainement tels qu'une puissante carte graphique. Ainsi, nous avons plutôt choisi de prendre une version de YOLO pré-entraînée sur le jeu de donnée COCO [8], disponible sur internet.

2. Hypothèses de performance dans des cas spécifiques

Dans ce laboratoire, nous avons choisi d'étudier trois cas d'utilisation. D'abord, nous avons choisi d'étudier un cas très simple afin de comparer les deux méthodes, soit un cas où il y a une bonne luminosité sur l'image, de bons contrastes et que les objets de l'avant-plan se distinguent bien par rapport à l'arrière-plan. Il s'agit d'une séquence vidéo de véhicules circulant sur une autoroute en plein jour. Nous pensons que la méthode par classification, entraînée sur le dataset de COCO, sera plus performante que la méthode traditionnelle de segmentation. En effet, nous avons survolé les données de COCO et les objets dans les données d'entraînement sont bien définis dans plusieurs contextes, surtout lorsqu'il y a une bonne luminosité et un bon contraste. Dans le cas de la méthode par segmentation, il faut construire un modèle de l'arrière-plan pour pouvoir classifier les objets en circulation. Or, dans la séquence vidéo que nous traitons, certains véhicules, dus à leur couleur et aux reflets du soleil, peuvent avoir une couleur similaire à celle de la route. Ainsi, lors de la détection, le véhicule pourrait ne pas être détecté dans son entièreté. La figure 3, ci-bas, montre un exemple de trame qui, selon nous, risque de complexifier la détection de véhicules dans leur entièreté. Le véhicule encadré en rouge serait un défi à surmonter pour la détection.

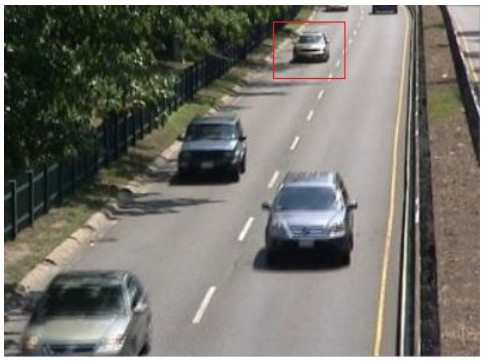

Fig. 3 : Trame contenant un véhicule difficile à détecter dans le cas de base

Pour le second cas d'utilisation, nous avons choisi une séquence vidéo de véhicules circulant sur une intersection en pleine nuit. Nous avons la ferme conviction que la méthode par classification, YOLO, entraîné sur une la base de données COCO performera beaucoup mieux que la méthode par segmentation. Effectivement, il existe plusieurs données dans la base de données d'entraînement COCO contenant des véhicules dans un environnement sombre, notamment de nuit. Pour ce qui est de la méthode par segmentation, le modèle d'arrière-plan pourrait ne pas permettre la détection des portions plus sombres du véhicule. La méthode pourra certainement détecter les phares du véhicule, mais la détection de l'arrière du véhicule, notamment le toit sera difficile, surtout pour les voitures ayant des couleurs sombres comme le noir. Aussi, nous pensons que la méthode YOLO aura aussi du mal sur cette séquence vidéo dans la mesure où elle n'a pas été entraînée sur un grand nombre de trames de véhicule avec une

faible luminosité. La figure 4, ci-dessous, montre un exemple de cas où il serait difficile pour la méthode par segmentation de détecter le véhicule.

Fig. 4 : Trame contenant un véhicule difficile à détecter dans le cas d'une faible luminosité

Finalement, la dernière expérience que nous voulons réaliser est en lien avec une séquence vidéo à très bas taux de trames qui représente la circulation sur une autoroute. Ainsi, les changements dans la vidéo d'une trame à l'autre sont très grands. Nous pensons que la méthode de segmentation aura une mauvaise performance dans ce type de séquence vidéo dans la mesure où le modèle d'arrière-plan ne pourra pas s'adapter rapidement aux changements dans la scène. Il sera plus compliqué de déterminer si un pixel fait partie de l'avant-plan ou de l'arrière-plan, car l'échantillonnage pour construire le modèle gaussien de l'arrière-plan est faible. Concernant la méthode YOLO, nous pensons qu'elle sera plus performante, car les données COCO utilisées pour son entraînement comportent beaucoup d'images de véhicule sous divers angles.

