**COMPTES RENDUS INDIVIDUELS PROJET CODEV**

*KHELLA Éric*

**Compte rendu individuel hebdomadaire (semaine du 05/02/24 au 11/02/24)**

Premier point lors de la réunion du 07/02. Explication des objectifs attendus, présentation du projet par l’intervenant.

Création d’un chat de discussion avec les membres de l’équipe.

**Compte rendu individuel hebdomadaire (semaine du 12/02/24 au 18/02/24)**

Semaine calme ; premiers pas vers les débuts concrets

Relecture de mon ancien projet d’analyse vidéo en Python ; relecture rapide de la documentation d’OpenCV

**Compte rendu individuel hebdomadaire (semaine du 19/02/24 au 25/02/24)**

Première prise en main du projet, quelques interrogations évoquées par mail

En termes de code :

* Création du main, des principales méthodes et classes
  + Import des différents modules et début de prise en main concrète du projet
  + Création d’une classe de messages (erreurs, informations)
  + Récupération d’une vidéo motrice arrière, ouverture, et lecture des frames

**Compte rendu individuel hebdomadaire (semaine du 26/03/24 au 03/03/24)**

Préparation des questions à poser lors de la réunion du 28/02.

Préparation du partage commun sur GitHub et initiation à son usage auprès des autres membres de l’équipe ( git pull, git push, git commit, git add... )

En termes de code :

* Segmentation de la zone où se trouve la vitesse puis traitement avec Pytesseract ( librairie de reconnaissance sur image )
* Barre de chargement dynamique pour afficher la progression du traitement
* Analyse d’une frame toutes les N ; afin d’accélérer le processus lors des tests

***Éric KHELLA* - Compte rendu individuel hebdomadaire (semaine du 04/03/24 au 10/03/24)**

Semaine calme. Contribution à la rédaction du cahier des charges.

En termes de code :

* Récupération du nombre total de frames
* Lecture de documentation d’OpenCV et recherche de méthodes d’optimisation

***Éric KHELLA* - Compte rendu individuel hebdomadaire (semaine du 11/03/24 au 17/03/24)**

Contribution à la rédaction du cahier des charges.

En termes de code :

* Récupération de l’heure et de la borde kilométrique de chaque frame puis ajout au CSV
* Création d’un fichier de stockages de variables externes ( points clés de la segmentation image, taille de la barre de chargement...)
* Affichage d’un diagramme circulaire de segmentation du temps de traitement
* Tentative de prétraitement vidéo – preprocessing :
  + Reprise de la vidéo passée en gris
  + Application d’un flou gaussien avec OpenCv
  + Application d’un filtre seuil
    - Conclusion : Difficulté de réglage des paramètres, peu de temps gagné, perte en efficience des résultats -> Abandon de la méthode
* Restructuration en fonctions des différentes analyses
* Ajout de la détection de la borne kilométrique au CSV

***Éric KHELLA* - Compte rendu individuel hebdomadaire (semaine du 18/03/24 au 24/03/24)**

Semaine très calme (peu d’avancées sur le projet) car en déplacement depuis jeudi pour se rendre au Challenge Centrale Lyon avec l’IMT Atlantique et partiels d’économie mercredi soir.

Planification d’une réunion le 25/03 pour discuter du cahier des charges et des remarques faites sur celui-ci.

En termes de code :

* Pas d’avancées

Correction du cahier des charges, réécriture des parties techniques peu détaillées, de la conclusion

***Éric KHELLA* - Compte rendu individuel hebdomadaire (semaine du 25/03/24 au 31/03/24)**

Semaine calme dans son ensemble (partiels, championnat de rugby à Vannes) mais week-end productif.

