Documentation Track Trends

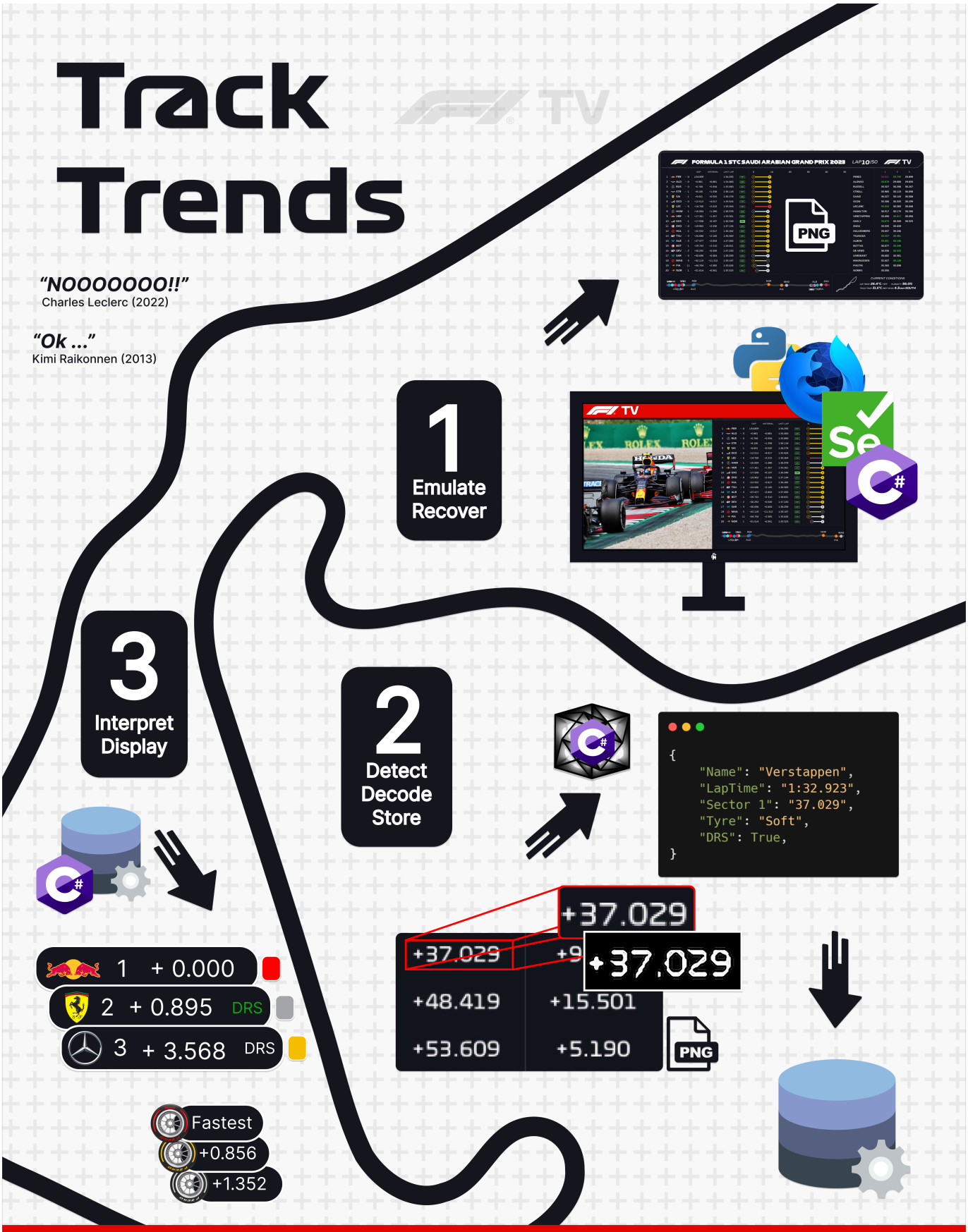
Vroum Vroum

Rohmer Maxime

Table of contents

1. Rapport Track Trends V1.0

Rohmer Maxime Travail de diplôme Technicien ES 2023


Track
Trends
"NOOOOOOO!!"
Charles Leclerc (2022)
"Ok ..."
Kimi Raikonnen (2013)
PNG
1
Emulate
Recover
2
Detect
Decode
Store
3
Interpret
Display
{
"Name": "Verstappen",
"LapTime": "1:32.923",
"Sector 1": "37.029",
"Tyre": "Soft",
"DRS": True,
}
1 + 0.000
2 + 0.895 DRS
3 + 3.568 DRS
+37.029
+37.029
+37.029
+48.419
+15.501
+53.609
+5.190
PNG
Fastest
+0.856
+1.352
Maxime Rohmer
Diploma work Tech II 2023
Tesseract OCR
CSharp
Selenium
Python
Headless Firefox

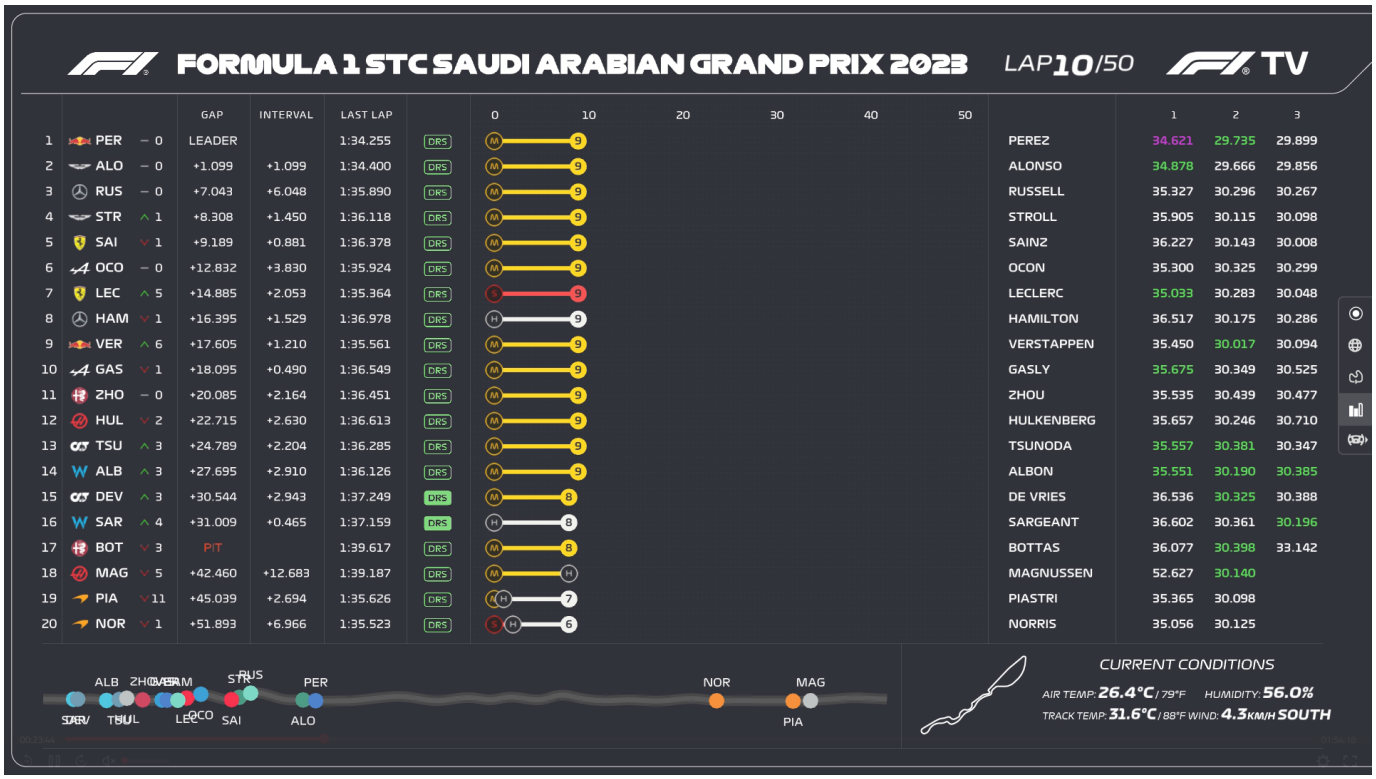
1.1 Introduction

1.1.1 Résumé

Track Trends est un outil de récupération et d'analyse de données de courses de Formule 1.

Pour le contexte, en dehors des cours, j'exerce différentes activités dont celle de Live Ticker F1 pour le 20 minutes. Pour m'aider dans ce travail, j'utilise actuellement la F1TV à laquelle je suis abonné qui me propose non seulement un feed vidéo de meilleure qualité avec des commentaires plus pertinents que ceux de la RTS mais qui aussi me permet d'accéder à un feed vidéo très important : la chaine data.

Ce dernier ressemble à cela :



(Attention ce n'est pas un joli tableau HTML, mais bien une vidéo qui contient un tableau.)

Sauf que toutes les informations sont étalées pêle-mêle sans hiérarchie ce qui fait que cela me prendrait trop de temps de tout déchiffrer à chaque fois, ce qui me fait rater des choses intéressantes.

Le but du projet est donc de fournir un outil qui hiérarchise et affiche différemment les données pour faciliter leur lecture et me permettre de faire de meilleurs commentaires.

1.1.2 Abstract

Track Trends is a Formula 1 data and analysis tool.

To understand everything, a little bit of context. In my free time I have multiple activities and one is to be the Live Ticker F1 for the local journal "20 minutes". to help me in this work I'm currently using the F1TV to which I'm currently subscribed because it provides me with a better video feed with better commentary than the ones from the RTS (in my opinion) but also because it gives me access to a very important video feed : the data channel

See the screenshot above to see what it looks like.

[note: It's a pretty HTML table but a full on video feed that contains a table (probably, so you can't access data directly)]

You can see a lot of data all well and good BUT! All the data is displayed the same in a big table which make it really hard to read totally in a hurry, which means that I miss a lot of useful information.

The point of the project then is to provide with a tool that can display those data by taking into account their relevance. That would help me not miss any and provide a better commentary by never missing out battles, and be able to better write with the time I saved by using it.

1.1.3 Description du besoin

Comme expliqué dans le résumé, je suis Live Ticker F1. Mais pour mieux comprendre le besoin que j'ai, je pense qu'il est pertinent de comprendre comment je travaille.

Pendant un Grand Prix de Formule 1 j'ai plusieurs tâches à effectuer :

- Suivre les différents évènements du Grand Prix
- Changer le titre et la photo de titre du Live
- Chercher des Tweets ou des Images à intégrer
- Ecrire les commentaires en faisant attention à dire ce qu'il se passe mais aussi l’expliquer, ce que cela implique, mais aussi ce que cela veut dire pour le reste de la course.
- Comprendre et expliquer les stratégies

Tout cela toutes les cinq minutes max...

Cela veut dire que je dois être le plus rapide possible quand je cherche des informations. Et comme le tableau en comporte trop et bien, je suis obligé de le lire en diagonale.

Par exemple dans le tableau, les infos que je cherche le plus sont :

- Le nombre de places gagnées (surtout au départ)
- Les écarts entre les pilotes (surtout ceux qui sont en dessous de deux secondes).
- Les pneus de chaque pilote et combien de tours, ils ont fait dessus
- Les temps d'arrêts aux stands
- Les temps au tour (surtout pour la stratégie)

Mais pleins d'autres informations existent si on les recoupe sur plusieurs tours.

Un outil qui permettrait de mettre en évidence les informations importantes serait donc une très grosse plus-value pour mon travail et s'il est facile à installer et à utiliser, il se pourrait qu'il devienne indispensable.

1.2 Cahier des charges

Il s'agit d'une version coupée du cahier des charges qui ne contient pas l'explication du contexte. Mais l'original est disponible sur ce site également. Il est toutefois normal d'y voir des choses répétées ou légèrement différentes, en effet, il n'a pas été écrit en même temps que le reste de ce document.

1.2.1 Projet

Un outil de style compagnon sous forme d'application C# Windows Form qui récupère en temps réel les informations de la course et affiche les informations les plus importantes. Le but est non seulement de faciliter mon job, mais aussi faire en sorte d'améliorer la plus-value de mon travail en me permettant de fournir des commentaires qui ne sont pas disponibles pour le tout venant qui regarde simplement le flux RTS.

Exemples :

- Les pilotes qui sont proches (moins de 1-2 secondes qui sont donc en train de se battre).
- Les pilotes qui améliorent leur temps au tour et ceux qui perdent le plus de temps
- Le classement pondéré tenant compte des futurs arrêts au stand

Maintenant afficher différemment les infos, c'est sympa, mais cela serait encore mieux de traiter ces data et de permettre des petites prédictions.

