Chaimaa Aajafir
Jiacheng Liu
Hugo Paugam

Luminaire Intelligent

P2RV

27/03/2019

Présentation du projet :

Le projet de luminaire intelligent s'inscrit dans la démarche actuelle de rénovation de l'éclairage public en France et à l'étranger. Le luminaire s'appuie sur des technologies existantes (captation de mouvement, de qualité de l'air, de décibels et de consommation énergétique) pour proposer un équipement public d'éclairage utile à la récolte d'indicateurs physiques de l'espace public.

Afin de croiser l'ensemble de ces données avec les flux de circulation de l'espace public (auto, cyclo, piéton, autres), le projet envisage de développer une solution de comptage d'objets en mouvement sur l'espace public à partir d'une analyse de flux vidéo capté en direct sur la voirie.

对象的检测是研究的一个非常活跃的领域，寻求分类和定位图像或视频流的地区/区域。此区域是在两个人的十字路口：图像分类和对象的定位。事实上，在检测对象的原理如下：对于给定的图像，区域的后寻求可能包含一个对象，并为每个这些区域发现的，提取和分类的例如，使用图像分类模型。保留具有良好分类结果的原始图像的区域，并拒绝其他区域。因此，具有用于检测对象的好方法，有必要具有用于检测区域和良好的图像分类算法的高效算法。

Etat de l'art

Introduction "Détection d'objet" :

La détection d'objets est un domaine très actif de la recherche qui cherche à classer et localiser des régions/zones d'une image ou d'un flux vidéo. Ce domaine est à la croisée de deux autres : la classification d'image et la localisation d'objets. En effet, le principe de la détection d'objets est le suivant : pour une image donnée, on recherche les régions de celle-ci qui pourraient contenir un objet puis pour chacune de ces régions découvertes, on l'extrait et on la classe à l'aide d'un modèle de classification d'image par exemple. Les régions de l'image d'origine ayant de bons résultats de classification sont conservés et les autres rejetés. Ainsi, pour avoir une bonne méthode de détection d'objets, il est nécessaire d'avoir un algorithme performant de détection de régions et un bon algorithme de classification d'images.

成功的关键在于图像分类算法。自2012年ImageNet挑战的结果以来，深度学习（尤其是卷积网络）已成为解决此类问题的头号方法。
当然，物体检测研究已经包含了图像分类模型，这使得创建物体检测方法成为可能，例如SSD，t R-CNN和Faster-RCNN。

La clef de la réussite est détenue dans l'algorithme de classification d'image. Depuis les résultats du challenge ImageNet 2012, le Deep Learning (et notamment les réseaux de convolution) est devenue la méthode numéro une pour résoudre ce genre de problème. Le recherches en détection d'objets ont donc tout naturellement intégré les modèles de classification d'image, ce qui a permis de créer méthodes de détection d'objet tels que SSD,t R-CNN et Faster-RCNN.

SSD和R-CNN从不单独使用，它们与预先计算的图像分类网络相结合。已加权的网络称为模型。有Mobilenet SSD， R-CNN Inception等。

SSD et R-CNN ne sont jamais utilisés seuls, ils sont couplés avec un réseau pré-calculé de classification d'image. Un réseau auquel on a calculé les poids est appelé un modèle. Il existe ainsi des SSD Mobilenet, des R-CNN Inception, etc.

使用大的图像数据集计算图像分类网络的权重。通常，它使用的是MSCOCO数据集（通常简化为COCO）。 MSCOCO是一组超过330k的图像，包含许多不同类型的对象。

Le poids des réseaux de classification d'image a été calculé avec un gros dataset d'images. Généralement c'est avec le dataset MSCOCO (souvent réduit à COCO) qui est utilisé. MSCOCO est un ensemble de plus de 330k images contenant des dizaines de types d'objets différents.

Quelques logiciels propriétaires :

我们咨询了一些专有软件的网站提供的工具，用于检测视频流中的对象或从相机记录下来，有关于功能，它们的价格来帮助我们选择一个想法，我们的规格上的功能。

Nous avons consulté quelques sites de logiciels propriétaire qui fournissent des outils pour la détection des objets sur des flux vidéo enregistrés ou à partir d'une caméra et cela afin d'avoir une idée sur les fonctionnalités qu'ils offrent pour nous aider à choisir nos fonctionnalités sur le cahier de charge.

Camlytics :

Fonctionnalités :

行人，人群，被遗弃/被盗物品

- Détection : piéton, foule, objets abandonnés/volés
- Identification

- Classification entre piétons et voitures
- Tracking

Différents scénarios :

计算一个区域的人数

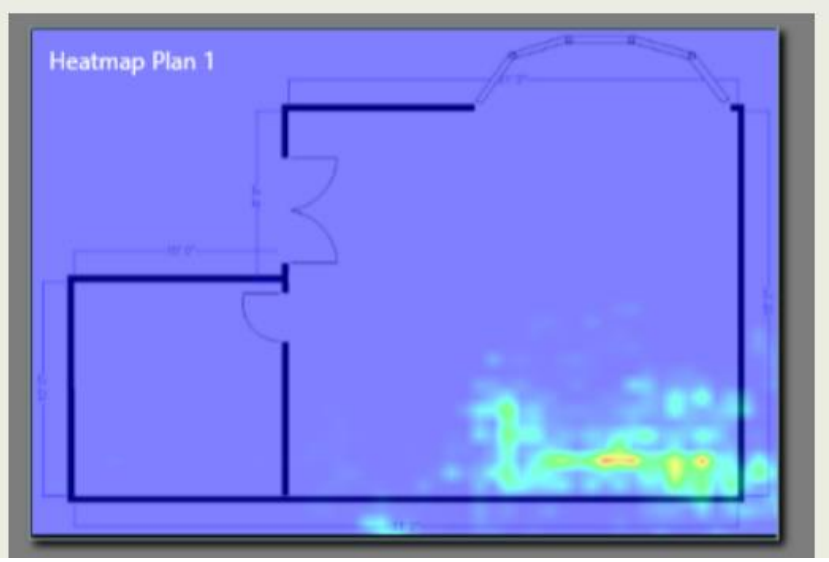
- Counting : compte le nombre de personne dans une zone
- Crowding : s'il y a un certain nombre de personne dans une zone pendant un certain temps, on déclenche une alerte
- Motion detection : si quelque chose bouge dans l'image on implémente (piéton, voiture, feux tricolore, …)
- Speeding : alerte si un objet à une vitesse au dessus d'un certain seuil
- Dwelling : si objet dans une certaine zone pendant une période de temps s'est déplacé d'un certain rayon
- Abandoned item : détecte si un objet est laissé

Name	Source	Description	Has object Id	Pedestrian / Vehicle classified
Line crossed	Line	Fired when line is crossed	Yes	Yes
Zone joined	Zone	Fired when object has joined the zone	Yes	Yes
Zone left	Zone	Fired when object has left the zone	Yes	Yes
Motion started	Zone	Indicates of motion start in the zone	No	No
Motion finished	Zone	Indicates of the end of motion un the zone	No	No
Running	Zone	Indicates running of object in the zone	Yes	Yes
Object dwell	Zone	Fired when object that has been in the zone for long enough time and has moved to enough distance	Yes	Yes
Crowd appear	Zone	Fired when many enough objects have been in the zone for long enough time	No	No
Crowd disappear	Zone	Fired when Crowd appear condition is not met anymore	No	No
Abandoned item	Zone	Fired when static object has been in the zone for long time	No	No
Object appear	Video scene	Fired when any object is created	Yes	Yes
Object disappear	Video scene	Fired when any object is deleted	Yes	Yes
Camera obstructed	Video scene	Indicates that camera has been obstructed partly or completely by light, huge object, etc.	No	No
Camera disconnected	Video scene	Fired when there are no frames from camera for long time.	No	No