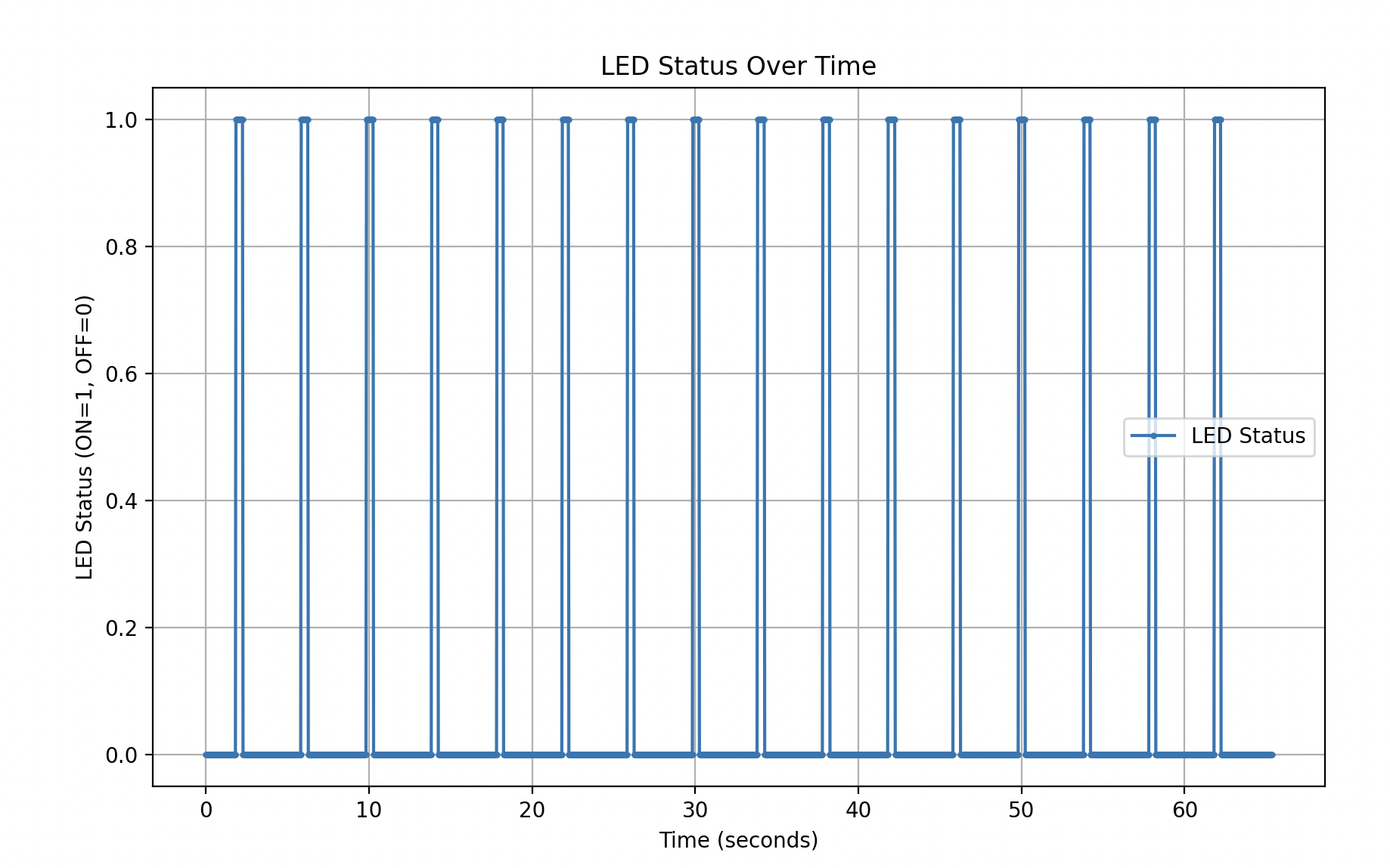
En termes de code :

* Réorganisation du code sur les conseils de l’encadrant ( Nom des variables/ fonctions, Classes, Imports
* Détection de l’état de la LED ( filtre saturation, luminosité...) Dont les paramètres du filtre sont à perfectionner
* Suppression des variables globales
* Mise en place d’un indicateur de temps de calcul restant à côté de la barre de progression

***Éric KHELLA* - Compte rendu individuel hebdomadaire (semaine du 01/04/24 au 7/04/24)**

Semaine calme dû aux partiels d’électronique et du week-end de cohésion Cartel.  
Problème technique suite au commit de Baptiste sur GitHub ; ces changements n’ont pas été effectués et j’ai récupéré une version bien antérieure en voulant récupérer les avancées.