Exemples :

- Prédire les arrêts aux stands en prenant en compte les baisses de performances des pneus
- Prédire le pneu que le pilote va chausser s'il rentre aux stands dans le prochain tour
- Prédire dans combien de tour tel pilote va rattraper tel autre pilote
- Prédire combien de temps le pilote va perdre dans les stands en fonctions des arrêts précédents

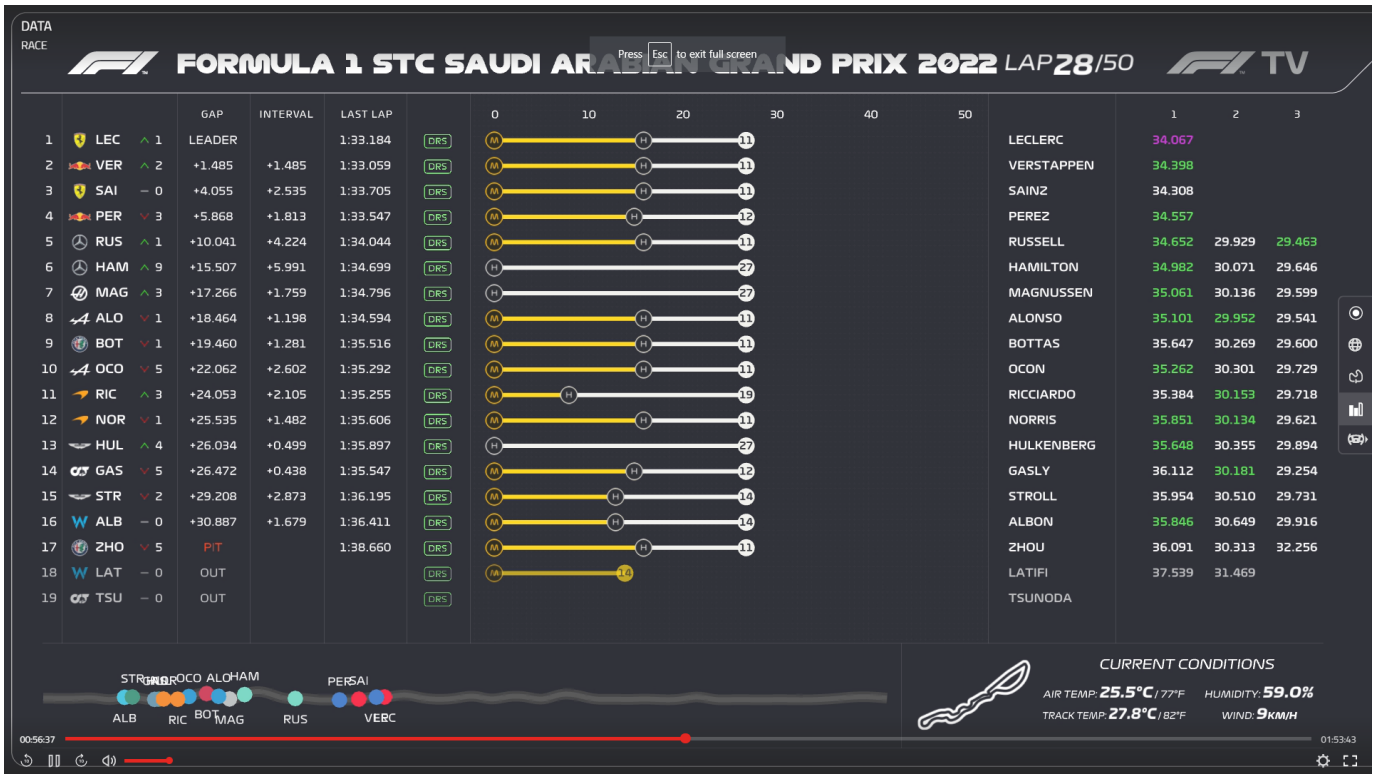
1.2.2 Réalisation

Malheureusement, la Formula 1 Management ne propose aucune API publique qui puisse nous permettre de faire ce projet "simplement". La raison la plus probable étant qu'Amazon avec son service AWS propose exactement ce genre de services pour le flux télévisé et il doit y avoir un contrat d'exclusivité.

Il existe des API "Pirates" faites par la communauté, le problème est qu'elles ne sont pas forcément des plus pratiques à utiliser.

Mais comme je possède un abonnement premium ++ à la F1TV, j'ai accès pour chaque grand prix à un flux vidéo nommé : DATA F1 CHANNEL

Qui ressemble à ça :



Donc la seule façon que je vois de récupérer ces données est de les prendre directement sur ce feed.

Même si le but final de l'application est de faire pleins de choses super avec les datas, le gros du projet va surtout être la récupération des données et leur stockage.

Les données viennent du flux vidéo et ainsi dans un premier temps, il va falloir récupérer d'une manière ou d'une autre des images qui viennent d'un grand prix en direct ou en rediffusion.

Ensuite, dans un second temps, il faut lire les informations directement sur l'image en utilisant une librairie prévue pour (exemple Tesseract) et vérifier l'intégrité de ces dernières pour qu'on puisse ensuite les stocker.

Dans un troisième temps, il faut stocker toutes ces données dans une forme qui permette d'aller facilement faire des requêtes de récupération et déjà préparer des méthodes qui permettent de récupérer des infos importantes (ex : la moyenne des cinq derniers tours, le temps moyen d'arrêt etc.) pour faciliter la dernière étape

Quand tout cela est fait, on peut ensuite s'amuser un peu avec les Data.

La dernière étape est donc l'affichage. L'idée est de créer une Windows Form qui contienne toutes ces informations dans un format beaucoup plus lisible et avec laquelle on pourrait interagir pour permettre de plus facilement commenter les Grands Prix. (exemple plus bas avec un croquis de ce à quoi l'application pourrait ressembler)

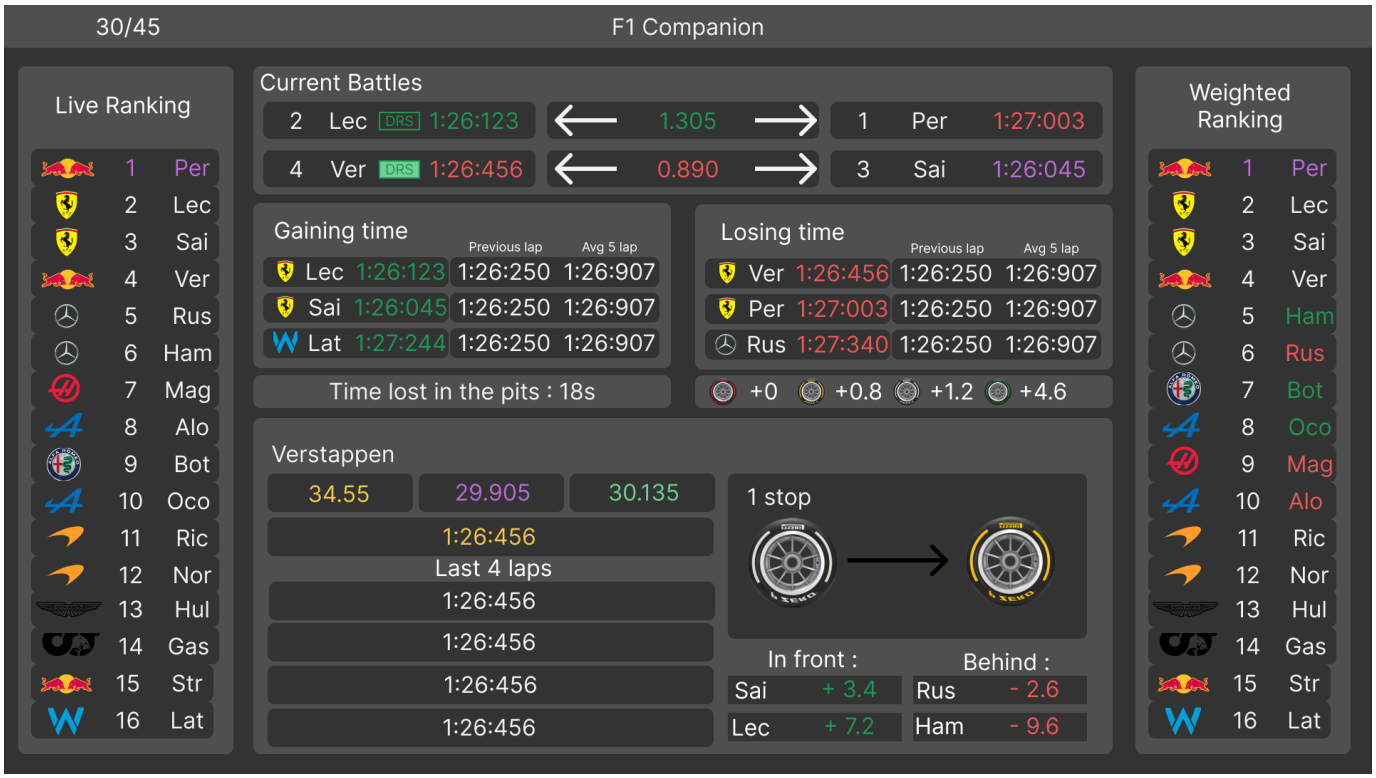
Voici la liste des données qui pourraient être affichées (Non contractuel, simplement des idées).

- Les pilotes qui sont proches (moins de 1-2 secondes qui sont donc en train de se battre).
- Les pilotes qui améliorent leur temps au tour et ceux qui perdent le plus de temps
- Le classement pondéré tenant compte des futurs arrêts au stand
- La moyenne de temps que les pilotes perdent dans les stands
- La performance moyenne des 5 types de pneus
- La moyenne de temps de chaque pilote sur le pneu actuel
- Le nombre de points que chaque pilote gagnerait selon sa position
- Le classement de la course

Voire même si c'est possible :

- Prédire les arrêts aux stands en prenant en compte les baisses de performances des pneus
- Prédire le pneu que le pilote va chausser s'il rentre aux stands dans le prochain tour
- Prédire dans combien de tour tel pilote va rattraper tel autre pilote
- Prédire combien de temps le pilote va perdre dans les stands en fonctions des arrêts précédents
- Prédire les temps au tour de chaque pilote selon l'usure des pneus

Voici un exemple d'interface possible pour une page :



1.2.3 Cas d'utilisation

'*'On va considérer que tous les user ont un abonnement F1 TV PRO

Un user veut récupérer les data :

- Il ouvre son navigateur et lance la page DATA de la F1 TV
- Il calibre la capture des data via le programme (pour la première utilisation).
- Il confirme que les données initiales sont bonnes (pour la première utilisation).
- Il regarde tranquille son Grand Prix

Le programme récupère les data :

- Il récupère des images depuis la F1TV
- Il utilise Tesseract (ou autre) pour en récupérer les infos.
- Il met ces infos dans un Objet Pilote, dans un Objet course avec un attribut tour pour hiérarchiser les data

Pour ce qui est de l'affichage, l'idée est de faire une application C# comme on l'a appris à l'école, mais avec assez de style pour qu'elle puisse être agréable à utiliser.

Quand le programme affiche les data :

- Il prend les données venant directement de la F1TV.
- Il affiche différemment les données pour permettre une meilleure lisibilité
- Il interprète avec des règles plutôt simples certaines data pour faire des miniprédictions ou aider à la lecture
- Il récupère des infos d'autres courses pour les comparer et proposer des prédictions plus intéressantes

1.2.4 Difficultés techniques

- Récupérer un flux vidéo plutôt propre malgré les contres mesures de la F1 TV pour en empêcher la lecture par un logiciel
- Si on doit passer par une capture d'écran, trouver un moyen de stocker les données de manière à prévoir que parfois un tour pourrait avoir plus de données qu'un autre, que le user peut mettre pause, ou même qu'il revienne en arrière.
- Développer des algorithmes pour récupérer les données comme les différents pneus utilisés ou l'activation du DRS ainsi que développer des moyens de nettoyer les résultats de l'OCR (Par exemple utiliser différents champs redondants pour comparer les résultats)
- Stocker les données sur une base pour les traiter plus tard tout en prévoyant un moyen de voir les stats live
- Développer des algorithmes de prédiction qui prennent en compte d'anciennes courses pour tenter de prédire des choses comme les arrêts aux stands par exemple.

1.3 Différences sur le cahier des charges

[À remplir dans les dernières semaines du travail de diplôme]

1.4 Planning prévisionnel

Mes suiveurs m'ont demandé un planning de type GANTT pour ce travail de diplôme

Je n'ai pas utilisé un logiciel particulier pour faire ce dernier, mais je me suis inspiré des principes fondamentaux d'un diagramme de ce type.

Comme l'original a été fait sur Excel, je ne peux pas vraiment l'insérer de manière lisible ici, mais il est disponible dans le dossier Planning.

Mais voici un résumé de son contenu :

1.4.1 Tâches

J'ai décidé de décomposer mon planning en trois grands types de tâches.

1. Programmation
2. Documentation
3. Tests

L'idée est de permettre une meilleure lisibilité et de me permettre à moi de me faire plus facilement à l'idée de ce qu'il m'attend.

Voici la liste des tâches par rubrique :

PT

Cette rubrique contient les tâches qui n'ont pas leur place dans les trois catégories principales.