Afin de pousser notre raisonnement, nous pensons que la méthode de segmentation n'est pas adaptée aux caméras en déplacement. En effet, comme cette méthode construit un modèle de l'arrière-plan, ce modèle changera toujours à chaque trame en raison du déplacement de la caméra. Il sera ainsi difficile d'identifier les objets faisant partie de l'avant-plan. Pour la méthode YOLO, sa performance pour la détection dépend fortement des données d'entraînement. Il est évident que si YOLO n'a pas appris, lors de son entraînement, un certain type d'objet, il ne pourra pas le détecter. Fort heureusement, les séquences vidéos utilisées dans notre expérience sont très communes et YOLO est en mesure de détecter les classes d'objets dans ces séquences.

Le tableau 1, ci-dessous, permet de résumer les hypothèses émises dans la présente section :

Cas d'utilisation	Segmentation (MOG2)	Classification (YOLO)	Meilleur méthode
Cas de base (bon contraste, luminosité, etc)	- Bonne détection -Difficulté de détection des véhicules ayant une couleur similaire au sol, dû aux reflets de la lumière	-Bonne détection -Beaucoup de donnée d'entraînement de véhicule	YOLO
Vidéo avec faible luminosité et contraste	-Mauvaise détection -Difficile d'identifier un objet dans un environnement sombre dû à un faible contraste et une faible luminosité. -Forte adaptation de l'arrière-plan aux couleurs sombres, les véhicules peu illuminés ne seront pas détectables.	-Détection acceptable -Il peut être difficile d'identifier un objet dans un environnement sombre, peu de donnée d'entraînement avec environnement sombre	YOLO
Vidéo avec faible taux de trames	-Mauvaise détection -Adaptation difficile du modèle d'arrière-plan à la séquence vidéo dû aux changements très dynamiques.	-Bonne détection -Beaucoup de données d'entraînement de véhicules sous plusieurs angles.	YOLO

Tableau 1 : Résumé des hypothèses émises pour chacune des méthodes en fonction du cas d'utilisation

3. Description des expériences et bases de données

Dans le cadre de ce laboratoire, nous avons utilisé des séquences vidéos provenant de la base de données 2014 du site «changedetection » [7]. En effet, l'avantage de cette base de données est qu'elle fournit aussi les trames de vérité absolue (ground truth) que nous pouvons utiliser lors de l'évaluation des méthodes. Afin de réaliser nos expériences, nous avons choisi trois cas d'utilisation : un cas de base, un cas contenant un faible contraste et une faible luminosité et le dernier cas qui contient une séquence vidéo avec un faible taux de trames. Le cas de base se nomme «Highway » de la catégorie « Baseline ». Le cas avec un faible contraste et une faible luminosité se nomme « streetCornerAtNight » de la catégorie « Night videos ». Finalement, le cas avec un faible taux de trames se nomme « turnpike » de la catégorie « Low framerate » [7].

La séquence « Highway » est le regroupement de trames recueillies par une caméra qui filme des véhicules circulant sur une autoroute. La caméra observe les automobiles de face et légèrement de profil. La caméra est fixe et les véhicules sont bien identifiables. Une des deux difficultés que nous avons identifiées est le fait qu'il y a parfois des automobiles qui, avec le reflet du soleil, peuvent avoir une couleur similaire au sol qui fait partie de l'arrière-plan, ce qui peut compliquer la tâche de détection à la méthode par segmentation. La deuxième difficulté est le fait que deux automobiles peuvent être proche l'une de l'autre, ce qui peut empêcher la méthode YOLO de bien les détecter dû à leur proximité dans la trame.