* Adaptation des filtres HSV utilisés pour détecter l’état de la LED
* Mise en place d’un Git revert et réorganisation des branches qui avaient mal fusionné
* Réunion avec Baptiste pour voir ses avancées et lui discuter du fonctionnement du code déjà mis en place concernant la LED qu’il est en train d’améliorer
* Planification de l’extrapolation du temps exact de chaque frame

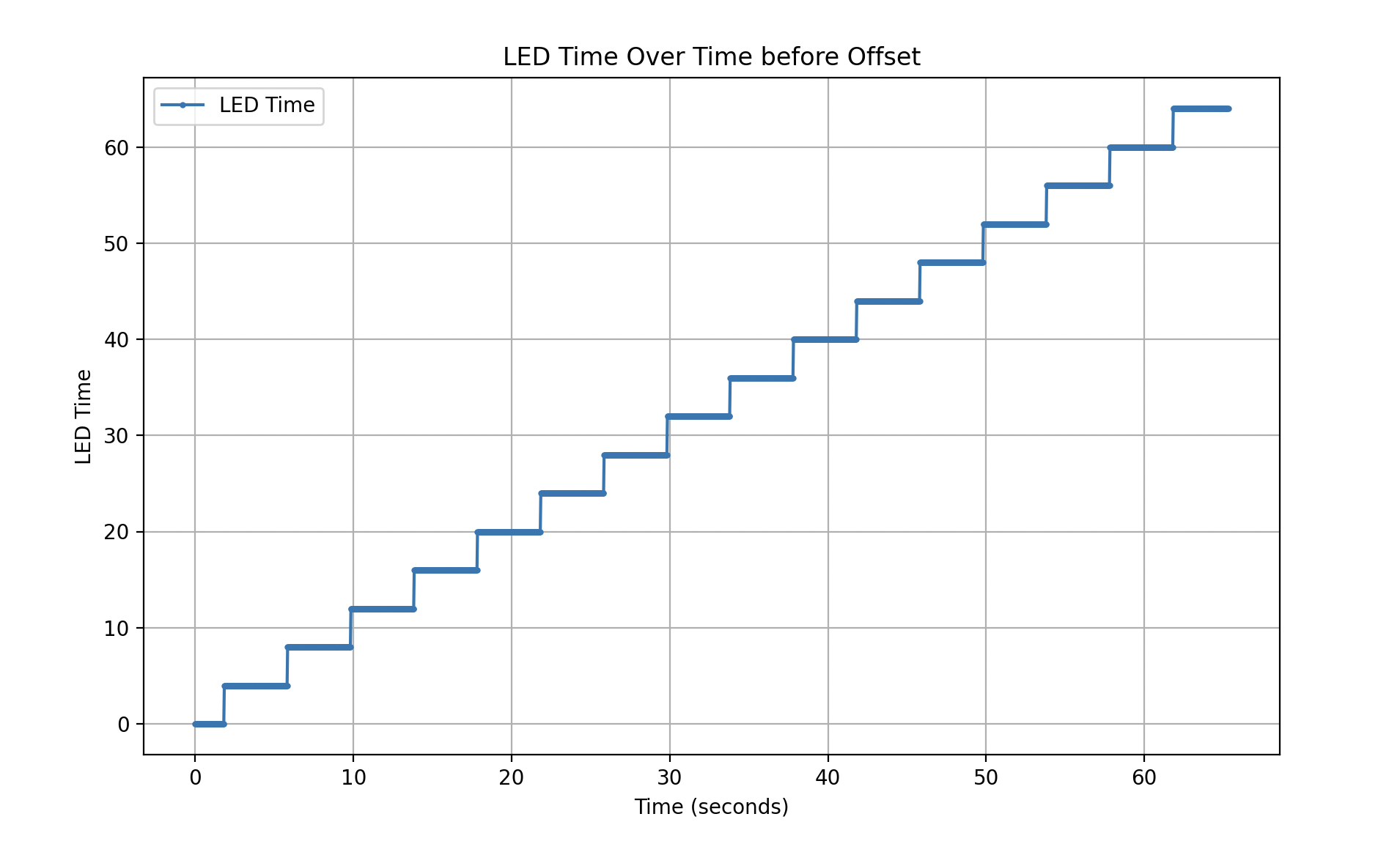