Fonctionnement :

通过数据库中的图形可视化数据
完成（在这样的日期/时间在这样的地方举办的活动）

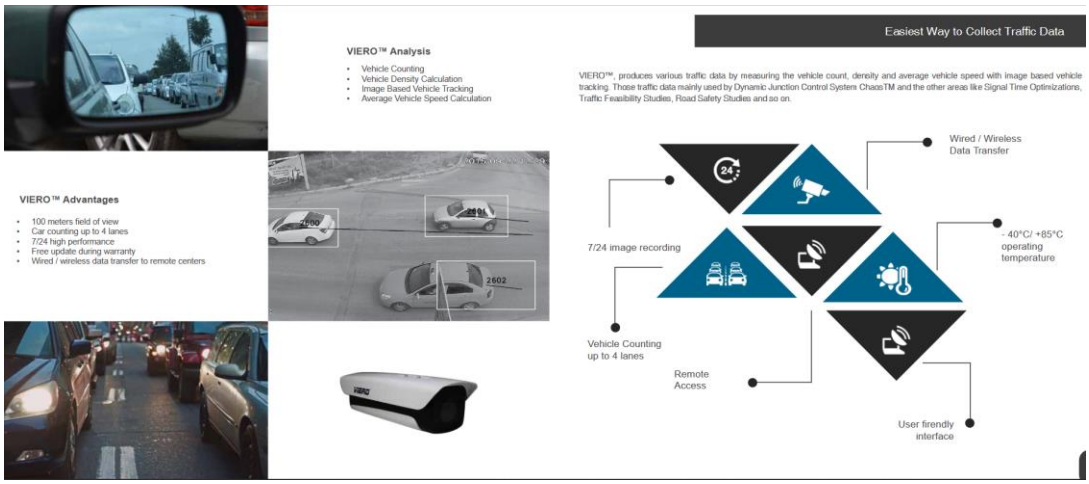
- Utilisation de lignes et zones pour la détection
- Visualisation des données par des graphiques à partir d'une base de données complète (Events à tel endroit à telle date/heure)
- Création de HeatMaps, carte de zone qui affiche suivant différentes couleurs la concentration de déplacements :

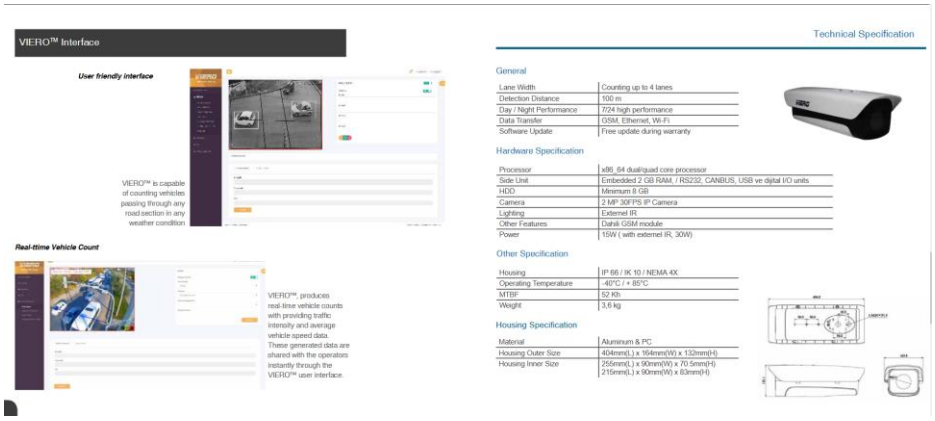


ISSD

Viero : Comptage de voiture

- Compte le nombre de véhicule jusqu'à 4 voies
- Capte l'occupation de la route
- Track les véhicule
- Capte la vitesse des véhicules





Quelques bibliothèques de machine learning et détection d'objet :

Tensorflow:

TensorFlow est un outil open source d'apprentissage automatique développé par Google. Le code source a été ouvert le 9 novembre 2015 par Google et publié sous licence Apache.
Il est fondé sur l'infrastructure DistBelief, initiée par Google en 2011, et est doté d'une interface pour Python et Julia.
TensorFlow est l'un des outils les plus utilisés en IA dans le domaine de l'apprentissage machine.

Pytorch:

PyTorch est une bibliothèque logicielle Python open source d'apprentissage machine qui s'appuie sur Torch (en) développée par Facebook.
PyTorch permet d'effectuer les calculs tensoriels nécessaires notamment pour l'apprentissage profond (deep learning). Ces calculs sont optimisés et effectués soit par le processeur (CPU) soit, lorsque c'est possible, par un processeur graphique (GPU) supportant CUDA. Il est issu des équipes \de recherche de Facebook, et avant cela de Ronan Collobert dans l'équipe de Samy Bengio9 à l'IDIAP.
PyTorch est dérivé d'un logiciel antérieur, Torch, qui s'utilisait avec le langage Lua. PyTorch est indépendant de Lua et se programme en Python.

PyTorch permet de :

- Manipuler des tenseurs (tableaux multidimensionnels), de les échanger facilement avec Numpy et d'effectuer des calculs efficaces sur CPU ou GPU (par exemple, des produites de matrices ou des convolutions);
- Calculer des gradients pour appliquer facilement des algorithmes d'optimisation par descente de gradient. PyTorch utilise la bibliothèque autograd10.

Keras:

Keras est une bibliothèque open source écrite en python2 et permettant d'interagir avec les algorithmes de réseaux de neurones profonds et de machine learning, notamment Tensorflow et Theano. Elle a été initialement écrite par François Chollet

Recherches :在我们的主题文档中，我们发现了几种用于对象检测的科学论文和算法。将附上这些文件的链接。

Dans notre documentation sur sujet nous avons trouvé plusieurs papiers scientifiques et différents algorithmes dans le cadre de la détection d'objet. des liens vers ces documents seront mis en annexe.

Comparaison entre Tensorflow et Pytorch:

尽管Tensorflow和PyTorch都是开源的，但它们是由两个不同的供应商创建的。 Tensorflow基于Theano， 由Google开发， 而PyTorch基于Torch， 由Facebook开发。
Bien que Tensorflow et PyTorch soient tous deux open-source, ils ont été créés par deux constructeurs différents. Tensorflow est basé sur Theano et a été développé par Google, tandis que PyTorch est basé sur Torch et a été développé par Facebook.

Graphiques de calcul :

La différence la plus importante entre les deux réside dans la manière dont ces biblothèques définissent les graphiques de calcul. Alors que Tensorflow crée un graphique statique, PyTorch croit en un graphique dynamique.

两者之间最重要的区别在于这些库定义计算图的方式。虽然Tensorflow创建了静态图表，但PyTorch相信动态图表。

Facilité d'apprentissage :

Tensorflow est plus compliqué à apprendre que PyTorch. PyTorch est plus pythonique et la construction de modèles semble plus intuitive. Par contre, pour utiliser Tensorflow, vous devez en apprendre un peu plus sur son fonctionnement (sessions, espaces réservés, etc.) et il devient donc un peu plus difficile d'apprendre Tensorflow que PyTorch.