Dans la séquence « streeCornerAtNight », la caméra observe l'intersection de deux rues durant la nuit. La scène est très sombre et le seul élément qui permet de distinguer un véhicule est la lumière de leurs phares. La carrosserie du véhicule est peu visible. Ainsi, la méthode par segmentation aura beaucoup de difficulté à détecter le véhicule en entier. Pour sa part, la méthode YOLO peut parfois ne rien détecter. En effet, il existe quelques images de véhicules dans un environnement sombre dans la base de données d'entraînement COCO. Cela complexifie aussi la tâche de détection pour la méthode YOLO ainsi que le fait que la caméra soit positionnée sur le côté droit de l'autoroute, ce qui permet d'observer les véhicules uniquement de profil.

Finalement, dans la séquence « turnpike », la caméra observe le haut des voitures circulant sur une autoroute. La particularité de cette séquence vidéo est qu'elle possède un taux de trame très bas. Ainsi, d'une trame à l'autre, il y a beaucoup de changements. Le modèle d'arrière-plan de cette vidéo, pour la méthode par segmentation, sera difficile à constituer dû aux grands changements de trame en trame. Quant à la méthode YOLO, elle devrait être en mesure de détecter adéquatement les véhicules. Par contre, il existe, parfois, des cas où des véhicules sont proches les uns des autres. Ainsi, tout comme pour la séquence vidéo « Highway », un seul véhicule peut être détecté dans une même cellule.

Afin de mesurer la performance des deux méthodes et de les comparer, nous avons identifié une méthode communément utilisée pour mesurer la performance des méthodes de classification nommée IoU (*intersection over union*). La figure 5, ci-dessous, illustre l'idée de cette méthode.

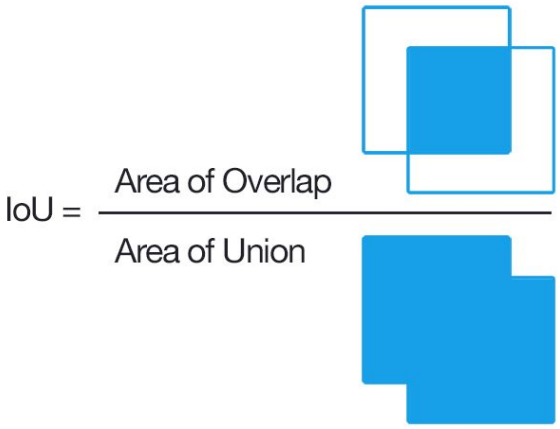

Fig. 5 : Illustration de la mesure IoU

L'idée est de mesurer l'intersection entre les boîtes englobantes des objets illustrés dans les séquences de vérité absolue avec les boîtes englobantes des objets obtenues à l'aide d'une méthode expérimentale. L'air représentant l'intersection de la boîte est ensuite divisé par l'union des boîtes englobantes pour obtenir un ratio, soit une mesure de la précision de la détection. Ainsi, un ratio égal à 1 signifie que la méthode expérimentale détecte parfaitement l'objet et 0 signifie que la boîte ne le détecte pas (aucune intersection). Il est important de préciser que nous avons dû tracer, nous-mêmes, les boîtes des objets autour des trames de vérité absolue dans la mesure où ceux-ci n'étaient pas fournis avec les données. Nous avons réussi à faire cela à l'aide de la librairie d'OpenCV.

Il peut paraître surprenant d'utiliser la méthode IoU avec la méthode par segmentation. Or, pour pouvoir comparer les deux méthodes, il nous faut une métrique qui peut être commune aux deux méthodes. Comme la méthode YOLO fournit déjà les données des boîtes englobantes des objets, nous avons jugé plus facile de construire les boîtes englobantes autour des objets de l'avant-plan de la méthode par segmentation. Nous avons ensuite relevé des métriques comme la moyenne, la médiane, le maximum et le minimum des ratios IoU pour chaque trame vidéo. Ces métriques seront présentées dans une section ultérieure du présent document.