***Éric KHELLA* - Compte rendu individuel hebdomadaire (semaine du 08/04/24 au 14/04/24)**

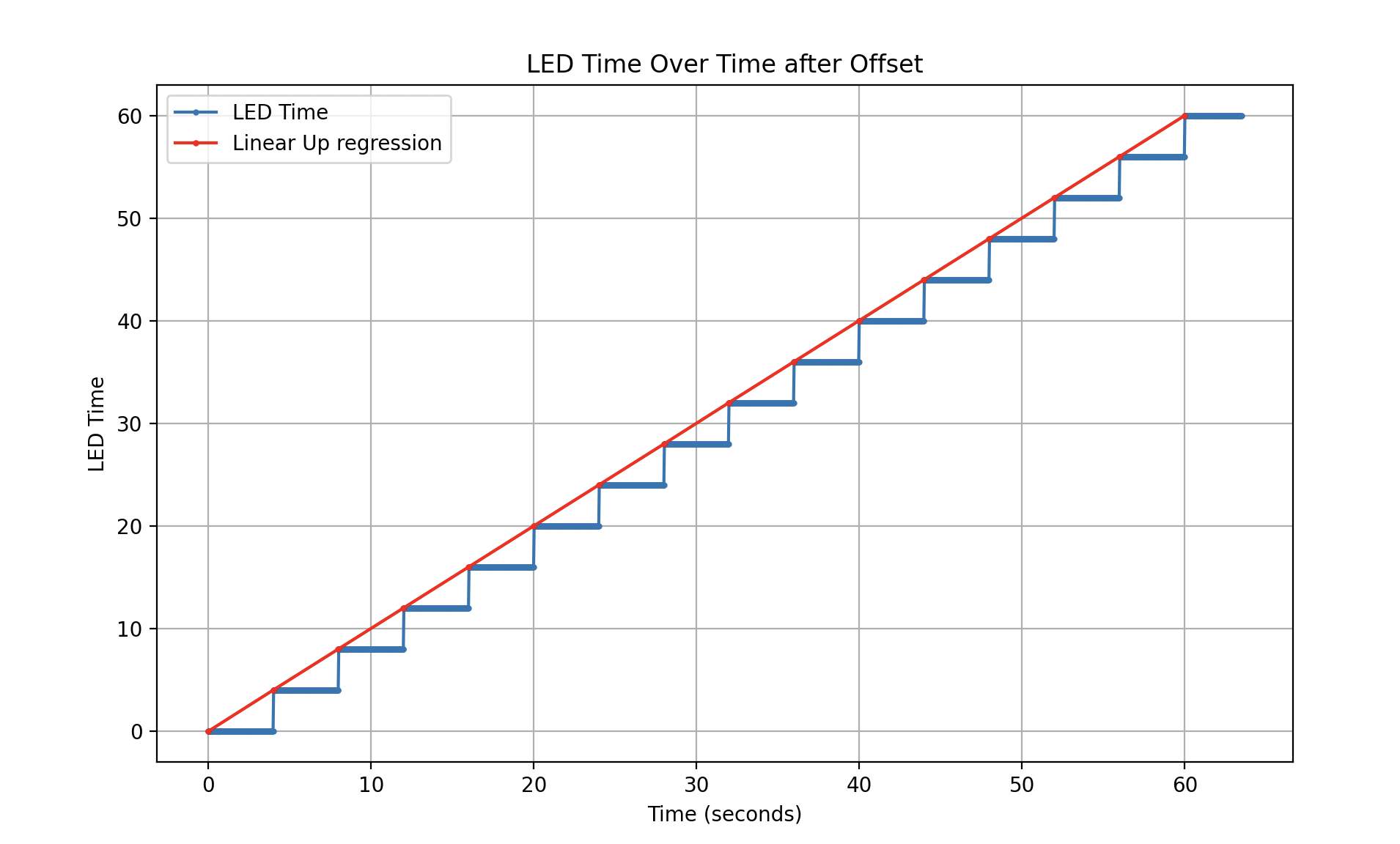
Réunion avec Baptiste et Antoine. Récupération des avancées sur l’ordinateur d’Antoine – Réinstallation des librairies utilisées ( OpenCv, Numpy, Matplotlib... ) sur son ordinateur.  
Récupération des avancées de Baptiste et discussion sur les idées avancées sur le mail de l’encadrant.