Communauté :

Tensorflow a derrière elle une communauté beaucoup plus grande que PyTorch. Cela signifie qu'il devient plus facile de trouver des ressources pour apprendre Tensorflow et, également, pour trouver des solutions à vos problèmes. En outre, de nombreux tutoriels et MOOC couvrent Tensorflow au lieu d'utiliser PyTorch. En effet, PyTorch est un framework relativement nouveau par rapport à Tensorflow. Donc, en termes de ressources, vous trouverez beaucoup plus de contenu sur Tensorflow que PyTorch.

Tensorflow比PyTorch拥有更大的社区。这意味着更容易找到资源来学习Tensorflow并找到问题的解决方案。此外，许多教程和MOOC涵盖了Tensorflow， 而不是使用PyTorch。实际上，与Tensorflow相比， PyTorch是一个相对较新的框架。因此，就资源而言，你会发现Tensorflow上的内容比PyTorch要多得多。

Notre choix :

Au début nous avons pensé à répartir le projet en deux parties :

- Comptage d’objets
- Détection d’objet.

但是通过查看检测对象的API，可以在检测到它们后直接对它们进行计数。为此，我们最终选择将这两部分直接结合起来。

Mais en regardant des API qui permettent de détecter des objets, il est possible de les compter directement après les avoir détectés. Pour cela nous avons choisi finalement de directement combiner ces deux parties.

虽然这是最难学的，但鉴于周围的大型社区，我们尚未选择使用它。实际上，我们在互联网上发现了几个API，用于检测和计算使用Tensorflow库的视频流中的对象。由于我们有足够的时间来完成我们的项目，我们认为依靠现有的API来创建我们的解决方案，而不是从头开始。

Même s’il est le plus difficile à apprendre, nous avons pourtant opter pour son utilisation vu la grande communauté qui se trouve autour de lui. En effet, nous trouvons sur internet plusieurs API de détection et comptage d’objet dans un flux vidéo qui utilisent la bibliothèque Tensorflow. Et puisque nous avons une durée assez limitée pour faire notre projet, nous avons pensé à se baser sur une API déjà existante pour créer notre solution au lieu de commencer from scratchs.

Quelques API de détection et comptage d’objet utilisant Tensorflow :

Tensorflow object detection API:
https://github.com/tensorflow/models/tree/master/research/object_detection
YOLO with Tensorflow:
https://github.com/thtrieu/darkflow
Tensorflow object counting API :
https://github.com/ahmetozlu/tensorflow_object_counting_api
How To Train an Object Detection Classifier for Multiple Objects Using TensorFlow (GPU) on Windows 10
https://github.com/EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10

Cahier des charges

La solution à développer est une application qui permet de détecter des objets dans un flux vidéo.

Entrées de la solution :

- Vidéo temps réel
- Vidéo enregistrée

Objets à détecter :

- Personnes
- Voitures
- Vélos.

Sorties de la solution :

Fichier log de type csv contenant des informations sur les objets détectés :

- Date et heure
- Type d'objet
- Vitesse

Conception et développement

Configuration de l’environnement de travail :

Première configuration :

Étapes d'installation :

Installation de driver de Nvidea

```
$ sudo apt update
$ sudo ubuntu-drivers autoinstall
$ sudo reboot
$ sudo prime-select nvidia
$ sudo reboot
```

Installation de anaconda

https://repo.anaconda.com/archive/Anaconda3-5.2.0-Linux-x86_64.sh

Création d’un environnement virtuel (recommandée)

```
$ conda create -n envTF python=x.x
```
(Python 3.3+ recommandé)

(Si utilisation d’un environnement virtuel, le reste des étapes sera fait à partie de l'environnement virtuel)

Connexion à l'environnement créé

```
$ source activate envTF
```

Installation de python et pip

```
$ sudo apt-get install python3-pip python3-dev # for Python 3.n
```

Installation de opencv

```
$ conda search -c conda-forge --spec 'opencv=3*'
```
(Pour voir la version existante d’OpenCV sur Internet)

```
$ conda install -c conda-forge opencv
```
(Installer OpenCV et l’environnement)

Installation de tensorflow

Pour connaître la procédure d'installation de Tensorflow, suivez les instructions d'installation de Tensorflow :

https://www.tensorflow.org/install/

Un utilisateur typique peut installer Tensorflow à l'aide d'une des commandes suivantes :

```
$ conda install tensorflow-gpu
```

Ou

```
$ conda install tensorflow
```

Problème avec la configuration précédente :

Nous avons passé beaucoup de temps sur la partie d'installations ci-dessus. Nous avons essayé des intallation sur différents OS, (windows, ubuntu, mac). Avant d'arriver à quelques installations fonctionnelles.
Avec windows, nous avons eu des problèmes avec les variables d'environnement, en utilisant ubuntu ce problème ne paraît pas, mais il y avait à chaque fois des packages manquants ou des configuration incompatibles (problèmes de version).

Installation de l'API Object Detection:

Pour utiliser le projet "VEHICLE DETECTION, TRACKING AND COUNTING", il est indispensable d'utiliser d'installer l'API Object Detection.

Ajout des modèles :

Ajouter les modèles entraînés de tensorau tensorflow tensorflow
<path_to_tensorflow>/models
Lien vers des modèles : https://github.com/tensorflow/models
(On peut utiliser git clone)

Ajout d'autres librairies :

```
$ sudo apt-get install protobuf-compiler python-pil python-lxml python-tk
$ pip install --user Cython
$ pip install --user contextlib2
$ pip install --user jupyter
$ pip install --user matplotlib
```

Installation du coco API :

(Ajout du dossier pycocotools de coco API au dossier research de models)

```
$ git clone https://github.com/cocodataset/cocoapi.git
$ cd cocoapi/PythonAPI
$ make
$ cp -r pycocotools <path_to_tensorflow>/models/research/
```

Compilation protobuf:

(à partir de tensorflow/models/research/)
$ `protoc object_detection/protos/*.proto --python_out=.`

Ajout des librairies à pythonpath:

(à partir de tensorflow/models/research/)
export PYTHONPATH=$PYTHONPATH:`pwd`:`pwd`/slim

Tester l'installation
$ `python object_detection/builders/model_builder_test.py`

Conception de notre solution :

Nous avons choisi de créer notre solution en nous basant sur des projets déjà existants : Dans notre réalisation nous avons travaillé sur deux projets, le projet total Object_Counting_API ainsi que l'exemple de projet qui qui utlise Tensorflow Object Counting

Object_Counting_API	https://github.com/tensorflow/models/tree/master/research/object_detection
Projet utilisant Tensorflow Object Counting API (vehicule counting)	https://github.com/ahmetozlu/vehicle_counting_tensorflow

Et nous avons procédé à plusieurs modifications afin de personnaliser notre solution et l'adapter à notre cahier de charge.

Test de la solution existante :

En exécutant une de ces commandes afin de tester le projet :
$ `python3 vehicle_detection_main.py`

Ou

$ `python vehicle_detection_main.py`

On tombe sur l'erreur suivante :

```
Traceback (most recent call last):
  File "vehicle_detection_main.py", line 39, in <module>
```

```
    raise ImportError('Please upgrade your tensorflow installation to
v1.4.* or later!'
ImportError: Please upgrade your tensorflow installation to v1.4.* or
later!
```

Or nous avons une version supérieur de tensorflow!!
Raison : On a tensorflow version > 1.4.0 (par exemple 1.10.0), mais il pense “ 1.10.0 < 1.4.0 ” avec une string dans la comparaison.