PT1 / PRÉPARATION AU TRAVAIL DE DIPLÔME (2)

Cette tâche est un peu hors catégorie, mais c'est normal, c'est une supertâche qui regroupe beaucoup de choses. C'est une tâche qui est planifiée pour deux jours et qui normalement devrait être faite les deux premiers jours du travail.

Le but est de préparer tout ce qui peut être préparé en avance niveau documentation et mise en place pour ne pas avoir besoin de s'en soucier ensuite.

DT

Rubrique documentation qui contient toutes les tâches en rapport de près ou de loin avec la documentation du projet.

DT1 CRÉATION DU POSTER (1)

Cette tâche consiste à faire une version numérique du poster qui soit en accord avec les consignes qu'on nous a données. Le but est aussi et surtout de faire poster dont je sois fier et que je sois content de montrer.

Il y a déjà des croquis de poster et j'ai clairement prévu de travailler sur ça pendant les vacances alors, je n'ai mis qu'un jour et je l'ai placé juste avant le rendu de ce dernier.

DT2 DOCUMENTATION ANALYSE DE L'EXISTANT (2)

Cette tâche est dédiée à l'écriture de la documentation et plus précisément de l'analyse de l'existant.

Comme il y a pas mal de technologies utilisées dans mon projet, j'aimerais faire correctement un vrai debrief de pourquoi j'ai utilisé l'une ou l'autre alors, j'ai assigné deux jours dessus.

DT3 DOCUMENTATION ANALYSE ORGANIQUE (5)

Cette tâche est la plus grosse dans la catégorie documentation. Il s'agit de documenter comment l'application fonctionne.

J'y ai mis cinq jours et je pense que c'est un minimum car c'est dans cette tâche que je vais devoir détailler exactement comment fonctionne chaque partie du projet.

Ces cinq jours sont éparpillés sur le projet en général à la fin du développement de chaque grande partie de projet. Le but est de ne rien oublier et de ne pas avoir à tout faire en même temps.

DT4 DOCUMENTATION ANALYSE FONCTIONNELLE (2)

Cette tâche est déjà moins grosse, elle consiste à documenter le fonctionnement de l'application et comment utiliser les composants que j'ai développés.

Je l'ai mis en fin de projet, car comme j'ai l'habitude de faire des analyses fonctionnelles plutôt précises, le moindre changement dans l'UI peut tout rendre obsolète.

J'y ai mis deux jours, car j'aimerais correctement documenter avec de bonnes photos et scénarios pour qu'on puisse voir toutes les possibilités de l'application.

DT5 DOCUMENTATION TESTS (1)

Cette tâche est un peu plus petite qu'elle ne le devrait. Elle concerne la documentation des différents tests. Je n'y ai mis qu'un seul jour, car en réalité les différentes tâches de tests contiennent aussi beaucoup de documentation,

DT6 DOCUMENTATION RESTE (2)

Cette tâche est une tâche un peu vague. Elle contient toutes les actions autres que j'aurai besoin de faire (Mise au propre, orthographe, génération de PDF ...). J'y ai mis deux jours pour avoir un peu de marge, car ce sont toujours des tâches qui paraissent faciles, mais qui à la fin prennent beaucoup de temps si on les fait bien.

PT

Rubrique programmation qui contient toutes les tâches qui touchent à la programmation et au développement de l'application.

PT1 PROGRAMMATION RÉCUPÉRATION DES IMAGES (3)

Cette tâche est estimée à seulement trois jours, il ne faut pas s'y méprendre, c'est une des tâches les plus dures et lourdes niveaux documentation en explications. Cependant, un POC (Proof Of Concept) assez avancé a déjà été fait et donc cela permet de n'envisager que trois jours, car il suffit de l'implémenter et de la paufinner.

Cette tâche consiste à prendre en entrée un lien de Grand Prix et de sortir une image tous les x secondes de la page DATA. Cela peut sembler simple, mais pour le faire sans prendre d'espace d'écran et ne demandant pas à l'utilisateur de copier-coller quoi que ce soit où de donner ses identifiants F1TV c'est un challenge.

Cela peut paraitre curieux alors de mettre cette tâche loin dans le planning même si c'est la première étape du projet. Encore une fois cela s'explique avec le fait qu'il y a déjà un POC qui fonctionne à peu près et que donc préfère commencer avec des tâches plus incertaines dans le cas où elles prendraient plus de temps que prévu.

PT2 PROGRAMMATION OCR (5)

Cette tâche consiste à développer la partie qui reconnait le texte sur les images. C'est très certainement la tâche qui risque le plus de déborder car c'est celle qui est la plus complexe techniquement puisqu'elle demande non seulement la lecture sur image, mais aussi le développement d'algorithmes de traitement de cette donnée pour être sûr qu'elle a bien été lue.

J'y ai ainsi alloué cinq jours, mais j'espère que j'arriverai à gagner du temps sur les autres pour y allouer plus dans le planning effectif, car je suis convaincu que plus, on y passe du temps, meilleur sera le résultat.

PT3 PROGRAMMATION, STOCKAGE ET MODÈLE (5)

Cette partie est moins technique, mais concerne le stockage des données que nous retourne la lecture des images. Mais elle va demander de la réflexion et de l'intelligence de programmation, car il faut que cette partie anticipe les besoins de la vue et prépare un terrain fertile qui ne demande pas un refactor quand on passera au développement de la vue.

C'est pour cela que je lui ai aussi assigné cinq jours de travail et elle doit absolument être commencée après la lecture.

PT4 PROGRAMMATION VUE DE L'APP (5)

Cette partie peut être horrible comme très facile, cela dépend complètement de la qualité du travail avant. Si le modèle est parfait et que les données sont intègres, cela devrait être plutôt simple de les afficher de manière intéressante. Cependant, cette partie débordera sûrement un peu, car tout le temps gagné avec de bonnes données sera utilisé pour tenter de faire de la prédiction.

Pour ces raisons, je lui ai assigné également cinq jours de travail et elle doit absolument être faite après le modèle.

PT5 PROGRAMMATION MISE EN COMMUN (3)

Cette tâche est aussi un petit peu spéciale, car elle regroupe plusieurs choses. En gros, chaque partie de programmation sera sûrement assez indépendante et il faudra à un moment faire un seul projet C# qui contient tout.

Il est difficile d'estimer à quel point cela va être compliqué alors, j'ai été conservateur et j'ai mis trois jours.

TT

Cette rubrique contient les tâches qui sont uniquement des tests. La plupart des tâches de programmations contiennent déjà des tests, mais certaines demandent une attention particulière.

TT1 TESTS OCR (2)

Cette tâche est une des tâches les plus importantes. Son but est de faire un protocole de tests complet qui permette de comparer les différents algorithmes de reconnaissance de texte.

Je l'ai mise après la reconnaissance, mais même maintenant en écrivant ces lignes, je me dis que dans le planning effectif, elle sera faite pendant la tâche de programmation. En effet, comment savoir si mon tout nouvel algorithme est réellement mieux que le précédent.

Je prévois deux jours, car je pense que faire le dataset va prendre beaucoup de temps, il faut prévoir un certain nombre d'images et de texte qui pourront ensuite être données sous forme de tests. C'est certes une tâche de test, mais c'est aussi de la programmation.

TT2 TESTS FINAUX (2)

Cette tâche de tests est un peu vague, elle regroupe les différents tests pour vérifier que les données sont bien affichées correctement. Ce qui serait cool si j'ai du temps en fin de travail de diplôme serait de faire un système de test qui permet d'entrainer le programme à mieux reconnaitre certaines choses comme des arrêts aux stands si on lui donne les trois dernières années de grands Prix.

J'ai mis une durée arbitraire de deux jours, mais je ne sais pas vraiment combien de temps cela va vraiment durer. Elle est évidemment à effectuer une fois que tout est à peu près terminé.

1.5 Planning effectif et différences

[A remplir dans les dernières semaines du travail de diplôme]

1.6 Analyse fonctionnelle

[A remplir au fur et à mesure dans la seconde moitié du travail de diplôme]

1.7 Analyse Organique

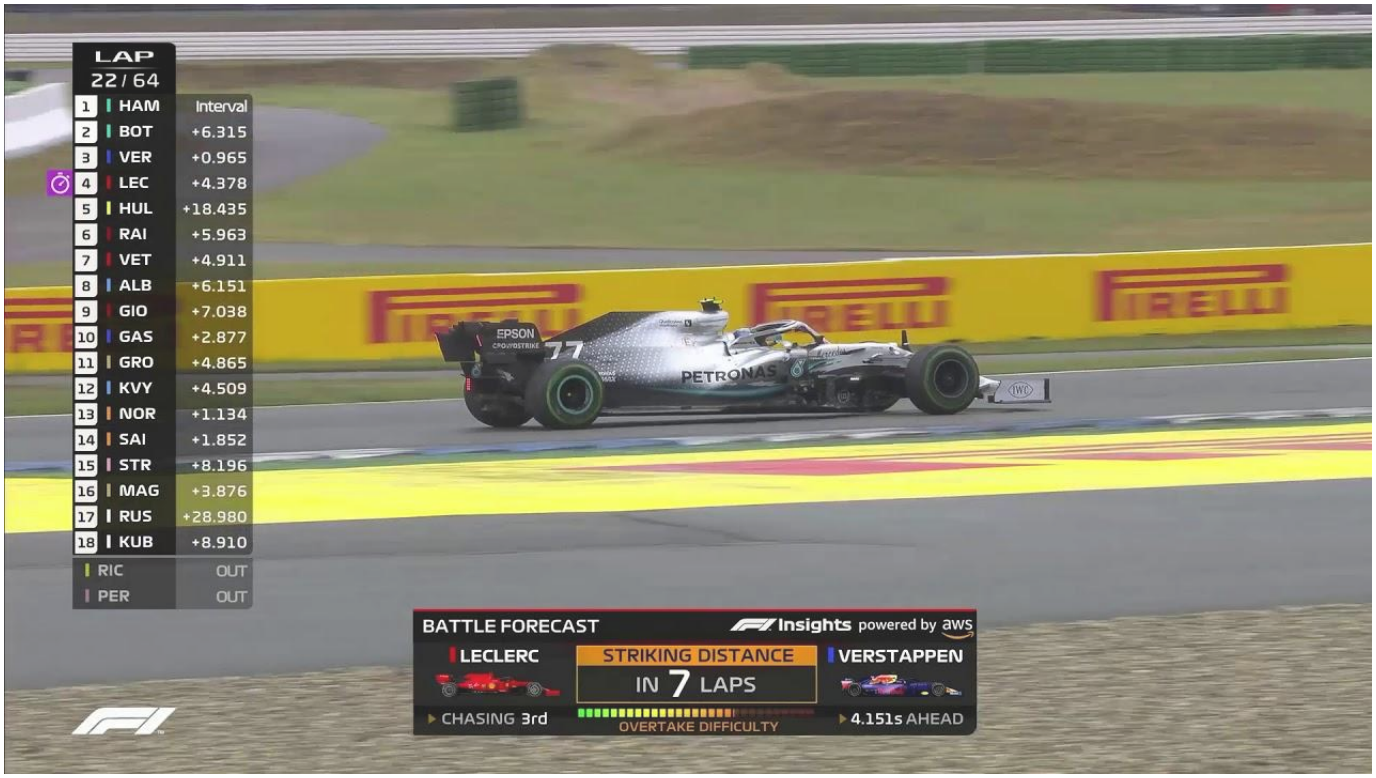
[A remplir au fur et à mesure que les features sont développées]

1.7.1 Récupération des images

Voici la première grande étape du projet.

Pour rappel, Amazon héberge directement le site de la F1TV et possède les droits sur les données de la F1. C'est sous le nom de AWS (le service d'hébergement d'Amazon) que la firme apparait en tant que sponsor.