En terme de code :

* J’avais précédemment récupéré l’état de la LED dans la semaine du 25 Mars avec des filtres HSV que j’ai adapté ( 0 : éteint, 1 allumé )
* J’ai alors utilisé cette fonctions pour tracer le temps perçu par rapport au temps d’écoulement de la vidéo ( équivalent au numéro de frame étudié )



* On constate que la vidéo commence entre deux états d’allumage de la LED. Ceci est alors dérangeant car il y’a alors un décalage entre les deux échelles temporelles.
* Pour cela, on détecte le premier front montant, et on définit un offset à cet instant là.  
  On définit alors l’origine des temps à partir du moment où la LED s’allume pour la première fois
* Ensuite, on effectue une régression linéaire sur tout les points du front montant.



***Éric KHELLA* - Compte rendu individuel hebdomadaire (semaine du 15/04/24 au 21/04/24)**

Semaine calme -

Réunion avec les membres du groupe et l'encadrant pour faire une mise au point sur les avancées. Discussions sur la clarté du code et les techniques utilisées.

Discussion avec Baptiste quant à la réalisation de la récupération des valeurs RGB.

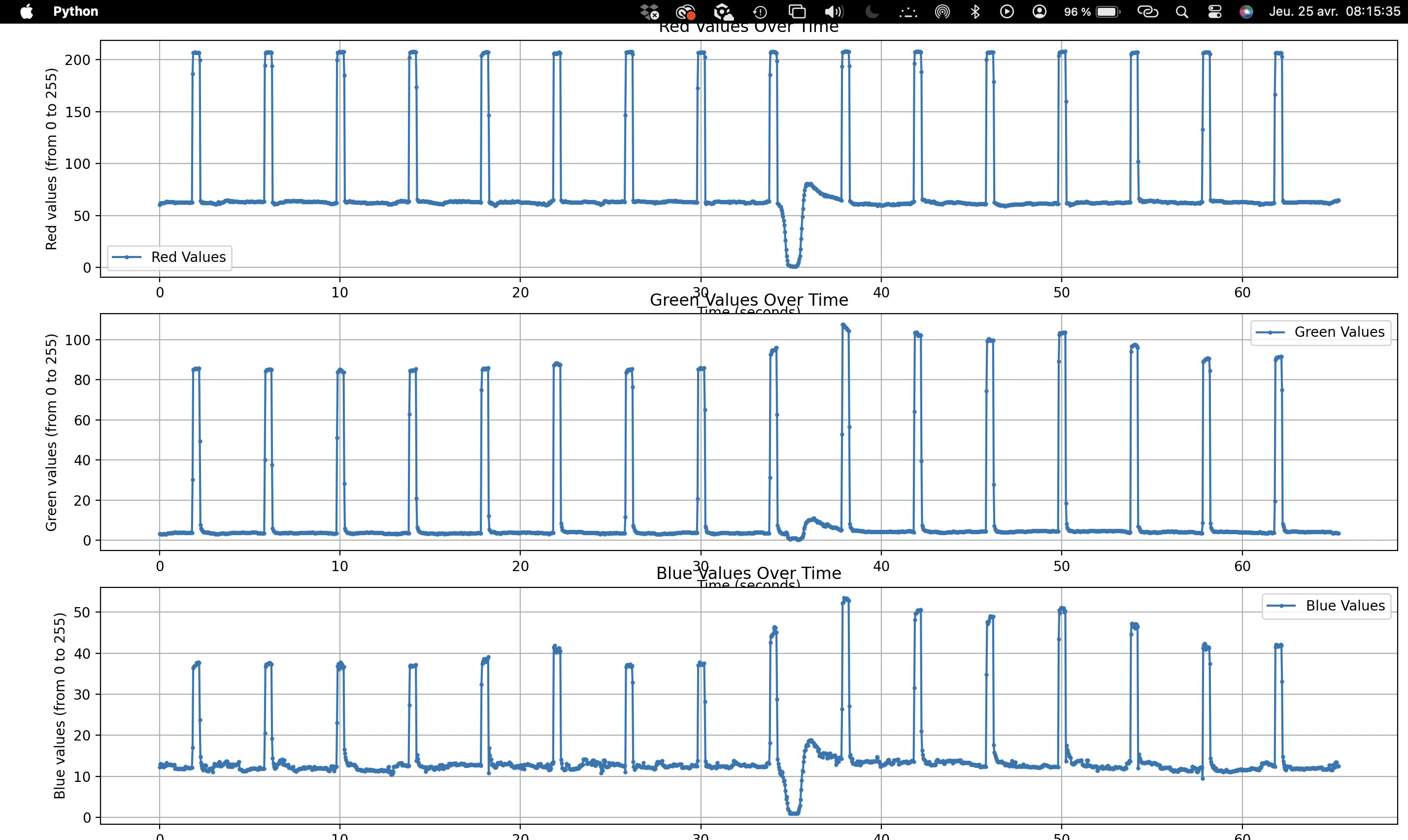
Travail à venir dans le train pour le cartel Jeudi.

***Éric KHELLA* - Compte rendu individuel hebdomadaire (semaine du 21/04/24 au 28/04/24)**

Semaine du Cartel

En terme de code :

* Corrections sur le code de Baptiste ( sur OpenCv, les variables sont stockées en BGR et non pas RGB, donc lorsque l’on affiches les valeurs selon chaque couleur, il semblait incohérent d’avoir + de bleu que de rouge)  