Il faut changer cette comparaison dans le code de “vehicle_detection_main.py” :

```
if tf.__version__ < '1.4.0':
  raise ImportError('Please upgrade your tensorflow installation to v1.4.*
or later!')(((((((
```

====>

```
MIN_TF_VERSION_MAJOR = 1
MIN_TF_VERSION_MINOR = 9
tf_ver = [int(s) for s in tf.__version__.split('.')]

if tf_ver[0] < MIN_TF_VERSION_MAJOR or (tf_ver[0] == MIN_TF_VERSION_MAJOR
and tf_ver[1] < MIN_TF_VERSION_MINOR):
  raise ImportError('Please upgrade your tensorflow installation to
v{0}.{1}.* or later!'.format(MIN_TF_VERSION_MAJOR, MIN_TF_VERSION_MINOR))
```

Réalisation de notre solution :

Projet complet :

Nous intégrons toutes les fonctionnalités nécessaires à un fichier python applée “real_time_counting.py”. Nous pouvons utiliser ce fichier pour la détection vidéo d'entrée ou la détection en temps réel.
Nous mettons « input_video » égale nulle pour utiliser le modèle ssd_mobilenet_v1_fpn_coco. Si vous voulez utiliser l’autre modèle, vous changez le nom du modèle dans la fonction « backbone.set_model »

Voici la comparaison des modèles différents :

Model name	Speed (ms)	COCO mAP[^1]	Outputs
ssd_mobilenet_v1_coco	30	21	Boxes
ssd_mobilenet_v1_0.75_depth_coco ☆	26	18	Boxes
ssd_mobilenet_v1_quantized_coco ☆	29	18	Boxes
ssd_mobilenet_v1_0.75_depth_quantized_coco ☆	29	16	Boxes
ssd_mobilenet_v1_ppn_coco ☆	26	20	Boxes
ssd_mobilenet_v1_fpn_coco ☆	56	32	Boxes
ssd_resnet_50_fpn_coco ☆	76	35	Boxes
ssd_mobilenet_v2_coco	31	22	Boxes
ssd_mobilenet_v2_quantized_coco	29	22	Boxes
ssdlite_mobilenet_v2_coco	27	22	Boxes
ssd_inception_v2_coco	42	24	Boxes
faster_rcnn_inception_v2_coco	58	28	Boxes
faster_rcnn_resnet50_coco	89	30	Boxes
faster_rcnn_resnet50_lowproposals_coco	64		Boxes
rfcn_resnet101_coco	92	30	Boxes
faster_rcnn_resnet101_coco	106	32	Boxes
faster_rcnn_resnet101_lowproposals_coco	82		Boxes
faster_rcnn_inception_resnet_v2_atrous_coco	620	37	Boxes
faster_rcnn_inception_resnet_v2_atrous_lowproposals_coco	241		Boxes
faster_rcnn_nas	1833	43	Boxes

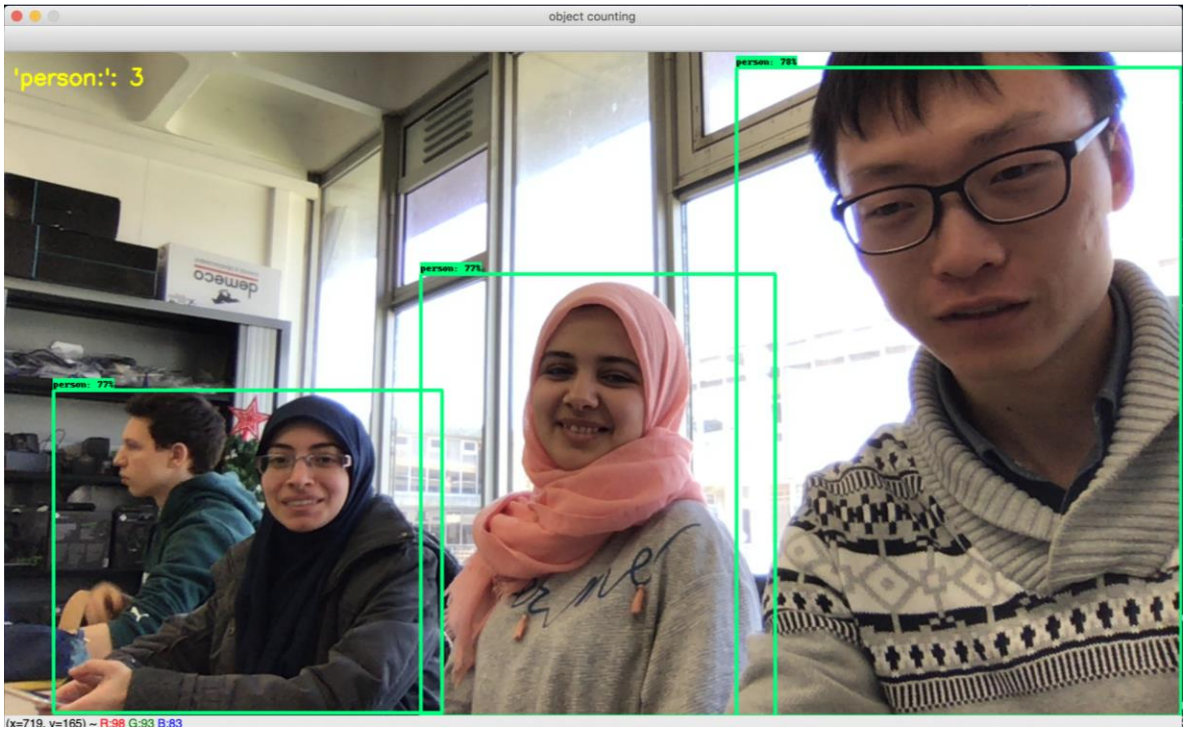
Vous pouvez aussi le trouver sur le lien:
https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md

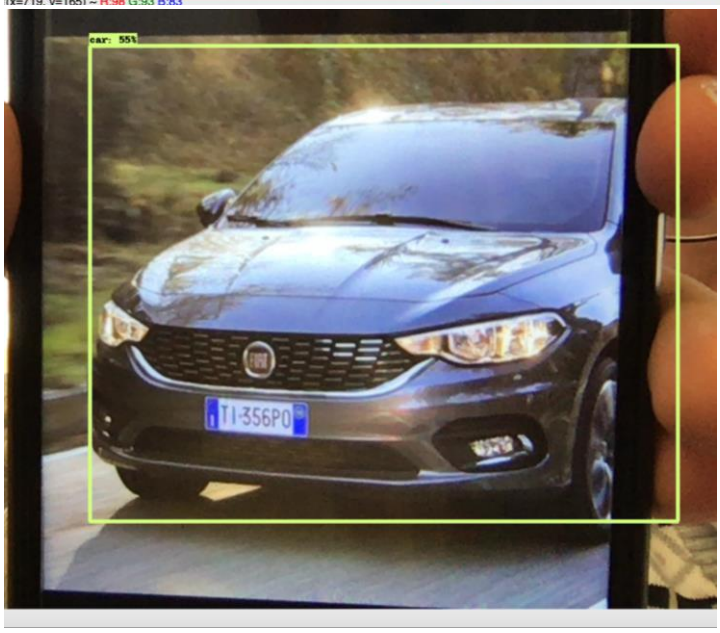
Quand l'utilisateur démarre le programme, il a le choix entre utiliser une caméra comme sa webcam ou d'utiliser une vidéo déjà enregistrée. Pour se faire il doit lancer le script python en ajoutant soit use_cam soit use_videofile en argument :

python vehicle_detection_main.py use_videofile

Détection Temps-Réel :