4. Description des deux implémentations utilisées

4.1 Méthode de segmentation avant-plan/arrière-plan

Afin d'implémenter la méthode par segmentation MOG2, nous avons utilisé la documentation d'OpenCV. En effet, il existait déjà un squelette du code dans la documentation d'OpenCV [2]. Nous avons toutefois introduit l'utilisation d'arguments en ligne de commande pour faciliter l'utilisation du programme. En effet, afin de lancer notre programme, il faut spécifier le chemin vers le répertoire contenant les images de la séquence vidéo, il faut aussi spécifier le nombre de trames de la séquence, le chemin vers les trames de vérité absolue, et deux paramètres permettant de borner l'analyse des trames de la vérité absolue. En effet, ces deux derniers paramètres sont nécessaires dans la mesure où nous avons remarqué que les séquences vidéo n'avaient pas toutes les trames de vérité absolue. Certaines trames étaient tout simplement en gris, ce qui peut fausser notre analyse.

Outre les arguments, nous avons aussi ajouté une méthode permettant d'obtenir le nom du fichier de trame. En effet, nous avons remarqué que les données de « Changedetection » étaient toutes nommées suivant un patron bien défini. Cette méthode nous permet ainsi de lire les images de la séquence vidéo. De plus, nous avons ajouté deux méthodes, « intersection » et « union » qui permettent de détecter la région d'intersection et l'union entre deux rectangles.

Afin de traiter les images de la vidéo pour la segmentation, nous utilisons la méthode « cv2.createBackgroundSubtractorMOG2() ». En effet, l'algorithme de mélange gaussien de segmentation est déjà implémenté pour nous, il suffit uniquement de l'appliquer aux trames de la séquence vidéo. L'algorithme de segmentation a été décrit dans la section 1.1 du présent document. Cette méthode retourne une image en ton de gris qui contient tous les objets qu'elle a pu détecter dans l'avant-plan. Ces objets auront une couleur blanche. L'arrière-plan sera, quant à lui, en noir. Afin de bien faire ressortir les objets détectés, nous faisons un traitement morphologique sur l'image afin de pouvoir faciliter la création des boîtes englobantes autour des objets de l'avant-plan. Pour ce faire, nous utilisons un noyau morphologique en forme d'ellipse de taille 11 x 11. En effet, cette dernière taille a été déterminée de façon empirique. C'est elle qui permet de maximiser les ratios IoU pour la méthode par segmentation; ces données seront présentées dans la section 5 du présent document. À l'aide de ce noyau, nous appliquons une transformation morphologique d'ouverture. L'ouverture est en fait l'application d'une érosion sur l'image suivie d'une dilatation. Ceci permet d'enlever le bruit dans l'image des résultats, ce qui impacte grandement la création des boîtes englobantes. Toujours de façon empirique, nous avons remarqué qu'on obtient de meilleurs résultats lorsqu'on applique, après cette opération d'ouverture, une opération de dilatation puis encore d'ouverture.

Une fois les opérations morphologiques appliquées, nous utilisons la méthode opencv « cv2.findContours » autant sur l'image de vérité absolue que sur l'image des résultats. Cette méthode permet de retrouver les contours saillants des objets dans l'image. En utilisant cette méthode, nous lui passons trois paramètres, soit la trame expérimentale ou la trame de vérité absolue, « cv2.RETR_EXTERNAL » afin de retrouver le contour le plus grand associé à un objet et « cv2.CHAIN_APPROX_SIMPLE » qui permet de stocker en mémoire uniquement quatre points représentant le rectangle, ce qui permet d'épargner en termes de mémoire.

Une fois les contours trouvés pour l'image courante et l'image correspondante de la vérité absolue, nous faisons le calcul du IoU. Nos calculs utilisent un monceau maximum (*max heap*) afin d'identifier la meilleure boîte englobante d'un objet dans la trame courante qui correspond à la boîte englobante d'un objet de la trame de vérité absolue. Ceci permet essentiellement de toujours prendre la meilleure approximation, car il se peut qu'un objet dans la trame courante ait plusieurs rectangles pour un même objet, dû au bruit résiduel après les opérations morphologiques. On accumule l'IoU pour toute la séquence vidéo pour calculer la moyenne, la médiane, le maximum et le minimum. Ces données seront présentées dans la section 5 du présent document.