On peut voir ce nom apparaître assez souvent quand on regarde un Grand Prix car comme ils ont la main-mise sur les données ils peuvent insèrer des bandeaux d'informations sur le flux public sur ce qu'il se passe voir même faire des prédictions (Bien qu'un peu bancales)



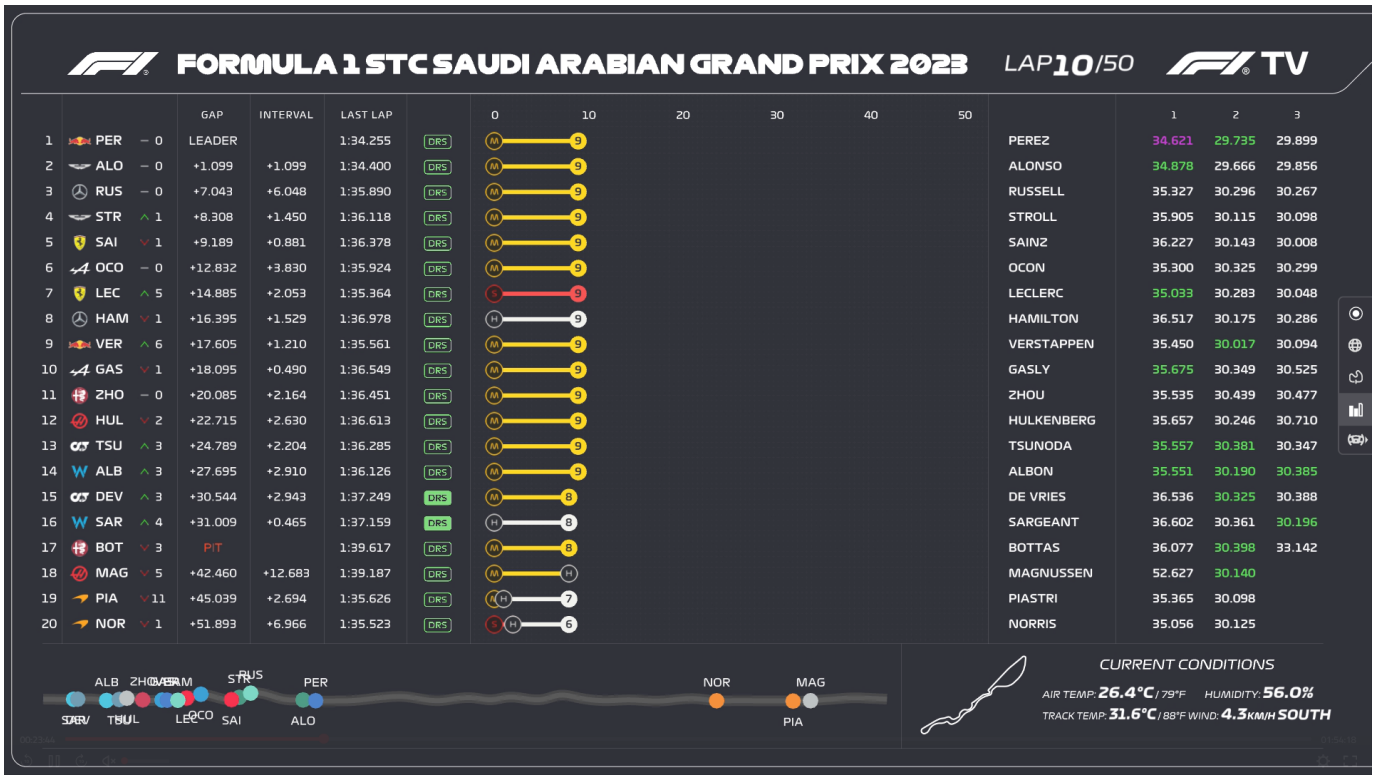
Ce service s'appelle F1 Insights (Oui c'est un meilleur nom de projet que F1 Companion mais bon) et c'est, je pense, la raison pour laquelle on ne voit aucune API publique qui permette de correctement se renseigner en donées en direct pendant un Grand Prix. Ils ont du dégotter un juteux contrat pour s'occuper de toute l'infrastructure digitale de la F1 (du moins publique) en échange d'une exclusivité totale sur certaines choses comme les Data

Evidemment je ne fais que conjecturer et ce que j'ai dit n'est pas à prendre au pied de la lettre mais c'est une explication possible je pense de pourquoi il est si difficile de trouver des données sur la F1 facilement en temps réel.

Il existe bien quelques API un peu bancales publiques, mais le problème c'est qu'elles ne sont vraiment pas suffisante et je ne peux pas leur faire confiance quand je commente. Ce qu'il m'aurait fallut c'est une API publique et officielle qui me permette d'être sur que les données sont les bonnes et qu'elles arrivent le plus vite possible.

On pourrait croire que c'est impossible car cela n'existe pas comme je l'ai dit MAIS ! Ce n'est pas complêtement vrai. En effet depuis que je possède un abonnement à la F1TV, il existe une source d'informations très précieuse qui m'aide énormément dans mon quotidien de commentateur de Formule 1. La "DATA CHANNEL".

La Data Channel est une page de la F1TV qui permet, pour chaque Grand Prix, de visualiser, sous la forme d'un flux vidéo, différentes informations capitales sur la course.



Le problème, c'est que comme je viens de le dire, ces données ne sont pas accessibles comme un tableau HTML ou un flux RSS ou un tableau JSON. C'est un flux vidéo. Il faut savoir qu'entretenir une diffusion de flux vidéo en 1080P pendant deux heures accessible par des milliers d'abonnés est EXTRÊMENT cher surtout quand on le compare à simplement afficher les données dans un tableau. Ce qui veut dire que ce choix est délibéré et a un sens au niveau économique. Je pense donc que c'est justement pour éviter que des petits malins puissent simplement venir scraper l'intégralité des données qu'ils proposent et fasse sa propre API. (C'est d'ailleurs un des sites avec la meilleure protection anti bot du monde)

MAIS ce n'est pas par ce que les données ne sont pas facile à avoir qu'elles sont IMPOSSIBLE à avoir. Et c'est la que ce projet entre en jeu. Mais pour décoder les données d'une image il faut dabord ... (roulement de tambours) ... Avoir des images !

Et c'est la que vient se glisser cette partie du projet.

Comment faire ?

Le but de ce segment est de se concentrer sur la récupèration et la mise à disposition pour le reste du programme, des images en direct de la F1TV dans la meilleure qualité possible et dans les meilleurs délais.

Pour ce faire il y a plusieurs solutions :

- Reverse engeneer la F1TV pour accèder directement au flux sans passer par la plateforme internet et pouvoir prendres images à volonté.
- Avoir tout simplement une page de la F1TV ouverte sur un écran et prendres des screenshots à intervals réguliers.
- Simuler un navigateur internet sans qu'il soit affiché et le contrôler automatiquement pour qu'il prenne des captures.

La première option aurait été la plus élégante mais lors d'un POC que je tentais de réaliser je me suis rendu compte que cela serait un peu trop compliqué et long à faire. Sans compter le fait que les rediffusions de Grand Prix ne sont pas gèrées de la même manière que les diffusions en live. Et que pour faire des Tests en live il faudrait attendre à chaque fois un weekend de Grand Prix et le faire en plus du commentaire que je dois produire.

Pour toutes ces raisons et bien d'autres je l'ai rangée dans la case "Trop dur, Trop chiant, Sûrement illégal" (Oui je sais c'est une catégorie bien spécifique mais c'est ma documentation je fais ce que je veux)

La troisième option aurait été la plus simple (et moins drôle) et je suis presque sûr que je peux implémenter cette dernière en moins d'une après-midi. Sauf qu'elle apporte de gros soucis.

1. On ne peux pas garantir l'intégrité et la continuité des données si l'utilisateur avance ou fait pause même par simple inadvertance.
2. La moindre fenêtre qui s'afficherait devant ruinerait toute la reconnaissance de caractères.
3. On ne peut pas contrôler la qualité du flux et on est obligé de faire confiance en l'utilisateur
4. On ne peut pas vraiment automatiser quoi que ce soit niveau tests ou même pour faire du scrapping auto pour remplir une base de donnée.
5. Et finalement le pire inconvénient : C'EST NUL ! Je ne pourrais jamais utiliser un projet qui fonctionne de cette facon, je ne peux pas me permettre d'avoir un écran inutilisable quand je commente et auquel je dois constamment faire attention pour ne pas perturber la reconnaissance. Pour moi cette option aurait été celle à choisir en cas d'extrême urgence et en dernier recours car le projet deviendrait inutile.

J'ai donc décidé de m'occuper de la seconde option : Simuler un navigateur.

Cette option bien que complexe et difficile à implémenter propose une solution à tous les problême et permet une récupèration quasi sans compromis.

Simuler un navigateur ?

Simuler un navigateur internet n'est pas forcément très difficile. Chromium par exemple offre une panoplie d'outils natifs et énormément de librairies existent permettant de facilement et en quelques lignes simuler un Google Chrome et le contrôler sans afficher son UI.

Cependant. La F1TV n'utilise pas simplement un player HTML5 basique. Elle utilise un service de streaming BitMovin qui permet de fournir un stream de bonne qualité et surtout qui implémente les DRM (Digital Right Management)

Cela veut dire que quand on ouvre un flux de la F1TV sur chrome et que l'on essaie de prendre une capture d'écran, le player se met en noir et ne permet pas de voir quoi que ce soit (Certaines version de Chrome le permettent pendant quelques semaines avant de bloquer à nouveau). Ce qui dans notre cas est un immense problème. Mais Firefox ne nous bloque pas de cette facon et il est donc assez facile de passer outre.

L'explication sans trop rentrer dans les détails est la suivante :

Dans chrome, le player par défaut utilise une technologie appellée "PCP" ou "Protected Content Playback" qui leur permet de bloquer au moins une partie des techniques de récupèration du flux vidéo et audio.

Cependant Firefox de pas sa nature Open Source utilise "OpenH264" pour lire ces mêmes flux soumis à des DRM et OpenH264 n'implémente pas les mêmes restrictions.

Sauf que Firefox n'est pas aussi facilement émulé que chrome et cela réduit notre choix de librairies à ... Une seule... Qui est Selenium. (Il existe aussi Pupetteer C# mais j'ai rencontré énormément de soucis avec cette dernière dès que je voulais lancer une vidéo)

Mais même si la documentation est plutôt maigre parfois, c'est une bonne librairie qui permet de très bien contrôler une instance de chrome ou de Firefox.

Contrôler le navigateur

Maintenant que l'on sait quel navigateur simuler et avec quelle technologie, on peut passer à la réalisation.

Ce qu'il y a de bien avec Selenium, c'est qu'on a un certain nombre de commandes très haut niveau qui nous permettent de contrôler un navigateur de manière plutôt précise.

Je vais décrire ici la procédure habituelle utilisée sous une forme de recette de cuisine pour que l'on puisse facilement comprendre ce qu'il se passe.

Durant cette explication je vais parler à un moment de Cookies, ne vous en faites pas c'est le sous chapitre suivant qui va vous en parler.

Recette de cuisine pour récupèrer des images de la F1TV :

1. Démarrer une instance de navigateur avec les bons arguments
2. Ajouter les bons paramêtres pour ne pas se faire flag comme un bot
3. Naviguer sur la page de la F1TV
4. Ajouter les cookies de connexion pour avoir accès au contenu de la page
5. Naviguer sur la page du Grand Prix demandé
6. Attendre un peu que la page se charge
7. Cliquer sur l'invite de cookies
8. Attendre cinq secondes le temps que la page se reload
9. Cliquer sur le bouton qui permet de passer du feed live à la DATA CHANNEL
10. Appuyer sur Espace pour faire apparaitre le bouton d'accès au paramêtres
11. Cliquer sur le menu déroulant des résolution
12. Trouver l'option 1080P et la selectionner
13. Cliquer sur le bouton qui met la vidéo en plein écran
14. Prendre de screenshots à intervales réguliers

Pour faire toutes ces actions on doit récupèrer les éléments selon leur ID ou leur classe.

Voici un exemple qui récupère le bouton de plein écran et qui clique dessus :

```
IWebElement fullScreenButton = Driver.FindElement(By.ClassName("bmpui-ui-fullscreentogglebutton"));
fullScreenButton.Click();
```

Récupèrer les cookies ?

[FINIR CETTE EXPLICATIOn]

Calibration

[AJOUTER EXPLICATION]

1.7.2 OCR

Ici je vais parler de la seconde partie du projet qui parle du processus de reconnaissance de data sur une image du feed DATA de la F1TV.

C'est je pense la partie qui a demandé le plus tests et de refactor.