  Problème de virgule manquante lors de l’extraction avec slicing, ce qui expliquait pourquoi certaines valeurs étaient très faible ( il y’avait trop peu de valeurs extraites)



* Début du travail sur la mémoïsation pour l’analyse des valeurs contenues sur les caméras des motrices :
  + L’idée était la suivante :  
    Une fois avoir passé la zone d’intérêt en niveaux de gris, on soustrait terme à terme les termes de la matrice correspondante à la précédente. Ensuite, on somme l’ensemble des termes, et on trace en fonction du numéro de frame cette somme.  
    En effet, si ceux images sont « identiques », la vitesse( ou la borne km/l’heure...) n’a pas changé. Or l’égalité des matrices est difficile à vérifier car une valeur de pixel peut très sensiblement changer ; c’est pourquoi on étudie la soustraction terme à terme
  + Néanmoins, lors de la réalisation, il est très difficile d’analyser le résultat ; il serait alors intéressant d’effectuer cette opération sur une image binarisée ( un seuil pertinent est à déterminer pour faire le NB )



***Éric KHELLA* - Compte rendu individuel hebdomadaire (semaine du 29/04/24 au 05/05/24)**

En termes de code :

* Poursuite du travail sur la mémoïsation ; l’image a été binarisée mais après le traitement précédent, j’aimerais savoir si l’analyse est correcte, pour cela voici mes idées :
  + Je raccourcis la vidéo, pour pouvoir faire un traitement frame par frame beaucoup plus rapide
  + Je trace, en fonction des valeurs contenues sur le csv la fonction indicatrice de changement ( 1 si changement, 0 sinon )
  + Je trace, en fonction du processus précédent la fonction indicatrice de changement
  + Je fais la comparaison des deux ( soustraction sûrement )