4.2 Méthode de détection par classification

Dans le cadre de ce laboratoire, il aurait été beaucoup trop long pour nous d'implémenter YOLO en entier et de l'entraîner. Nous avons trouvé une implémentation sur internet [4]. Afin d'utiliser YOLO, il faut aussi fournir les mêmes arguments que la méthode par segmentation décrits ci-haut. Il faut aussi fournir un chemin vers l'endroit où on voudrait stocker la vidéo résultante des détections de YOLO et deux seuils. Le premier seuil permet de filtrer les détections inférieures à celui-ci, l'autre seuil permet d'appliquer la suppression des non-maxima.

On commence, d'abord, par lire le dossier contenant les étiquettes de chaque classe et par générer les couleurs des boîtes associées à chacune des classes d'objets qui seront détectées. Ensuite, on effectue le chargement du modèle pré-entraîné de YOLO sur la base de données COCO. Pour ce faire, on utilise la méthode « cv2.dnn.readNetFromDarknet ». Pour chaque image de la séquence vidéo, il faut extraire sa dimension et déterminer le nom des couches de sorties. Ceci est fait à l'aide de la méthode « getLayerNamers ». Ensuite, la création d'un « blob » de l'image est faite. Un « blob » est une collection d'une image sur lequel des opérations ont été faites. Ces opérations permettent d'améliorer la performance du réseau de neurones, on peut y trouver des opérations de redimensionnement, de soustraction de la couleur moyenne afin d'obtenir une certaine résistance à l'illumination et de permutation de canaux de couleurs. Toutes les images d'un « blob » ont une même taille, un même nombre de canaux et subissent un même traitement. Une fois le « blob » créé, il est passé dans le réseau de neurones et on peut extraire les dimensions de la boîte englobante identifiant l'objet. C'est essentiellement les données de cette boîte qui seront utilisées en guise de comparaison avec les trames de vérité absolue. Il est à noter que nous extrayons de la

même façon que décrite dans la section précédente les objets contenus dans les trames de vérité absolue. Tout comme dans la dernière section, on effectue un calcul du IoU.

5. Présentation des résultats de tests

Lors de nos expériences, nous avons choisi de prélever la médiane, la moyenne, le minimum et le maximum des IoU. Pour chacune des séquences vidéos, voici les données que nous avons obtenues et une illustration de ce que notre programme fait sur chacune des séquences vidéos.

5.1 Résultats obtenus pour le cas de base - Highway

Tableau 2 : Résultat du test IoU sur la séquence Highway pour les méthodes MOG2 et YOLO

Séquence vidéo	Min (%)	Max (%)	Moyenne (%)	Médiane (%)
MOG2	0	97.2	61.6	73.7
YOLO	0	98.5	53.5	61.5