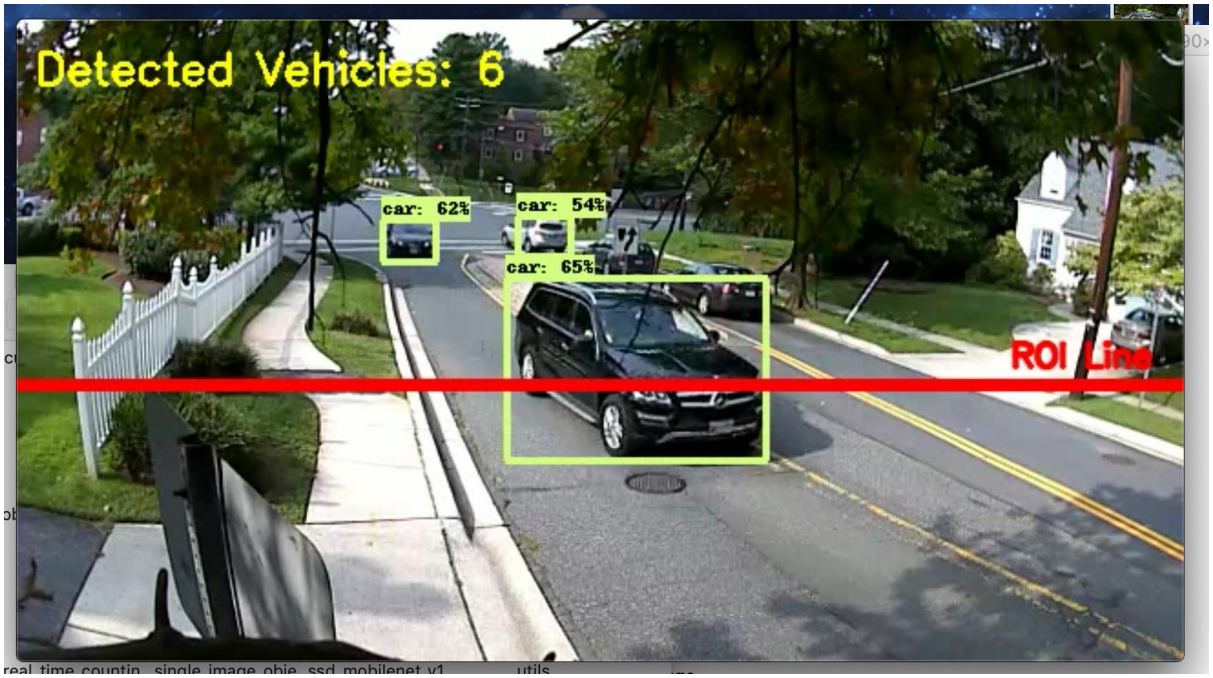
Nous mettons « input_video » égale nulle et activons la fonction « object_counting ».
Dans la fonction « object_counting », nous changeons « VideoCapture(0) » pour ouvrir la caméra.
Ensuite, il va détecter les voitures, les personnes, etc par notre caméra.



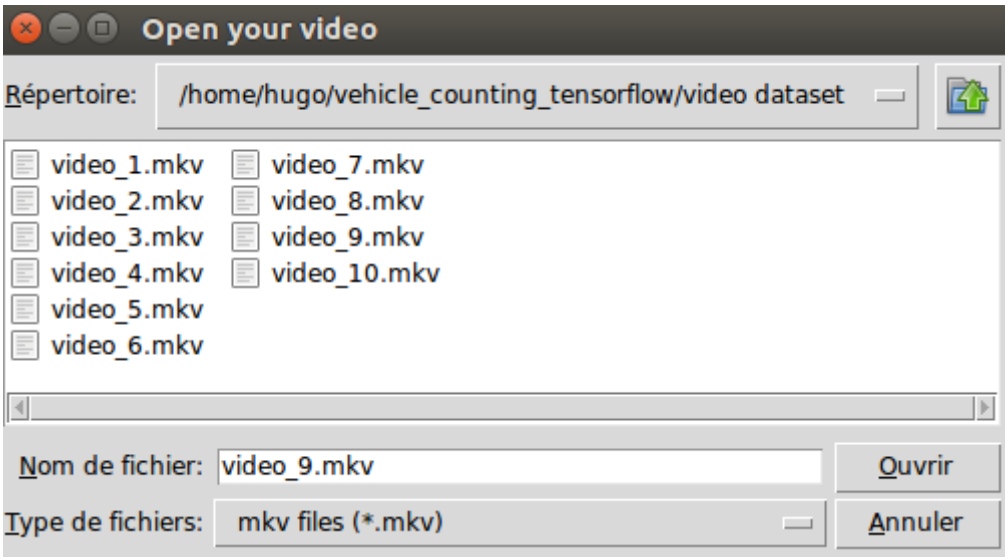


Vidéo détection :

Nous mettons le path de notre vidéo et activons la fonction « object_counting ». Dans la fonction « object_counting », nous changeons « VideoCapture(input_video) » pour obtenir et traiter la vidéo.
Ensuite, il va traiter la vidéo et nous donner un vidéo de sortie appelé « the_output.avi ». On peut voir la détection dans la vidéo.