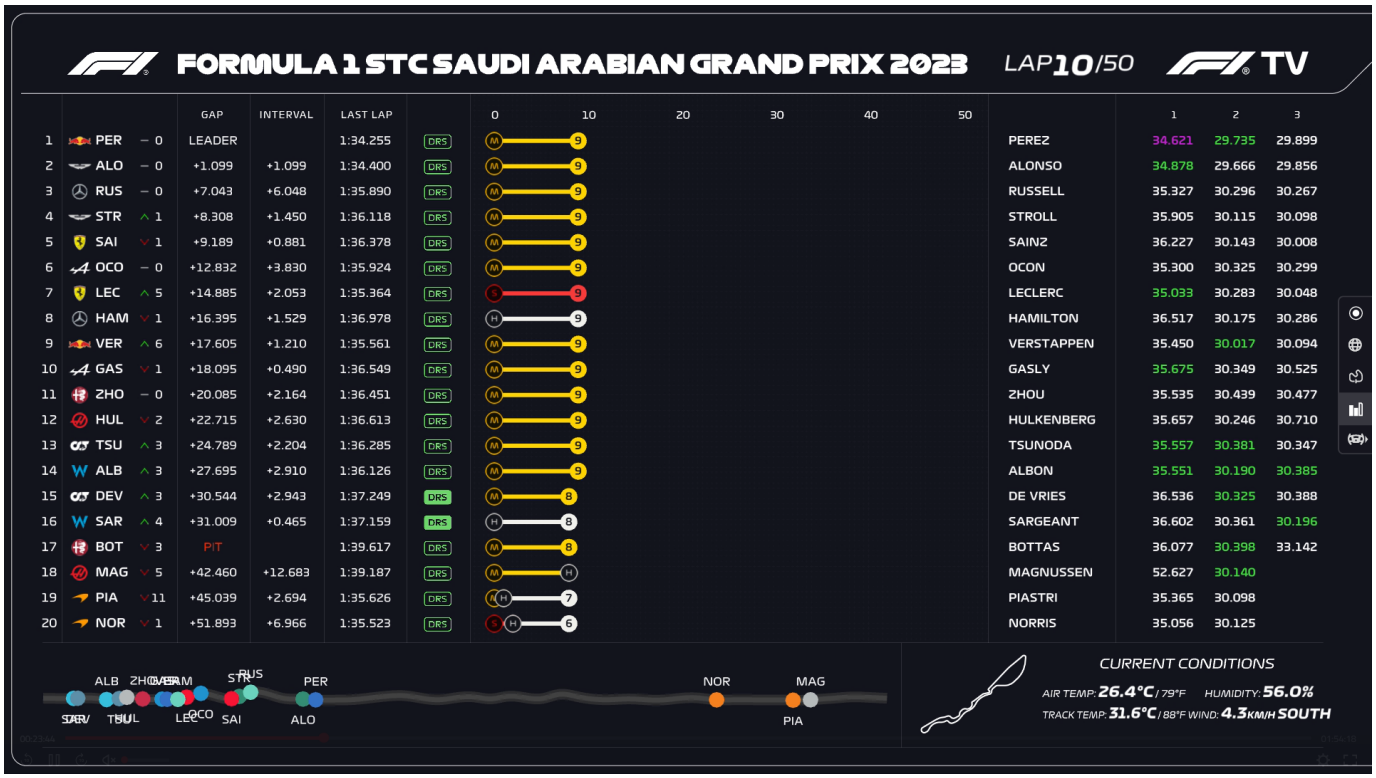
Toute la partie OCR a été développée dans un projet à part avant d'être intégrée dans le projet final.

Il faut savoir que la reconnaissance est différente celon ce que l'on cherche. Je vais donc décomposer cette partie du document en sous rubriques selon les données recherchées.

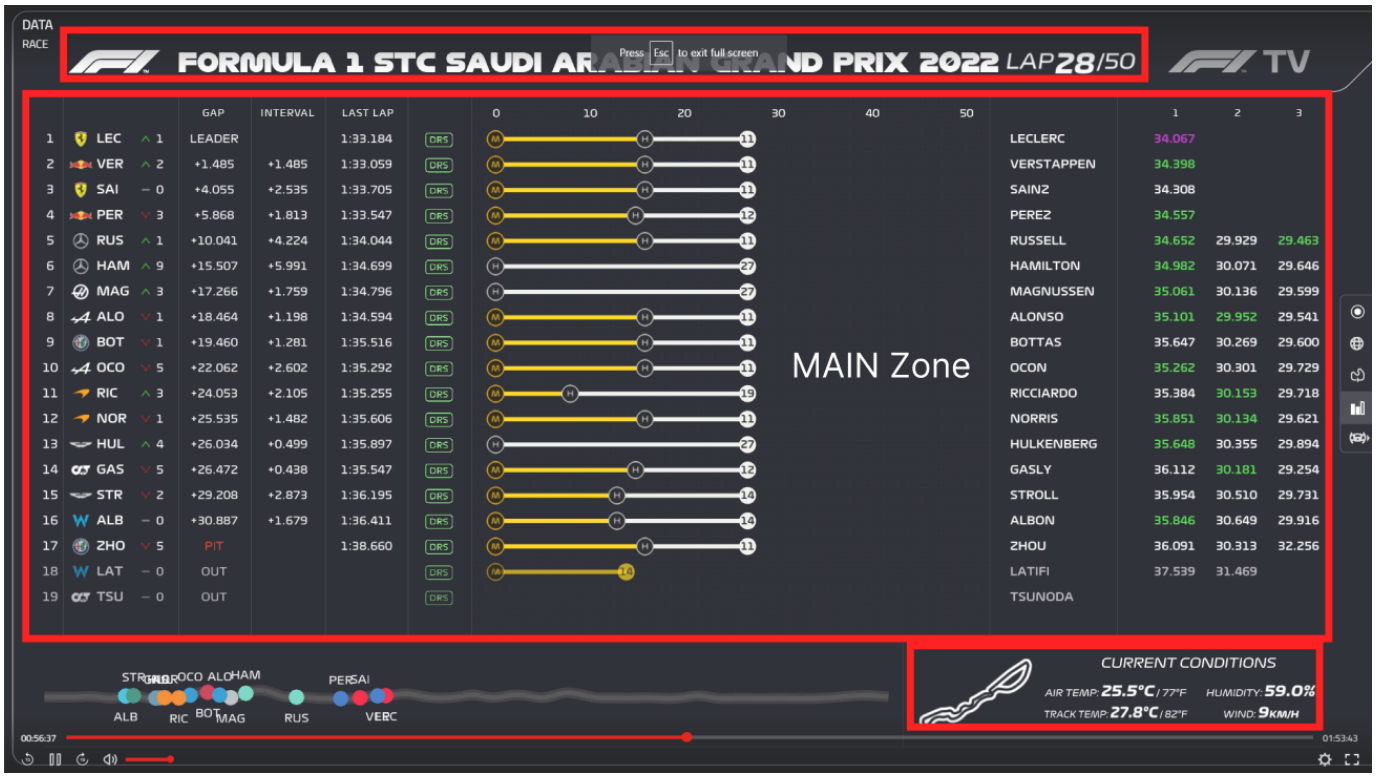
Mais avant ca je dois expliquer certains concepts qui seront importants.

Fonctionnement général

Voici un screenshot de la page DATA de la F1TV que le programme va recevoir :



Si on regarde de loin on peut se dire que la structure est plutôt simple mais c'est loin d'être le cas. On peut y voir au moins 4 zones contenant de l'information dans un format différent.



Dans l'exemple ci dessus on peut voir 3 zones mais on aurait également pu comprendre la zone de position des pilotes autour du circuit pour faire 4.

Ces 4 zones sont très différentes et contiennent d'autres informations. Pour ce travail de diplôme je ne m'occupe que de la zone principale. Mais je pense que le titre et les infos de circuit ne prendrait pas tant de temps que ca à implémenter.

J'ai utilisé le mot "Zone" plus haut et ca n'est pas juste un mot utilisé au hasard. C'est le nom de l'objet que j'utilise pour les représenter dans mon programme. Mais comme c'est important de bien comprendre ce concept avant de continuer je vais vous l'expliquer.

ZONE :

L'objet "Zone" parent est un objet qui est une zone d'image. Je m'explique, le but d'une zone est d'être un morceau d'une image plus grande.

Le but d'une Zone est de contenir une liste de plus petites Zones ou bien une liste de "Window" (j'explique ce que c'est juste après). Elle contient la portion d'image qui la concerne et ses propres dimensions.

Le parent zone ne prévoit que de pouvoir ajouter ou supprimer des éléments des listes de zones ou de windows ainsi qu'une methode qui permet d'aller chercher toutes informations des livres qu'elle contient.

L'intérêt d'une zone est de pouvoir compartimenter une image dans des parties intéressantes au niveau de la reconnaissance mais pas de traiter d'information.

WINDOW :

L'objet "Window" est un objet qui peut ressembler beaucoup à l'objet "Zone". En effet elle aussi est une partie d'une image plus grande et contient ses dimensions, mais elle se distingue en deux points importants.

- Elle ne contient pas d'autres Zones ou Windows
- Elle peut retourner les informations écrites sur son image.

Toutes les Window qui héritent du parent Window peuvent implémenter une methode qui permet de renvoyer ce qui peut être décodé sur son image. Les enfants peuvent aussi aller piocher dans les nombresues methodes de récupèration de données contenues dans le parent Window. Mieux vaut réutiliser le plus possible que de réinventer la roue pour chaque Window.

Une analogie un peu bancale pourrait se présenter comme la suivante :

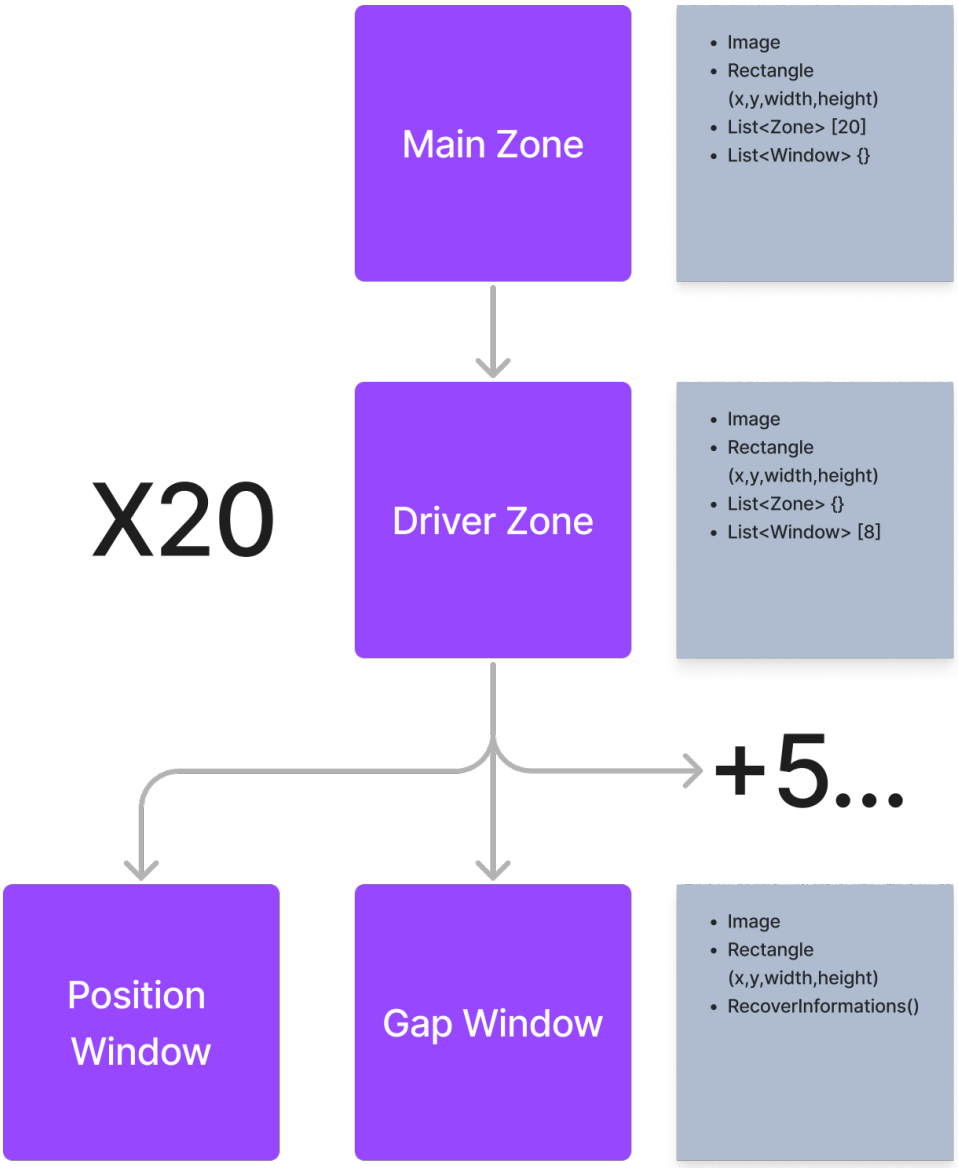
La zone est une armoire ou une bibliotèque. Si c'est une zone qui contient d'autres zones c'est une bibliotèque et chacune de ces sous-zones sont des armoires. Leur unique but est de contenir de manière ordonnée des objets qui eux contiennent de l'information.

Les livres ici sont les Windows. Ils contiennet de l'information et sont stockés dans des armoires et on y accède en allant dans la bonne bibliotèque et en allant dans la bonne armoire.