***Éric KHELLA* - Compte rendu individuel hebdomadaire (semaine du 06/05/24 au 12/05/24)**

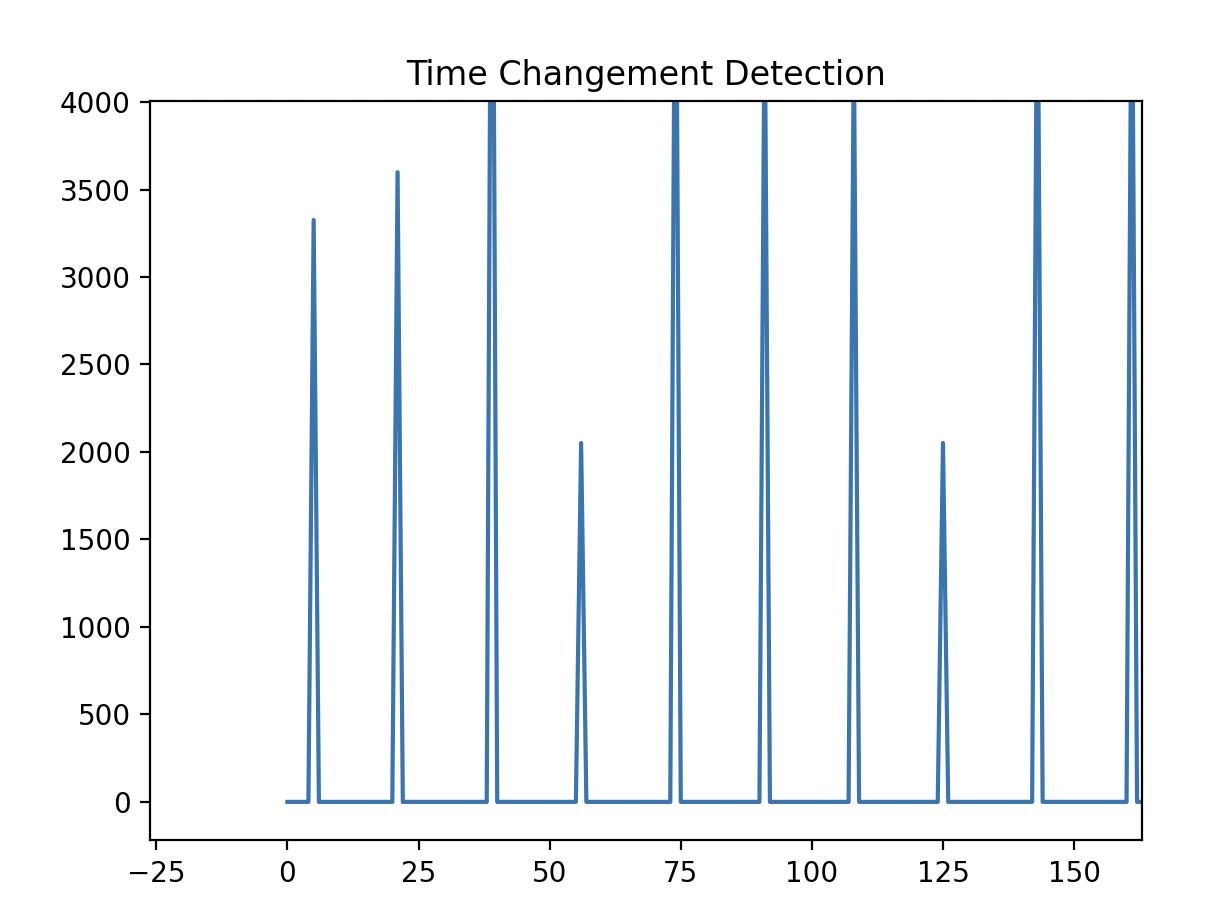
Petit rappel de la définition de mémoïsation : Technique d'optimisation de code consistant à réduire le temps d'exécution d'une fonction en mémorisant ses résultats d'une fois à l'autre

