**introduction générale**

(parler sur l’invention de voiture et l’historique des accidents qui se sont passés depuis cette invention )

Au fil des temps, les accidents routiers ne cessent de mettre en danger la vie des passagers ainsi que celle des piétons,surtout avec l'augmentation constante du trafic routier.

Les causes se cachant derrière la survenance d'un accident sont multiples, et parfois très difficiles à identifier et à expliquer. Parmi ces causes, on distingue la vitesse qui ne fait pas qu’aggraver les accidents de la route,mais elle les provoque. En tenant compte de leurs graves conséquences, les excès de vitesse sont des infractions lourdement réprimées par le Code de la route.

Afin d'améliorer la sécurité et diminuer les risques des accidents, on peut agir sur trois facteurs : l'infrastructure routière, la conduite et le véhicule.

Dans ce cadre-là, plusieurs mesures de sécurité ont été mises en place telles que des limitations de vitesse, des ralentisseurs et des radars.

En revanche, la gravité de cette situation est loin d’être contrôlée suite à l’inconscience de certains conducteurs irresponsables. D’où la nécessité d’adapter de nouvelles mesures de sécurité ainsi que des nouveaux outils technologiques pour inciter ces conducteurs à la prudence. Parmi eux, des drones, des radars multi-fonctions et des radars embarqués.

En effet, certains pays tels que la France disposent déjà ce genre d’outils ( on peut pas se permettre ces outils car ils sont trop chers, d’où le besoin de notre projet)

**Etat de l’art :**

* types de radars + les technologies utilisées
* Objectif de notre projet: Implémentation d’un radar de contrôle routier mobile (intégré dans une voiture civile banalisée) // dans l’intro générale.
* pourquoi le machine learning et pas le deep learning (un système embarqué sur des appareils android sera très lourd si on l'implémente avec le deep learning ) // pas dans l’état de l’art (on parlera de ça lorsqu’on utilisera LinearSVC)

Capteurs des données vidéo, accéléromètre et GPS : application mobile (Android Studio).

Traitement des tâches complexes (détection, suivi, calcule de vitesse) : application desktop (Python).

après l’état de l’art : parler des solutions possibles ( mentionner le capteur x\_.io) (utiliser le code de ce capteur dans notre tél)

présenter notre solution.

4 mois : 7%

4 Juillet

il nous reste 5 mois => 93%

**Objectifs :**

- Détection des voitures (entraînement et classification)

- calcul des distances (la suivie, Optical flow, stéréo….)

- Connaître la vitesse de la voiture cible (en fonction de la vitesse de la voiture radar).

- Connaître la limitation de vitesse dans la route actuelle.

- Détecter les infractions dans les deux sens.

**Tâches à faire :**

1- Récupérer les données à partir du GPS et de l’accéléromètre.\*\*\* a faire le plus tôt possible

2- Transférer les données du mobile au desktop.

3- entrainement et test du classificateur (HOG, SVM) \*\*\* Done

4- détection en temps réel \*\*\* trop de fausses positives

5- suivi d’objet. \*\*\* a faire le plus tôt possible

6- calcul des distances (reconstruction 3D : Optical flow, stéréo….)

7- calcule de vitesse (filtre de Kalman)

**Webographie :**

<https://www.securite-routiere.gouv.fr/radars/differents-types-de-radars/radars-mobiles/voiture-radar>

<https://www.liberte-algerie.com/actualite/des-radars-embarques-sur-des-vehicules-banalises-221220>

exemple d’utilisation du filtre de Kalman :

<https://lucidar.me/fr/kalman-filters/example-of-kalman-filter/>

https://www.youtube.com/watch?v=mwn8xhgNpFY

récupération des données à partir de l’accéléromètre ( android ) :

<https://www.techrepublic.com/blog/software-engineer/a-quick-tutorial-on-coding-androids-accelerometer/>

tutoriel socket pour l’envoi des données en temps réel :

<https://www.pubnub.com/blog/socket-programming-in-python-client-server-p2p/>

dataset d’images de voitures ( training + testing)

<https://www.gti.ssr.upm.es/data/Vehicle_database.html>

<http://www.gti.ssr.upm.es/~jal/Database/CaltechGraz.rar>

<https://www.kaggle.com/jutrera/stanford-car-dataset-by-classes-folder> (meilleure qualité)

partie classification (après training)

<https://pythonspot.com/car-tracking-with-cascades/>

partie rédaction

<https://blog.deloitte.fr/computer-vision-une-revolution-qui-ne-fait-que-commencer/>

Optical flow ( Lucas Kanade )

<https://docs.opencv.org/3.4/d4/dee/tutorial_optical_flow.html>

<https://stackoverrun.com/fr/q/10215100> optical flow

**Remarques :**

-éviter la connexion sans fil et s’appuyer sur la connexion avec câble USB (en mode debug).

-ne pas utiliser le deep learning car elle ne sera pas adaptable sur tablette plus tard.

-utiliser qlq parties du code du PFE précédant.

-comment trouver l’information en 3D ? dans le cas d’une caméra fixe : ( on doit se concentrer sur ça pour l’instant ). ( reconstruction 3D ) kima n7abo

réalité augmentée( 1 caméra ) , stéréovision ( 2 caméras ou 1 caméra avec 2 miroirs), optical flow (technique de Lucas kanade ou horn and chunk).

-si on veut utiliser plusieurs devices pour l'acquisition des données on doit les synchroniser.

-de préférence on utilise le téléphone.

partie vision pure (voiture fixe ) / partie capteur ( voiture en mouvement )

-éviter d’appliquer la détection sur la vidéo image par image.

à finir avant le S2 :

-le code du classificateur et la détection

-le code de correspondance

-faire un partage de github