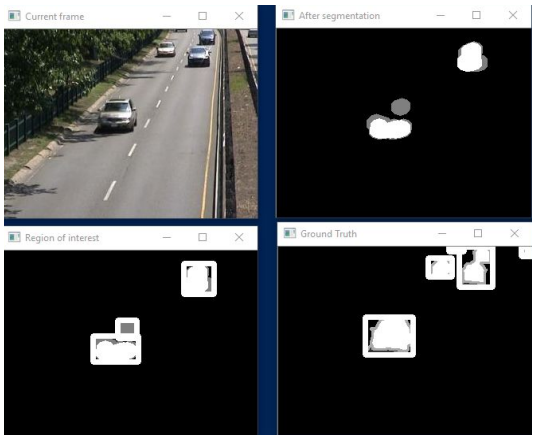

Fig.6 : Illustration de l'exécution de MOG2 sur la séquence Highway

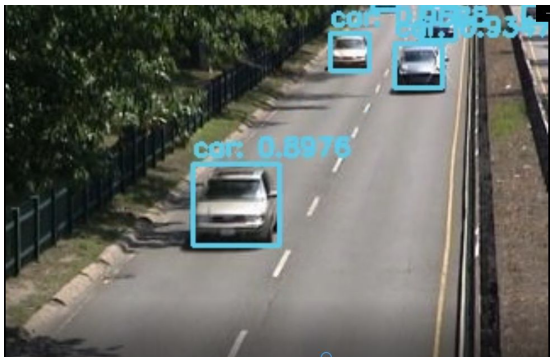

Fig. 7 : Illustration de l'exécution de YOLO sur la séquence YOLO

La figure 6 et la figure 7 montrent l'exécution des deux algorithmes sur une même trame. La figure 6 contient quatre fenêtres. De gauche à droite et de bas en haut, les fenêtres représentent : la trame courante, la trame après l'application de la segmentation MOG2, l'identification des objets de la trame courante à l'aide des boîtes englobantes, l'identification des objets de la trame courante des vérités absolues. L'identification des boîtes englobantes des objets dans les trames des vérités absolues est aussi faite; ceci n'est pas présenté dans la figure 7 afin de ne pas alourdir la représentation.

5.2 Résultats obtenus pour le cas de faible contraste / luminosité - StreetCornerAtNight

Tableau 3 : Résultat du test IoU sur la séquence StreetCornerAtNight pour les méthodes MOG2 et YOLO

Séquence vidéo	Min (%)	Max (%)	Moyenne (%)	Médiane (%)
MOG2	0	86.12	14.7	9.33
YOLO	0	98.6	78.9	79.8

Fig.8 : Illustration de l'exécution de MOG2 sur la séquence StreetCornerAtNight

Fig.9 : Illustration de l'exécution de YOLO sur la séquence StreetCornerAtNight

Note : La même explication qu'en 5.1 s'applique ici.

5.3 Résultats obtenus pour le cas d'un faible taux de trame - Turnpike

Tableau 4 : Résultat du test IoU sur la séquence Turnpike pour les méthodes MOG2 et YOLO

Séquence vidéo	Min (%)	Max (%)	Moyenne (%)	Médiane (%)
MOG2	0	96.4	43.6	43.9
YOLO	0	100	62.0	73.2

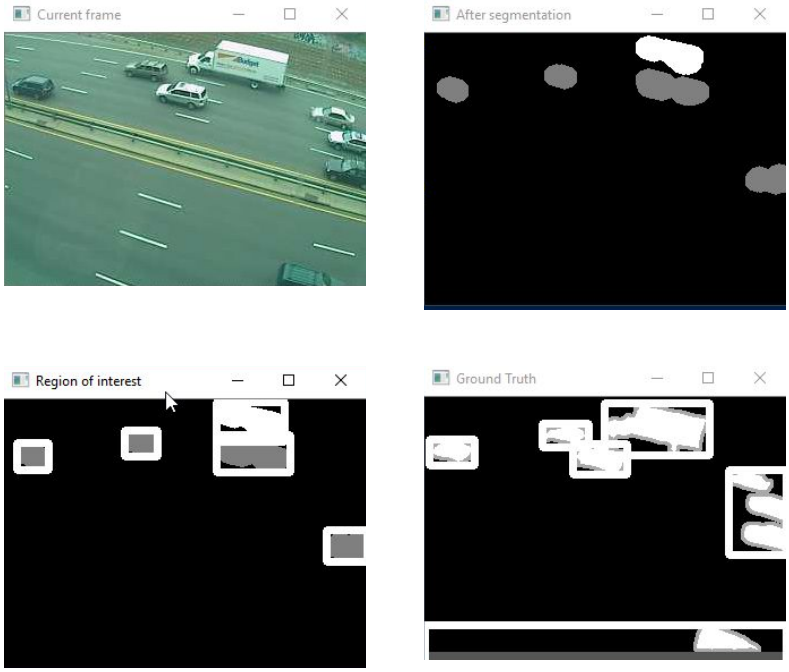

Fig. 10 : Illustration de l'exécution de MOG2 sur la séquence Turnpike

Fig. 11 : Illustration de l'exécution de YOLO sur la séquence Turnpike

Note : La même explication qu'en 5.1 s'applique ici.