On avait établi précédemment que pour optimiser le temps de calcul et se rapprocher du ratio temps\_réel/temps de calcul, il faut optimiser les détections de chiffres qui prennent beaucoup de temps de calcul

L’idée est la suivante : les opérations d’additions ( soustraction ici ) sont en O(1) et donc si l’on soustrait terme à terme les composantes d’une frame à l’autre et la modification des pixels est minime , alors la zone n’a pas changé et on n’a pas besoin de l’analyser de nouveau ; ainsi on supprime tout les dédoublements de calculs inutiles

Pour se faire, on met d’abord l’image en niveaux de gris puis en noir et blanc (ceci est cohérent car les zones traitées apparaissent noir et blanc ) puis on soustrait terme à terme les arrays d’une frame à l’autre, et on somme toutes les composantes. Ainsi, lorsque l’on a des pics de modifications, la zone traitée à changé, et ainsi les valeurs étudiées aussi.

Ça se fait assez bien, et lorsque l’on trace les modifications de la variable de temps, on retrouve bien une modification toutes les 17sec env. ( ce qui colle avec le 17.25 d’fps )



En terme de code :

* Le bruit précédent était du aux variations des pixels de gris, dont les valeurs sont élevées et peuvent beaucoup varier ; pour cela il fallait mettre en noir et blanc chaque frame
* Pour cela il faut établir un seuil sur les niveaux de gris de l’image ; OpenCv intègre une méthode de seuil pertinente sur les niveaux de gris : le seuil d’Otsu ( cette méthode repose principalement sur l’algorithme de clustering des k-moyennes )

Voilà ce qu’il reste à faire :

* Implémenter une fonction indicatrice de ce changement pour chacune des valeurs
* Faire reposer le processus de détection de caractères sur le conditionnement précédent
* Comparer la détection de changement sur image et les changements de valeurs issus de la détection de caractères

***Éric KHELLA* - Compte rendu individuel hebdomadaire (semaine du 13/05/24 au 19/05/24)**

Petites corrections sur le code de Baptiste :

* Corrections sur l’usage du slicing de listes
* Correction des écrasements de valeurs lors des itérations
* Correction sur l’usage de l’instance self lors de l’appellation des fonctions
* Application de la stratégie

Je rappelle que la stratégie de Baptiste était la suivante :

On sait que la zone dédiée au temps change toutes les secondes ; or à chaque seconde, on analyse de nouveaux les caractères constituant les heures et les minutes ; ce qui rajoute du temps de calcul inutile.

En terme de code :

* J’ai réécrit la fonction detect\_change pour intégrer la partie indicatrice du changement entre deux frames
* J’ai utilisé les valeurs des points définis par Baptiste pour chacune des zones ( heures, minutes, secondes)
* J’ai modifié l’indexation de process\_video car il y’avait une double indexation inutile, désormais on utilise self.frame\_id
* Désormais je fais la distinction entre la zone actuellement étudiée, et la zone précédente; au lieu d’essayer de jouer sur les indices; le code gagne alors en clarté
* Réécriture du code permettant de déterminer quelle portion du temps de calcul a été utilisé pour chaque aspect du code
* On gagne effectivement plusieurs secondes sur le traitement précedent mais quelque chose de plus préoccupant apparaît : le temps de calcul néecssaire à la détection de la borne kilométrique représente alors 85%. En effet, puisqu’il change très souvent, la mémoisation est beaucoup moins efficace que sur les données comme la vitesse ; qui elle change peu.
* Alors j’applique la stratégie de segmentation entre les valeurs dédiées aux kilomètres et aux mètres ; on gagne alors quelques secondes

On arrive aux conclusions suivantes :

La vidéo traîtée fait 2min28 ; le traitement total se fait en 4min26, le traitement de la borne kilométrique prend 3min47.