Dernières choses pour comprendre le diagramme:

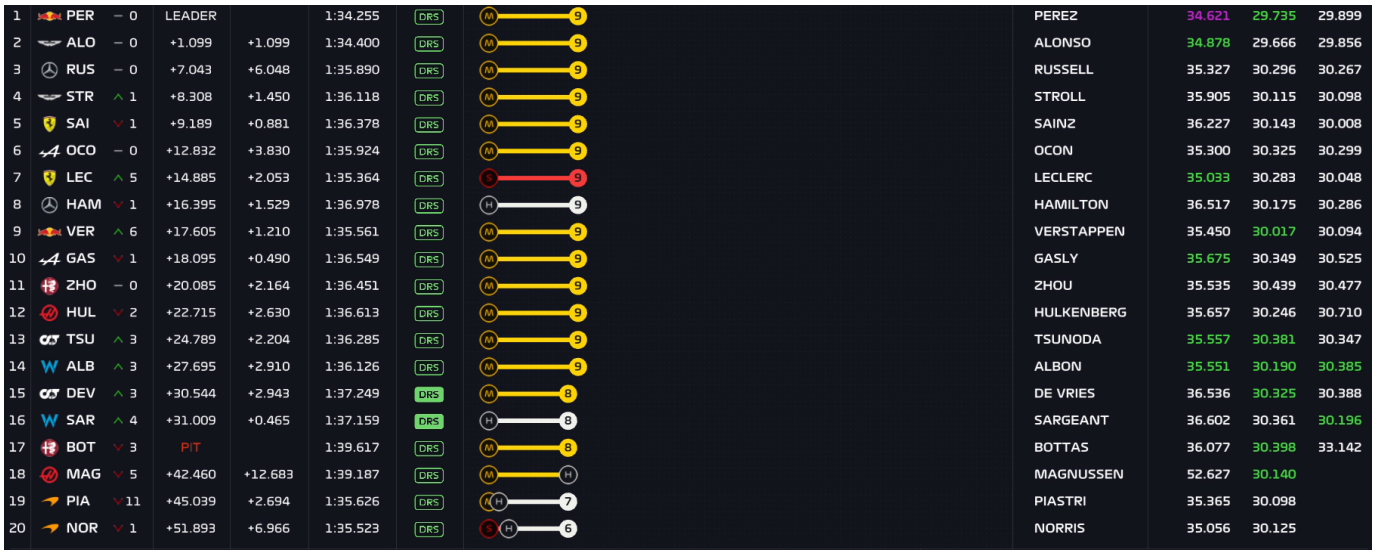
- Il existe une Main Zone qui est une des 4 grandes zones dont je parlais dans la décomposition de l'image.
- Il existe aussi des "Driver Zone" qui sont de plus petites zones contenues dans la Main Zone qui et qui ne contiennent que les informations d'un pilote.
- L'objet Window n'est quasi jamais utilisé, c'est presque tout le temps des enfants de Window plus spécifiques qui sont utilisés, le but est que chaque type d'information sur l'image aie son type de window.

Voila donc un petit diagramme qui montre le découpage du programme :



Pour visualiser encore un peu mieux comment ce découpage prend forme voici ce que chaque zone et Window contient.

Main Zone :



1	PER	– 0	LEADER		1:34.255	DRS	M 9	PEREZ	34.621	29.735	29.899
2	ALO	– 0	+1.099	+1.099	1:34.400	DRS	M 9	ALONSO	34.878	29.666	29.856
3	RUS	– 0	+7.043	+6.048	1:35.890	DRS	M 9	RUSSELL	35.327	30.296	30.267
4	STR	^ 1	+8.308	+1.450	1:36.118	DRS	M 9	STROLL	35.905	30.115	30.098
5	SAI	v 1	+9.189	+0.881	1:36.378	DRS	M 9	SAINZ	36.227	30.143	30.008
6	OCO	– 0	+12.832	+3.830	1:35.924	DRS	M 9	OCON	35.300	30.325	30.299
7	LEC	^ 5	+14.885	+2.053	1:35.364	DRS	S 9	LECLERC	35.033	30.283	30.048
8	HAM	v 1	+16.395	+1.529	1:36.978	DRS	H 9	HAMILTON	36.517	30.175	30.286
9	VER	^ 6	+17.605	+1.210	1:35.561	DRS	M 9	VERSTAPPEN	35.450	30.017	30.094
10	GAS	v 1	+18.095	+0.490	1:36.549	DRS	M 9	GASLY	35.675	30.349	30.525
11	ZHO	– 0	+20.085	+2.164	1:36.451	DRS	M 9	ZHOU	35.535	30.439	30.477
12	HUL	v 2	+22.715	+2.630	1:36.613	DRS	M 9	HULKENBERG	35.657	30.246	30.710
13	TSU	^ 3	+24.789	+2.204	1:36.285	DRS	M 9	TSUNODA	35.557	30.381	30.347
14	ALB	^ 3	+27.695	+2.910	1:36.126	DRS	M 9	ALBON	35.551	30.190	30.385
15	DEV	^ 3	+30.544	+2.943	1:37.249	DRS	M 8	DE VRIES	36.536	30.325	30.388
16	SAR	^ 4	+31.009	+0.465	1:37.159	DRS	H 8	SARGEANT	36.602	30.361	30.196
17	BOT	v 3	PIT		1:39.617	DRS	M 8	BOTTAS	36.077	30.398	33.142
18	MAG	v 5	+42.460	+12.683	1:39.187	DRS	M H	MAGNUSSEN	52.627	30.140	
19	PIA	v 11	+45.039	+2.694	1:35.626	DRS	M 7	PIASTRI	35.365	30.098	
20	NOR	v 1	+51.893	+6.966	1:35.523	DRS	S H 6	NORRIS	35.056	30.125	

Driver Zone :

2	ALO	– 0	+1.099	+1.099	1:34.400	DRS	M 9	ALONSO	34.878	29.666	29.856

Driver Position Window :

Driver name Window :

Driver LapTime Window :

Driver Tyre Window :

Il existe d'autres types de Window mais ce sont les principaux.

On se rend assez facilement compte que chacunes de ces windows va avoir besoin d'un traitement spécifique car la manière de reconnaitre le pneu utilisé et le temps au tour ne peut pas être la même.

Pour résumer, on a un programme qui prend en entrée un fichier de configuration, qui prend des images de la F1TV et les découpe dans des ZONES qui elles même sont découpées en WINDOWS pour qu'on puisse plus facilement les décoder.

Maintenant qu'on a une liste de différent types de zones on peut commencer à chercher ce qu'il y a marqué dessus.

Pour cela il faut dabord comprendre un petit peu comment l'OCR fonctionne et comment des libraries comme Tesseract fonctionnent pour donner du texte en partant d'une image.

Pour faire très simple, nous avons un modèle qui est entrainé. C'est à dire que on donne à un programme un très grand nombre de mots ou de lettres en lui disant ce que contiennent chaques images. Ensuite le programme va créer des matrices de convolutions pour chaque lettre avec comme objectif de detecter les points communs entre les lettres pour créer un alpphabet.

Par exemple la matric de la lettre 'H' donnerait un poids important à des lignes verticales connectées par une ligne centrale. Et si on fournis assez de données de bonne qualité au modèle, les matrices peuvent être très efficace à detecter si une lettre est un H ou un M.

Il y a pleins d'autres methodes comme l'utilisation d'un dictionnaire de mots de la langue pour permettre la reconnaissance de mots même si une lettre au milieu n'est pas comprise ou en ajoutant d'autres informations sur le contexte mais ca ne nous intéresse pas ici.

C'est important de comprendre comment cette reconnaissance de caractères avec des matrices fonctionne car cela va nous aider à préparer nos données pour lui rendre la vie facile et augmenter la précision de nos résultats.

FILTRES ET TRAITEMENT

On peut essayer de donner toutes nos images directement à Tesseract pour qu'il reconnaisse tout le texte qu'il y voit mais on risque de se retrouver avec des résultats au mieux inconsistents.

Dans notre cas, le soucis est que les chiffres et lettres sont beaucoup trop petits. Ils ne font parfoisd que 10 pixels de haut et cela fait que il n'est pas forcément facile de toujours les différencier. De plus, comme ils sont petits, les artéfacts d'aliasing sont assez violents et peuvent grandement déformer une lettre ou un chiffre.

Exemple :

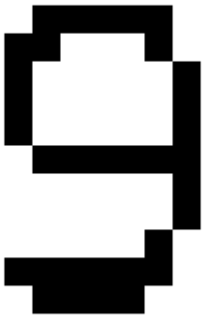
Prenons le chiffre 9. Dans l'image il peut être représenté de cette manière :

On peut voir qu'il est flou, pour nous cela ne pose pas de problème et je pense que à peu près nimporte qui peut dire que c'est un 9.

Cependant comme les contours sont flous et même si on essaie de retirer le background :

On voit que le 9 n'est pas clairement définit. En effet on pourrait le comprendre comme :

Ou comme :



Voire même simplement comme :

Et on se rend bien compte que les performances de detection ne sont pas les mêmes dans ces trois cas.

Il faut donc faire un certain post traitement des images pour supprimer les éléments parasites, les couleurs, et augmenter la visibilité des contours importants.

Mais chaque type de donnée va avoir des methodes de post traitement différents.

Donc voici les différents types de reconnaissance et leur post traitements :

Texte

Alors ce type de reconnaissance est utilisé par la WINDOW du nom de pilote et de la position du pilote.

C'est je pense la plus simple de toutes car Tesseract est particulièrement bien entrainé pour.

Cette reconnaissance concerne donc des lettres qui font des mots ou des noms.

Voici un exemple de la WINDOW nom de pilote en entrée :

Ce texte peut paraitre bon, cependant quand on le lance dans Tesseract, il ne va pas toujours donner un résultat parfait. Il faut aussi savoir qu'il y a des noms pas mal plus pénibles que Tesseract a plus de mal à reconnaitres, soit à cause des lettres utilisées, soit car le nom est un nom d'une autre région et qui ne veut rien dire en anglais ce qui empêche l'utilisation de dictionnaire (Ex : Tsunoda est un nom japonais et parfois il est difficile pour Tesseract de le reconnaitre car si une lettre pose problême il ne peut pas trouver de contexte qui puisse l'aider).

Donc pour le rendre plus facilement lisible et augmenter les chances que toutes les lettres soient découvertes, voici les étapes que j'ai mis en place.

1 : J'inverse les couleurs. Je me suis rendu compte que il était souvent plus facile de trouver un noir sur blanc que blanc sur noir. Je ne suis pas sur que cette étape soit capitale cependant.

2 : Je fais un Treshhold de 165 car avec moins le texte parfois prend trop du background et avec plus les lettres sont trop fines.

3 : Je fais un Resize de l'image pour avoir une meilleure résolution et permettre une meilleure détection. J'augmente la hauteur et la largeur par un facteur 2. J'ai trouvé cette valeur suffisante et aller plus haut consomme beaucoup de ressources.

4: Je fais une très rapide Dilatation du texte pour retirer le flou amené par la methode de Resize. Je n'utilise qu'une valeur de 1 car je ne veux pas trop changer comment le texte est modelé je veux juste retirer le flou.

Explication des methodes précises plus bas

Voila pour ce qui est du post processing. Je ne dis pas que ce sont les meilleurs paramêtres possibles mais dans mes tests ce sont ceux qui ont le mieux marchés.

C'est aussi les premières methodes que j'ai pu développer alors forcément elles n'ont pas le niveau de détails de certaines autres.

Mais comme même avec ce traitement il n'est pas rare que je me retrouve avec une ou deux lettres pas justes, il faut un moyen d'être sûr que c'est le bon nom qui est trouvé. Ce qu'il y a de pratique avec les noms de pilotes c'est que on sait déja comment ils s'appellent avant le Grand Prix.

En effet dans le fichier de configuration de la reconnaissance, il y a une liste de noms de pilotes. Cela veut dire que au lieu de chercher à trouver parfaitement les bonnes lettres, on peut simplement essayer de trouver quel nom de pilote ressemble le plus au nom trouvé sur l'image.

Pour ce faire j'ai utilisé une methode appelée la distance de Levenshtein. Pour faire simple c'est une methode qui va calculer les distances de lettres pour determiner entre des strings laquelle ressemble le plus à une autre.

Pour résumer le fonctionnement dans lordre :

- On prend l'image on la traite
- On envoie l'image traitée à Tesseract
- On trouve quel nom de pilote ressemble le plus à ce résultat
- On renvoie le nom du pilote

Chiffres

Cette methode en réalité utilise simplement la même methode que celle qui va récupèrer le texte sur une image. Cependant, la, on envoie à Tesseract l'information qu'il ne peut trouver que des chiffres sur l'image ce qui lui permet d'être beaucoup plus précis et de ne pas confondre un 9 avec un P ou un 11 avec un H PAR EXEMPLE (non pas que ca me soit arrivé très régulièrement et que ca me soit resté dans la gorge évidemment)

L'avantage c'est que cette methode ne demande même pas de traitement de la donnée en sortie de Tesseract. On éspère simplement que le post traitement aura suffit.

TEMPS :

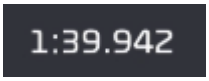
Cette methode regroupe la détection de temps au tour. Il y a trois grands types de WINDOW qui sont concernées :

- La WINDOW du temps au tour
- La WINDOW du retard sur le leader
- La WINDOW des secteurs

La grande différence ce sont les ordres de grandeur. Les temps au tour sont en général entre 50 secondes et 2 minutes. Tandis que les secteurs sont entre 20 et 30 secondes alors que le retard sur le leader peut-être de plusieurs minutes.

Cependant, tous ces temps possèdent le même type de post-traitement avant d'être envoyés à Tesseract.

Voici un exemple de temps au tour avant toute transformation :



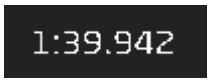
On peut avoir l'impression que ce texte est tout à fait lisible et facile à décoder surtout quand on le voit de loin comme ca. Cependant, il faut imaginer que ces chiffres font 13 pixels de haut en comptant le flou et comme expliqué plus haut ce flou dans ces echelles est terrible.