6. Discussion des résultats et retour sur les hypothèses

6.1 Cas de base - Highway

À la lumière des résultats de la séquence vidéo « Highway », qui est le cas de base, nous avons été surpris de voir que la méthode par segmentation MOG2 performait mieux que la méthode YOLO. En effet, tel que montré dans le tableau 2, la méthode MOG2 avait eu une moyenne de IoU de 61.6% alors que YOLO a obtenu 53.5%. Ce qui pourrait expliquer ce genre de résultat est le fait que les automobiles peuvent être très proches avec l'angle de vue de face de la caméra. Ainsi, les boîtes englobantes autour de certains véhicules n'étaient pas toutes visibles. Il est aussi important de préciser que dans certains cas, la technique YOLO est beaucoup plus précise que la méthode par segmentation. En effet, il est possible de constater que le IoU maximal de YOLO pour cette séquence vidéo est de 98.5%, ce qui est très précis en comparaison à MOG2 qui a obtenu 97.2%. Dans les deux cas, le IoU minimal obtenu est 0%. Ceci survient lorsque la boîte englobante de la trame possède une intersection avec un côté de la boîte englobante de l'objet correspondant dans la trame de vérité absolue. La médiane indique que plus de la moitié des IoU obtenus pour MOG2 étaient supérieurs à 73.7%. Pour YOLO, cette médiane est de 61.5%.

Bien que les résultats indiquent que MOG2 était plus performant que YOLO sur cette séquence vidéo, nous tenons à préciser que nous avons nous-mêmes dû tracer les rectangles englobants autour des objets d'avant-plan dans les trames de vérités absolues, car leurs coordonnées n'étaient pas fournies par « Changedetection.net ». Ceci peut avoir un impact sur les résultats tel qu'il est possible de le constater à la figure 6. Sur la fenêtre du bas à droite de cette dernière, il est possible de constater qu'une boîte englobante englobe deux voitures, car elles étaient très proches et les pixels de l'une étaient collés à l'autre. Ceci impacte grandement les valeurs IoU, puisque l'union de la boîte englobante trouvée par YOLO avec celle de la vérité absolue est très grande par rapport à l'intersection.

Les résultats obtenus pour le cas de base infirment les hypothèses que nous avons émises dans la section 2 du présent laboratoire. Il semble que le modèle d'arrière-plan maintient tout de même une excellente détection des véhicules circulant sur l'autoroute.

6.2 Cas avec faible contraste / luminosité - StreetCornerAtNight

Pour ce qui est de la séquence vidéo avec un faible contraste et une faible luminosité, il est possible de constater, dans le tableau 3, que la méthode YOLO est de largement plus performante que MOG2. En effet, la moyenne des IoU de YOLO est de 78,9% en comparaison avec 14.7% pour MOG2, ce qui est très décevant. La médiane de MOG2 est seulement de 9.33%, donc la moitié des IoU étaient inférieurs à 9.33%, en comparaison avec 79.8% pour YOLO. Or, dans certains cas, MOG2 obtient de bon IoU. En effet, le IoU maximal de MOG2 était de 86.1%, en comparaison avec 98.6% pour YOLO. Ces résultats sont facilement explicables en regardant la fenêtre du bas à gauche et la fenêtre du bas à droite de la figure 8. La fenêtre du bas à gauche montre la segmentation de MOG2, avec la boîte englobante des objets de l'avant-plan. Il est possible de constater que les reflets lumineux causés par les phares des véhicules de circulation ont été la cause du manque de précision de la méthode. En effet, ces reflets lumineux étaient considérés comme faisant partie de l'avant-plan, ce qui ne devait pas être le cas. Ceci génère de très grandes boîtes englobantes en comparaison avec ceux de la vérité absolue, ce qui explique les faibles IoU pour la méthode MOG2.

Cette expérience à permis de confirmer nos hypothèses. En effet, il est possible de constater que les véhicules ne sont pas détectés à la perfection, car la figure 8 montre que l'arrière du véhicule fait partie de l'arrière-plan. Il y a aussi une forte adaptation de l'arrière-plan aux couleurs sombres. Ceci explique pourquoi les reflets causés par les phares du véhicule font partie de l'avant-plan. Pour la technique YOLO, il est possible de constater qu'il y a une bonne détection des objets dans la scène. La figure 8 montre que YOLO a réussi à identifier les véhicules avec un excellent taux de confiance, YOLO a donc été entraîné avec des images de faible contraste et luminosité pour couvrir les séquences comme celle-ci.