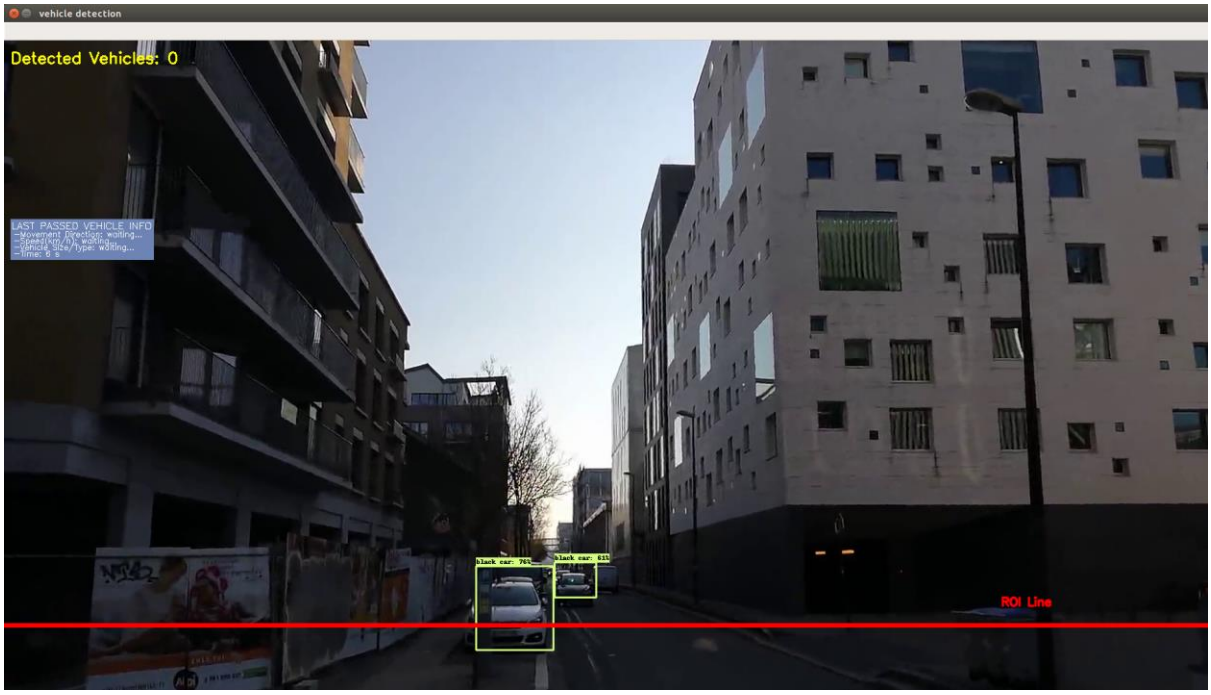


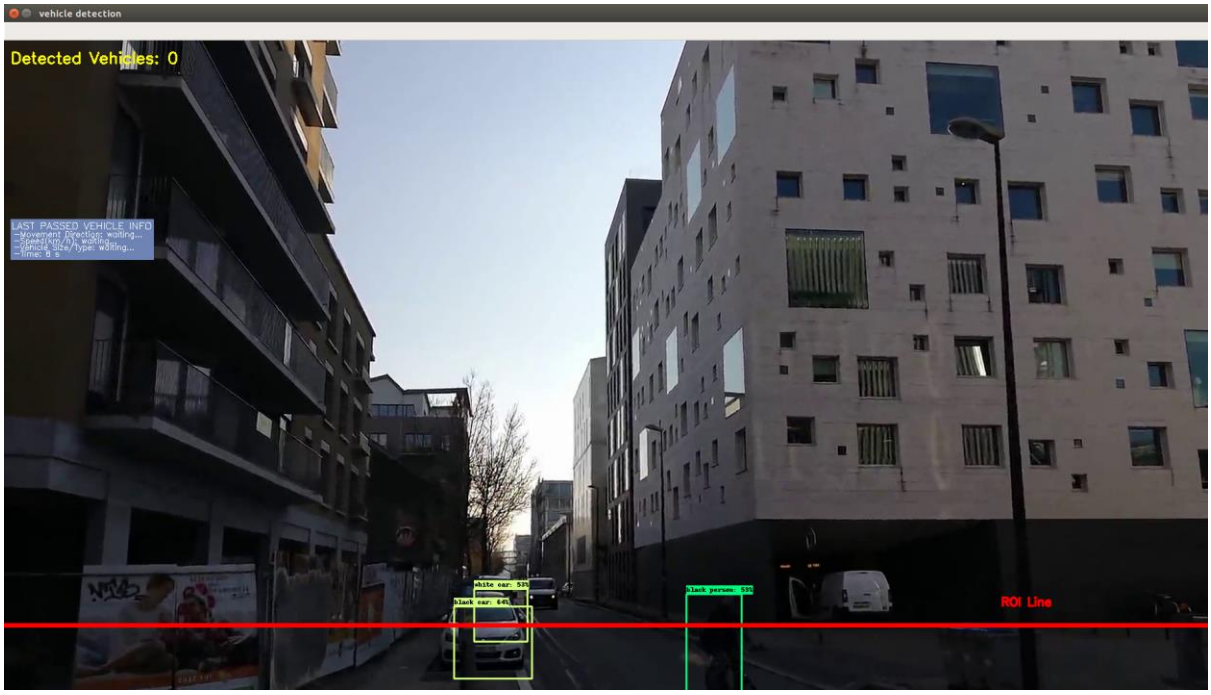
Nous avons intégré, grâce à Tkinter et sa fonction askopenfilename, un outil de recherche de documents. L'utilisateur peut désormais rechercher la vidéo de son choix plus facilement.



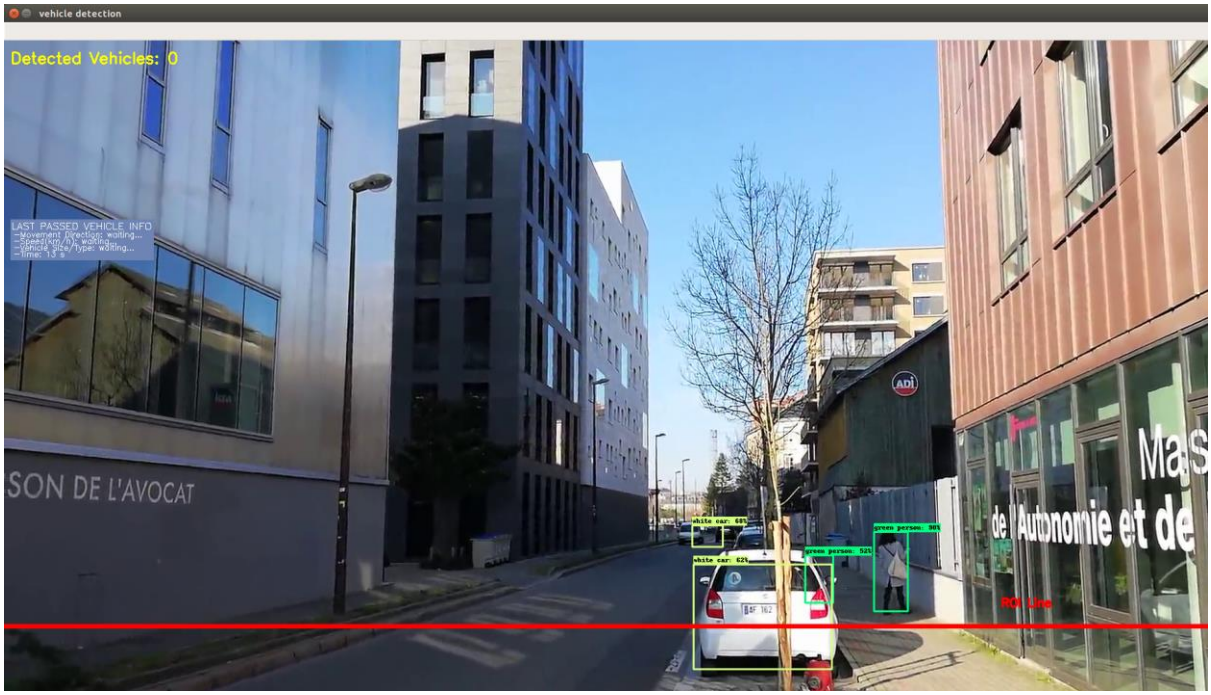
Nous avons pris quelques vidéos dans la rue La Noue Bras de Fer. Les premières sont plutôt sombres mais le programme arrive quand même à détecter les véhicules. Les prises de vues ont été prises à 2m de hauteur et il faudrait tester le programme avec des vidéos prises à hauteur de lampadaire pour voir si la détection s'opère aussi bien. En effet le modèle que nous avons intégrés dans notre programme ne s'est peut être pas entraîné sur des voitures vues de dessus.