Si on donne cette image à Tesseract, les '3' deviennent des '9', des '9' deviennent des '8', des '2' deviennent eux aussi des '9', le tout parfois inversement et de manière complètement imprévisible. Ca n'est simplement pas utilisable.

Cette partie est un peu plus complexe car si la detection n'est pas fiable les chiffres sont simplement inutilisables. Si à tout moment un temps au tour de 1:39.106 devient 1:32.108 c'est juste pas possible.

Voici donc les étapes de post-traitement que j'ai mis en place pour leur détection :

1: J'applique un Treshold de 185 pour enlever les ambiguités d'alisaising et avoir une image en noir et blanc claire. La valeur de 185 est assez élevée car le but est de vraiment garder uniquement les contours. Comme les chiffres se ressemlent beaucoup plu que les lettres, il faut tenter le plus possible de conserver leur formes spécifiques. Je me suis rendu compte que cette valeur était une de celles qui marchent le mieux.



2: J'applique un Resize de 2 pour augmenter la résolution des chiffres et permettre une meilleure détection. Le but est d'avoir plus de pixels et donc de permettre à Tesseract de mieux utiliser ses matrices de convolution.

3: Comme le Resize amène du flou, j'utilise une methode de Dilatation qui me permet de retirer ce flou et de remplir un peu plus certaines parties qui ont été un peu laissée par le Resize*;

4: Contrairement aux mots plus haut, la rondeur ajoutée par la dilatation n'est pas vraiment désirée. En effet, elle peut rendre confuse certains chiffres et empêcher Tesseract de bien trouver le chiffre. Alors j'applique une Erosion qui me permet de contrecarrer en partie les rondeurs ajoutées par la dilatation et retrouver des chiffres bien formées. Pour l'Erosion et la Dilatation j'ai utilisé une valeur de 1 car je ne voulais pas trop changer les chiffres.

Explication des methodes précises plus bas

Et avec ce post processing on retrouve de plutôts bon résultats qui demandent peu de traitement.

Le traitement dépend du type de WINDOW cependant.

- Pour les secteurs on indique à Tesseract que les caractères autorisés sont : "0123456789."
- Pour les temps au tour on autorise plutôt "0123456789.:"
- Et pour les écarts on autorise "0123456789.+"

Ensuite on récupère une liste de chiffres qui'il va falloir transformer en milisecondes pour faciliter le stockage et l'envoi.

Le programme nettoie un peu la chaine avant de la convertir. Par exemple parfois le ':' de 1:34.456 est compris comme un '1' ou un '2' et il faut faire attention à detecter quand ca arriver.

Je passe les détails du reste du nettoyage car c'est vraiment du cas par cas mais quand on a finit de nettoyer la chaine on peut transformer les chaines de minutes secondes et milisecondes en un total de milisecondes.

Pour résumer le fonctionnement dans l'ordre :

- On prend l'image et on lui applique une série de filtres
- On envoie l'image filtrée à Tesseract
- On nettoie le résultat Tesseract pour compenser certains biais
- On convertis le résultat en milisecondes

Pneus

La on arrive sur la partie la plus pénible.

Pour comprendre la problématique il faut d'abord faire un petit point sur comment les pneus fonctionnent en Formule 1.

Depuis 2019 en Formule 1 nous avons 5 grandes familles de pneus :

- Les pneus tendres
- Les pneus medium
- Les pneus durs
- Les pneus intermédiaires
- Les pneus pluie

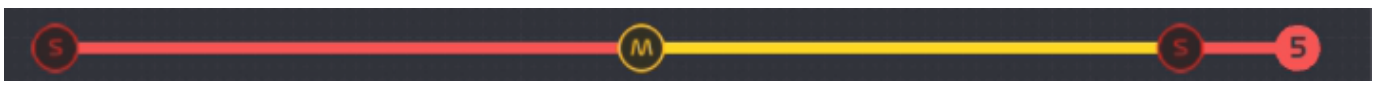
Les trois premiers pneus sont des pneus faits pour piste sèche, le pneu intermédiaire pour piste humide et le neu pluie pour la pluie.

Chaque pneu a sa durée de vie et son niveau de performance propre mais je ne vais pas rentrer dans le détail ici. Tout ce qu'il faut savoir ce que savoir sur quel pneu chaque pilote est et depuis combien de temps il les chausse est une information très importante.

Chaque pneu a une couleur donnée qui permet de les différencier.

Voici un exemple de ce à quoi une WINDOW de pneus peut ressembler :

Mais cette zone peut aussi ressembler à ca :



Mais aussi à ca :

Voire même ca :

Je pense que vous pouvez tout de suite comprendre la difficulté que représente la tâche de récupèration de données à partir de cette image.

En gros le fonctionnement de cette zone d'information est assez simple.

- Au fur et à mesure que la course avance, le trait fait de même.
- Le chiffre dans le round tout à droite indique le nombre de tour que le pilote a passé sur ce pneu.
- La couleur indique le type de pneu.
- Si il y a une lettre à la place d'un chiffre c'est que c'est le premier tour sur ce pneu. La lettre indique le type de pneu.

Et pas besoin de dire que si on essaie simplement de donner l'image à Tesseract on ne récupère ni les chiffres ni les lettres correctement si ce n'est pas du tout.

Il faut donc utiliser une methode qui permette d'isoler le rond le plus à droite, lui appliquer un traitement qui permette à Tesseract de lire ce qu'il y a marqué et qui puisse determiner quel pneu est en train d'être utilisé.

J'ai décidé de m'occuper dans un premier temps de trouver ce rond avant d'appliquer les filtres car plus l'image est petite plus les filtres sont rapides.

Le programme va tirer un trait depuis le bord droit de la zone, et il va avancer vers la gauche jusqu'à trouver un obstacle. Je détecte un obstacle si le pixel sur lequel est mon trait possède une valeur de plus de 0x50 dans le channel R,G ou B. J'ai trouvé en faisant des tests que les couleurs de background de la F1TV ne dépassaient jamais ces valeurs.

Ensuite après avoir trouvé le premier obstacle, je récupère une zone qui doit englober le cercle.

Voici un exemple avec cette image en entrée :

Elle est automatiquement coupée de cette facon :

Cela me permet d'isoler uniquement ce qui m'intéresse ce qui est très pratique pour Tesseract et pour la detection de couleur.

Ensuite avec cette image je peux commencer le processus de reconnaissance.

Je commence par faire une moyenne de tous les pixels de l'image en excluant les pixels trop sombres qui font sûrement partie du background ou du chiffre.

Ensuite j'utilise une methode qui calcule la différence entre la couleur obbtenue et la liste de couleurs possible.

Il y a cinq couleurs des pneus possibles :

"#ff0000" pneu tendre/soft

"#f5bf00" pneu medium

"#a4a5a8" pneu dur/hard

"#00a42e" pneu inter

"#2760a6" pneu pluie/wet

Ce qui est pratique c'est que même dans les cas ou il n'y a pas beaucoup de couleur comme celui la :

On arrive à une couleur moyenne de :

Et il est donc assez facile de determiner le type de pneu en question.

Attention, les résultats peuvent être très vite dérangés par la couleur du pneu précédent si le découpage de la fenêtre n'a pas été assez précis.

Ensuite il "suffit" de lire le chiffre dans le rond et si on arrive pas à le lire alors c'est que c'est une lettre et on sait que le nombre de tours est donc de 0.

Maintenant vient le moment très sympatique de la lecture du chiffre.

Vous saurez que Tesseract en plus de detester les grandes images et les images avec des couleurs, deteste également les formes dans une image. Donc dans notre cas, le round de couleur autour du chiffre, même si il n'est pas complet, il interfère avec la reconnaissance et empêche de bien lire le chiffre.

Il faut donc retirer le background et ensuite la couleur. Sauf que comme le chiffre est de la couleur du background, si on retire le background et ensuite la couleur il ne reste plus rien. Il faut donc retirer le background AUTOUR du rond, et ensuite si on retire la couleur il devrait rester le chiffre sur fond blanc.

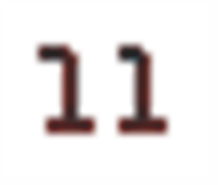
Pour se faire, j'ai tiré des traits depuis les bords de l'image jusqu'à ce qu'ils rencontrent le rond. Ensuite je retire tous les pixels entre le rond et les bords de l'image ce qui nous donne ceci :

Ensuite on peu retirer les pixels qui ont une valeur dans un channel RGB plus haute qu'un certain seuil :

Et la on a ce que l'on veut !

A partir de la c'est les filtres que l'on connait qui sont utilisés pour en faire une image plus facile à utiliser par Tesseract.

1 : On effectue un Resize de facteur 4 (oui c'est beaucoup mais en même temps le chiffre est vraiment petit à la base) qui permet d'avoir une image d'une bien meilleure résolution.



2: On fait une Dilatation de facteur 1 pour retirer tout le flou de l'image pour aider Tesseract

Et on a un chiffre qui est utilisable par Tesseract !

Explication des methodes précises plus bas

Pour résumer :

- On prend l'image de la zone et on la crop pour ne garder que la partie essentielle
- On détermine le type de pneu avec la couleur moyenne de la zone
- On retire le background autour de cette zone
- On retire la couleur qui reste pour ne garder que le chiffre
- On augmente la résolution du chiffre
- On rend ce chiffre net
- On envoie l'image traitée et filtrée à Tesseract
- On détermine le nombre de tours que le pilote a fait avec ses pneus avec le résultat de Tesseract

DRS

Bon ca c'était plutôt simple j'ai simplement vérifié si la moyenne de vert dépassait une certaine valeur et puis voila.

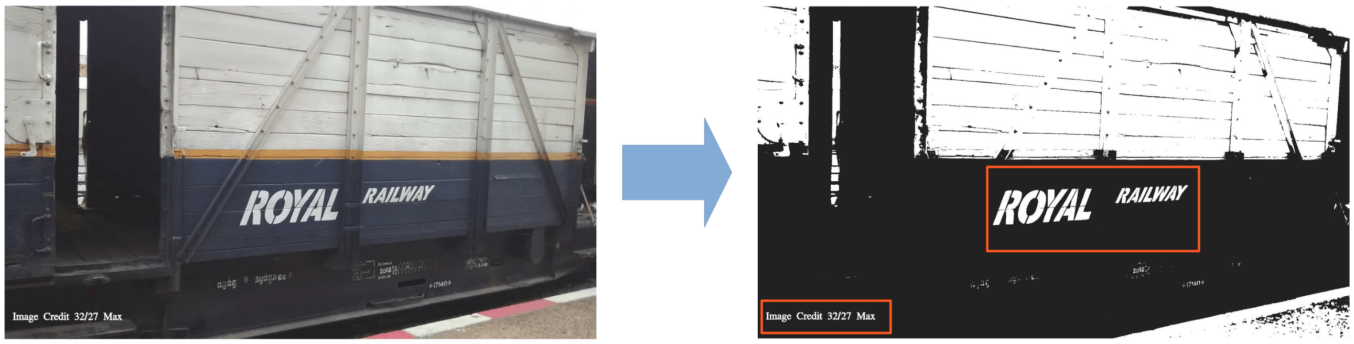
Filtres et methodes sur les images

Dans ce projet on a du utiliser différentes methodes d'édition d'image que ce soit sous forme de filtres ou de modification de l'image directement. Voici un sommaire des methodes utilisées et comment elles fonctionnent.

Tresholding

Cette methode sert à passer d'une image en couleurs à une image binaire noir blanc. C'est une étape très importante pour l'OCR car elle permet (si bien faite) d'isoler du texte de son background.