6.3 Cas avec un faible taux de trames - Turnpike

Pour ce qui est du cas d'utilisation où la vidéo possède un faible taux de trame, il est possible de constater, à l'aide du tableau 4, que la technique YOLO performe encore une fois mieux que MOG2. En effet, la moyenne des IoU de la méthode MOG2 est de 43.6% en comparaison avec 62.0% pour YOLO. La médiane des IoU est de 43.9% pour MOG2 et 73.2% pour YOLO. Le meilleur indice IoU pour YOLO est de 100% alors qu'il est de 96.4% pour MOG2. Ainsi, il y a eu un cas où YOLO a détecté de façon exacte un objet avec de la scène. La figure 10 montre un exemple d'exécution de l'algorithme MOG2 sur une trame de la séquence vidéo. Il est possible de constater, dans la fenêtre du bas à gauche, que ce n'est pas tous les véhicules qui sont détectés. En effet, ceci est dû au fait que le modèle d'arrière-plan ne s'adapte pas bien à ce qui est perçu d'une trame à l'autre dans la mesure où il y a beaucoup de changement d'une trame à l'autre. Inversement, YOLO détecte beaucoup mieux les objets de l'avant-plan avec un taux de confiance très élevé, tel que montré à la figure 11. Ceci est encore une fois dû à l'entraînement de YOLO. On constate qu'il a bien été entraîné sur des cas où la caméra observe un véhicule de haut dans de bonnes conditions météorologiques.

Tel que mentionné dans la section 6.1, lors de l'analyse du cas de base, nous avons dû nous même déterminer les boîtes englobantes des trames de vérités absolues; la figure 10 en montre un excellent exemple. La fenêtre du bas à droite montre les boîtes de la vérité absolue. L'algorithme d'OpenCV permettant de dessiner les boîtes englobantes n'a pas fait un bon travail dans la mesure où les trois véhicules à l'extrémité droite font partie d'une même et unique boîte, contrairement à trois boîtes distinctes. Ceci impacte de façon similaire l'analyse des méthodes MOG2 et YOLO.

Cette expérience a permis de confirmer les hypothèses que nous avons émises. Bien que MOG2 performait bien sur certaines trames de la séquence vidéo, la moyenne de détection est faible en comparaison avec YOLO, ceci est dû au fait qu'il y a beaucoup de changement d'une trame à l'autre. Ainsi, certains véhicules qui aurait dû faire partie de l'avant-plan n'ont font pas partie. Par contre, YOLO qui a été entraîné sur ce genre d'angle de vue et d'objet performe très bien.

7. Bibliographie

[1] Notes de cours, chapitre 2 [P.12 - P.28], INF6804 - Vision par ordinateur

[2] « OpenCV: Background Subtraction ». [En ligne]. Disponible à: https://docs.opencv.org/master/db/d5c/tutorial_py_bg_subtraction.html#gsc.tab=0. [Consulté le: 15-janv-2019].

[3] « Real-time Object Detection with YOLO, YOLOv2 and now YOLOv3 ». [En ligne]. Disponible à: https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088. [Consulté le: 15-janv-2019].

[4] « Intersection over Union (IoU) for object detection - PyImageSearch ». [En ligne]. Disponible à: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/. [Consulté le: 17-janv-2019].

[5] « YOLO object detection with OpenCV - PyImageSearch ». [En ligne]. Disponible à: https://www.pyimagesearch.com/2018/11/12/yolo-object-detection-with-opencv/. [Consulté le: 17-janv-2019].

[6] « changedetection.net ». [En ligne]. Disponible à: http://changedetection.net/. [Consulté le: 20-janv-2019].

[7] « Change detection benchmark web site ». [En ligne]. Disponible à: http://jacarini.dinf.usherbrooke.ca/dataset2014/. [Consulté le: 20-janv-2019].

[8] « COCO - Common Objects in Context ». [En ligne]. Disponible à: http://cocodataset.org/#home. [Consulté le: 30-janv-2019].