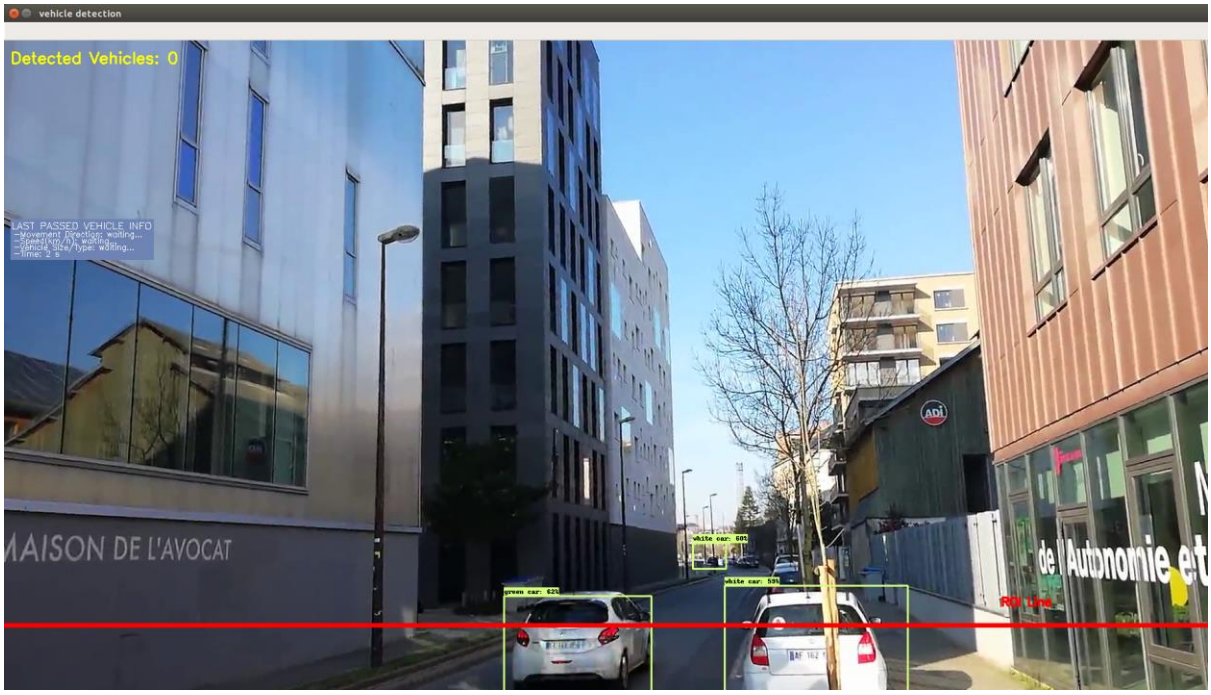
Voici les résultats sur nos vidéos prises dans la rue :



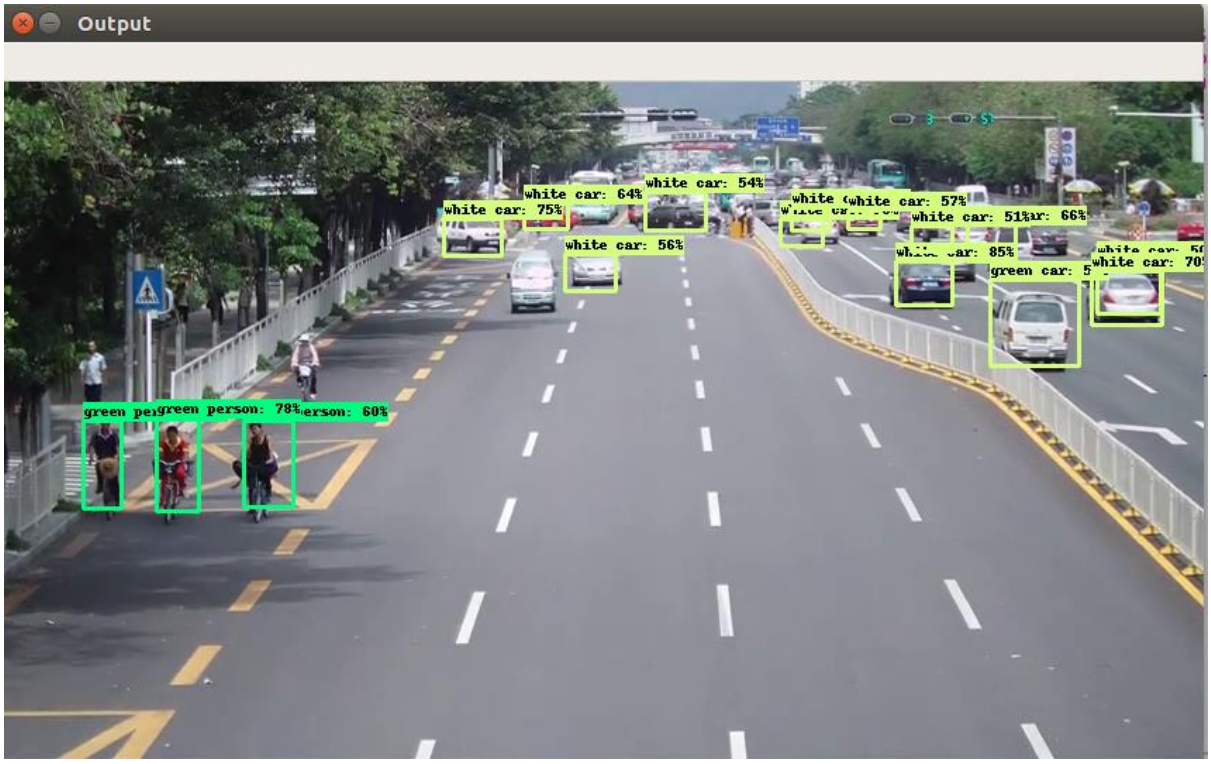

vehicle detection
Detected Vehicles: 0
LAST PASSED VEHICLE INFO
ROI Line


vehicle detection
Detected Vehicles: 0
LAST PASSED VEHICLE INFO
ROI Line


vehicle detection
Detected Vehicles: 0
LAST PASSED VEHICLE INFO
-Movement Direction: waiting...
-Speed(km/h): waiting...
-Vehicle Size/Type: waiting...
-time: 13 s
SON DE L'AVOCAT
white car: 68%
white car: 62%
green person: 30%
green person: 52%
ROI Line


vehicle detection
Detected Vehicles: 0
LAST PASSED VEHICLE INFO
-Movement Direction: waiting...
-Speed(km/h): waiting...
-Vehicle Size/Type: waiting...
-time: 2 s
MAISON DE L'AVOCAT
white car: 60%
white car: 59%
green car: 62%
ROI Line

Nous avons aussi exécuté le programme sur des vidéos de surveillance du trafic :



On remarque que l'algorithme a du mal à identifier les vélos même si ceux-ci sont inscrits dans la liste des catégories d'intérêt. Il retournera le plus souvent une personne à la place.

L'algorithme semble bien détecter les objets qui nous intéressent et cela rapidement. Le temps d'exécution de la détection augmente beaucoup quand la qualité de l'image est plus grande (plus de pixels à traiter). Nous avons dû diminuer la qualité de nos vidéos pour éviter des temps de calculs trop longs.

Object Counting API :

Personnalisation de la solution :

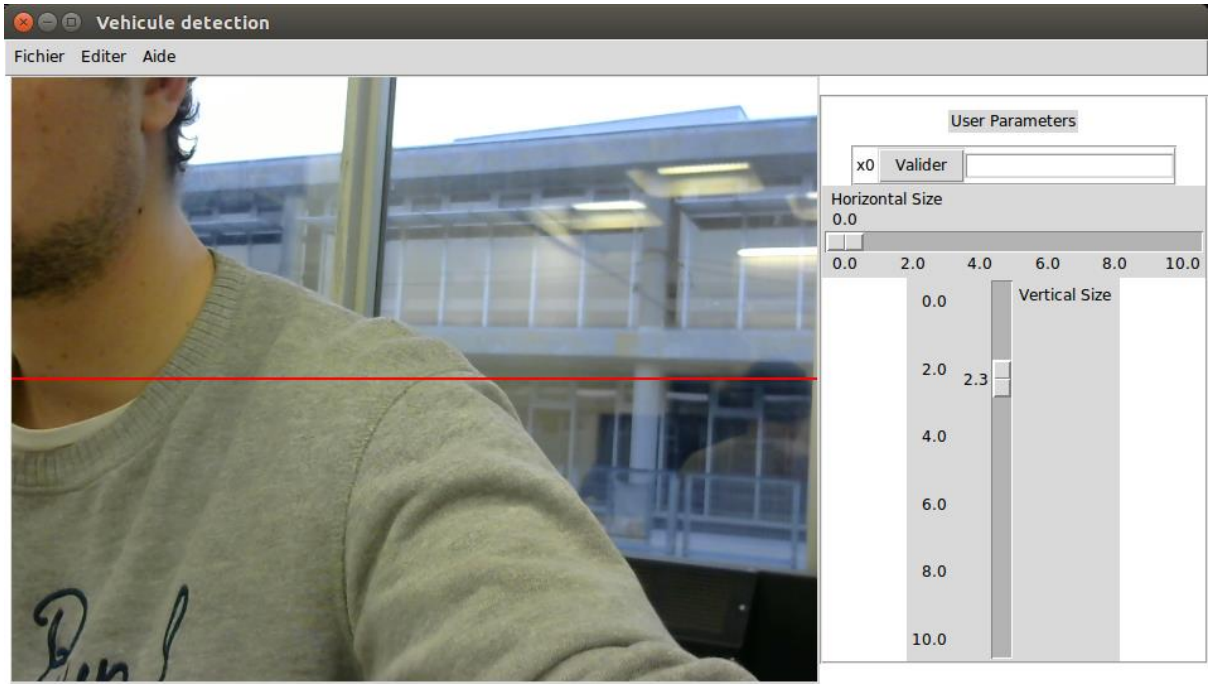
La solution existante ne prend en compte que la détection des voitures, en plus les logs ne correspondent pas totalement à ce qu'on veut faire. Alors nous avons procédé à quelques modifications :

1. Ajout d'un dossier video dataset : on y range toutes les vidéos que l'on veut tester
2. Modification du fichier utils/visualization_utils.py :
   Dans la fonction visualize_boxes_and_labels_on_image_array on ajoute les catégories qui nous intéresse en ajoutant : *or ("person" in display_str_list[0])* dans un if. Initialement le programme ne s'intéressait qu'aux catégories car et truck. Nous ajoutons les catégories person, bicycle et motorcycle.
3. Ajout du paramètre temps dans les fonctions pour ensuite pouvoir stocker quand l'objet en question a été détecté. Les paramètres catégorie d'objet, couleur, direction, vitesse et temps sont stockés dans un fichier excel : traffic_measurement.csv à la racine du dossier. Nous avons au final enlevé le paramètre couleur car l'algorithme ne le détermine pas bien et il n'est pas nécessaire à notre apllication.

4. Modification du *vehicule_detection_main.py* :
   - Ajout du temps en seconde dans l'affichage
   - Choix entre vidéo enregistrée ou flux vidéo
   - Sélection de la vidéo
   - Adaptation de la ligne ROI à la taille de l'image
   - Création d'une fenêtre graphique pour mieux communiquer et interagir avec l'utilisateur. Grâce à Tkinter, l'utilisateur pourra renseigner s'il décide utiliser le flux vidéo d'une caméra ou des vidéos déjà enregistrées. Il pourra aussi choisir la hauteur de la ligne de détection (ROI line) en fonction d'où se situe la route sur la vidéo (Non terminé).

Améliorations possibles :

Nous avons pensé à intégrer une interface graphique pour l'utilisateur. Il pourrait ainsi changer les valeurs du script pendant l'exécution et observer ces changements directement. Nous avons commencé à construire cette interface mais faute de temps nous n'avons pas pu la terminer.



Une autre piste d'amélioration serait d'entraîner notre propre modèle pour qu'il soit plus spécialiste de véhicules comme les voitures, vélos et motos mais il nécessiterait beaucoup d'heures d'entraînement avec beaucoup de puissance de calcul pour arriver à un résultat pas forcément plus qualitatif.

Nous aurions voulu implémenter une sorte de drag and drop pour la ligne ROI pour permettre à l'utilisateur de bien placer sa ou ses lignes d'intérêt. Etant donné que le flux vidéo proviendra d'une camera fixe, cette fonction n'est pas très importante puisqu'il suffit de placer une seule fois la ligne au bon endroit de l'image.

Conclusion :

Ce projet était pour nous une occasion pour la découverte des outils de détection d'objet et du machine learning.
Au début nous étions un peu perdus puisque pour certains d'entre nous c'est la première fois qu'ils travaillent sur ce genre de projets.Nous avons prévus au début de tester plusieurs bibliothèques avant de faire le choix de la bibliothèque à utiliser, mais en commençons nos expérimentation avec Tensorflow nous nous sommes rendus compte de la complexité des installations et configuration des environnements et packages nécessaires, et nous avions décidé vu le temps limité dont nous disposons de se limiter à l'utilisation de Tensorflow pour le présent.

Finalement, nous sommes arrivés à faire marcher des modèles et ajouter nos propres modifications pour construire notre solution. Nous aurions bien aimé avoir déjà des vidéos prises depuis le luminaire pour se rapprocher le plus possible de l'utilisation prévu.

Annexe :

TrackNet: Simultaneous Object Detection and Tracking and Its Application in Traffic Video Analysis
https://arxiv.org/abs/1902.01466

Comparison of Some Motion Detection Methods in cases of Single and Multiple Moving Objects : Octobre 2012
https://www.researchgate.net/publication/306357420_Comparison_of_Some_Motion_Detection_Methods_in_cases_of_Single_and_Multiple_Moving_Objects

http://vision.middlebury.edu/flow/eval/results/results-n1.php

http://vision.middlebury.edu/flow/eval/

Exemple de librairie : Background Separation Librairie
https://github.com/andrewssobral/bgslibrary/wiki/Algorithms-benchmark

Thèse du MIT : Juin 2009
https://people.csail.mit.edu/celiu/Thesis/CePhDThesis.pdf

Benchmarking de méthodes de détection de mouvement : Juillet 2014
https://orbi.uliege.be/bitstream/2268/157177/1/Jodoin2014Overview.pdf
(regarder directement la conclusion)

Comparing Different Visual Motion Detection Algorithms : Avril 2014
http://www.supercomputingchallenge.org/13-14/finalreports/143.pdf
(Code C++ fourni !)

Algo complet : Mars 2007
https://www.codeproject.com/Articles/10248/Motion-Detection-Algorithms

Benchmark sur les méthodes de détection de mouvement de caméras non-fixes : Mars 2018
https://hal.archives-ouvertes.fr/hal-01724322/document

An Efficient Approach for Object Detection and Tracking of Objects in a Video with Variable Background : May 2017
https://arxiv.org/pdf/1706.02672.pdf

Optical Flow Based Real-time Moving Object Detection in Unconstrained Scenes : Juillet 2018
https://arxiv.org/pdf/1807.04890.pdf

Comparaison de 5 algo de détection pour la vidéosurveillance : date >= 2005

https://homepages.inf.ed.ac.uk/rbf/CAVIAR/PAPERS/Nascimento.pdf

Comparaison d'algo de détection : date >=2012
http://www-labs.iro.umontreal.ca/~mignotte/IFT6150/Articles/ChangeDetectionBenchmark.pdf

Noise-robust motion detection for low-light videos : 2017
https://lib.ugent.be/fulltxt/RUG01/002/367/337/RUG01-002367337_2017_0001_AC.pdf

webographie:
https://www.actuia.com/contribution/jeancharlesrisch/segmentation-et-detection-dobjets-en-temps-reel-avec-tensorflow/

Script de lancement : (à exécuter à la racine)

```
#!/bin/bash
source venv/bin/activate
cd vehicle_counting_tensorflow
python vehicle_detection_main_graphique.py use_videofile    #use_cam or use_videofile
```