Un exemple ici :



Le fonctionnement est assez simple mais il peut être fait de différentes manières mais dans mon cas voici comment l'algorythme fonctionne sachant qu'il demande en entrée la Bitmap que l'on veut modifier ainsi que la valeur de Treshold :

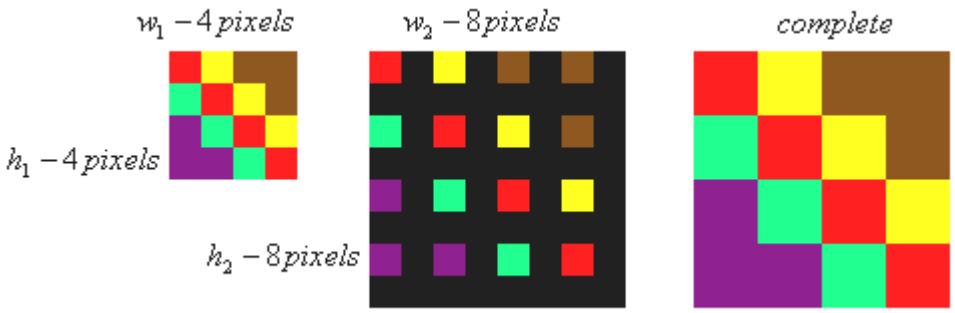
1. On parcours chaque pixel de l'image
2. On convertir la couleur du pixel en une valeur de gris pour avoir la même valeur en R,G et B (Formule utilisée : grey = R x 0.3 + G x 0.59 + B x 0.11)
3. Si le résultat de la valeur de gris est au dessus de la valeur de treshold, le pixel est passé en blanc complet et dans le cas contraire il est passé en noir complet
4. On retourne la Bitmap modifiée

Un algorythme pas forcément complexe mais qui peut augmenter de manière titanesque les chances de réussir une OCR

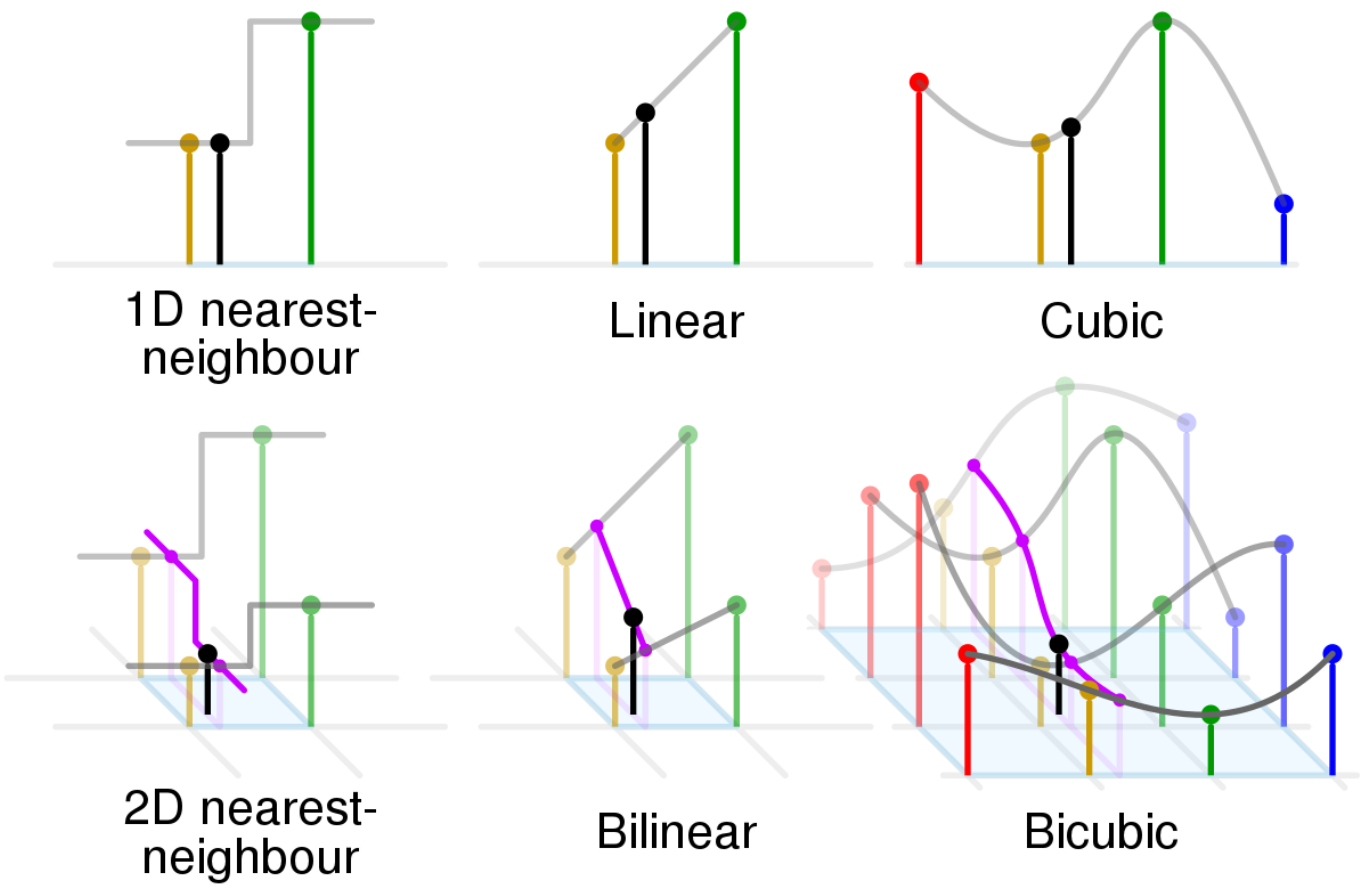
Resize

Cette methode sert à augmenter la résolution d'une image pour améliorer la précision de l'algorythme de Tesseract. En effet, avec trop peu de pixels, la matrice de convolution n'est pas toujours aussi efficace.

Il ne faut pas confondre cette methode d'augmentation de la taille avec une simple interpolation. En effet une augmentation de taille interpolée ne vas pas vraiment changer la résolution, l'image sera toujours aussi pixelisée, seulement, les pixels seront composées de plus de pixels comme dans l'exemple ci dessous :

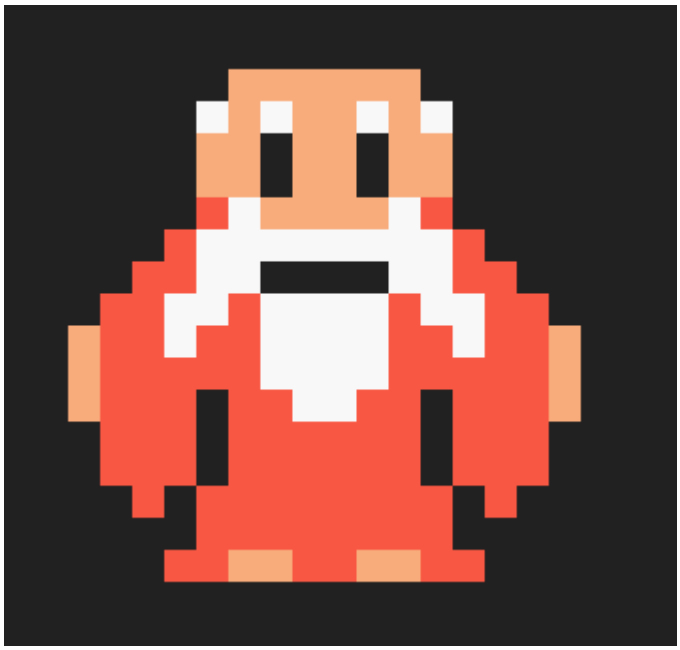


Dans mon projet j'utilise de l'interpolation bicubique qui va créer de l'information pour tenter de combler le vide et produire une image réellement plus grande et avec plus de details mais en ajoutant du flou.



Le but est d'aller chercher dans les pixels alentours les couleurs qui sont déja présente et de jouer avec des poids pour tenter de faire une prédiction de ce que ce pixel aurait été si l'image avait plus de detail.

Voici un exemple assez parlant :



On pourrait croire que c'est inutile mais dans le contexte de Tesseract ajouter des détails pour tenter de simuler une meilleure résolution même en créant du flou est intéressant pour mieux remplir la matrice de convolution.

Mais il est possible de réduire ce flou avec d'autres méthodes également.

(Dans mon code je n'ai pas utilisé du code fait main mais j'utilise une librairie qui me permet de le faire)

Il faut simplement faire attention car c'est un procédé assez lourd en performances.

Dilatation et Erosion

Cette methode et la suivante font partie des methodes de transformation morphologiques.

Ces methodes sont utilisées pour accentuer les formes et les epaissir ou les réduire et les affiner. Elles possèdent l'aventage également de retirer le flou d'une image ce qui est très pratique si utilisé après l'utilisation de methodes comme Resize.

Je ne vais pas trop rentrer dans les détails de ces methodes car leur fonctionnement est un peu plus lourd en math si on veut faire une véritable explication du pourquoi et du comment ca marche aussi bien. Pour notre projet je dirais que l'important est de savoir que ce sont deux outils très pratiques pour changer la morphologie des lettres et des chiffres et qu'on peut les utiliser pour corriger du flou et/ou des artéfacts apparus lors de la binarisation de l'image ou de la suppression de fond.

Remove Background

Cette methode est assez simple et est juste une methode qui va passer en revue tous les pixels de l'image et si la couleur d'un pixel s'apparente à celle d'un pixel de fond il est passé en noir total ou en blanc total. Le but est de permettre au reste du programme de fonctionner avec des couleurs moins ambigues.

Une variante spécialisée pour la reconnaissance des pneus appelée affectueusement Remove Useless cherche à atteindre le même bu mais est bien plus soffistiquée et spécialisée pour retirer le background autour d'un cercle de couleur pour ensuite retirer la couleur et qu'il ne reste qu'un chiffre. Pour plus de details voir la detection de pneus.

Il y aussi d'autre methodes comme un filtre Gaussien ou Highlight countour que j'ai du développer mais que je n'ai pas utilisé donc je ne vais pas en parler ici.

1.7.3 Interprétation des données

1.7.4 Stockage des données

1.7.5 Affichage des données

1.7.6 Prédictions

1.8 Tests

[A remplir au fur et à mesure de la création des tests]

1.9 Résumé des difficultés techniques

[A remplir au fur et à mesure dans la seconde moitié du travail de diplôme]

1.10 Améliorations futures

[A remplir dans les dernières semaines du travail de diplôme]

1.11 Conclusion

[A remplir la dernière semaine du travail de diplôme]

2. Cahier des charges

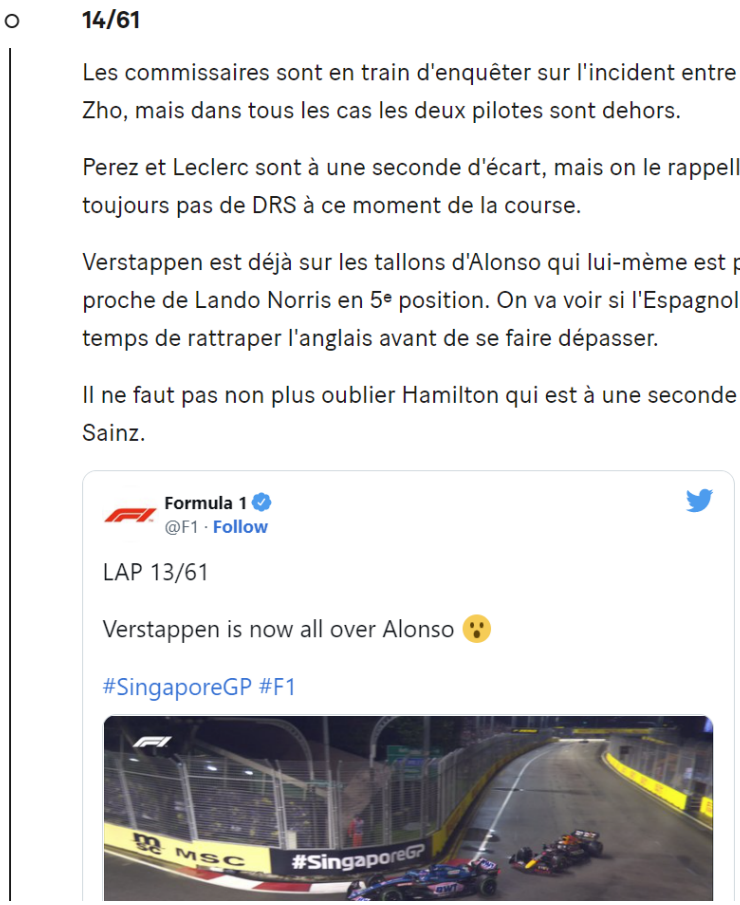
Cahier des charges "Track Trends" Travail de diplôme Maxime Rohmer 2023

2.1 Contexte

Je suis le "Live Ticker" chargé de la Formule 1 pour le 20 minutes. On peut traduire cela comme commentateur de F1, avec tout de même l'importante subtilité que je ne commente pas avec la voix, mais avec le clavier. Mes commentaires sont sous la forme de commentaires écrits live qui s'ajoutent au fur et à mesure de l'évènement. Par exemple : "Tour 28/54, Hamilton a fini par s'arrêter et chausser des gommes tendres 13 tours après Verstappen. L'Anglais va voir plus de 15 secondes à rattraper, mais les gommes neuves et plus tendres que son rival devraient lui permettre s'il ne se fait pas trop ralentir par le trafic". En général avec un peu plus d'infos quand même et cela tous les 3-4 tours

Voici quelques exemples de précédents commentaires (Conseil : il y a un bouton pour montrer le feed dans l'ordre chronologique) :

- "Commentaire Grand Prix de Belgique 2022"
- "Commentaire du Grand Prix de Singapour 2022"



Pendant un Grand Prix, je dois constamment :

- Écrire ce qu'il se passe dans le grand prix et expliquer les enjeux
- Chercher régulièrement des médias à inclure pour diversifier mon live (Tweets, Images etc.)
- Changer le titre et la description du live en fonction de l'évolution du Grand prix
- Et accessoirement regarder le grand prix pour y comprendre quelque chose

Avec tout ça, il est très difficile de garder un œil sur la page DATA de la F1TV qui fournit pourtant des informations précieuses.

Je me retrouve parfois par exemple à ne pas parler de dépassements dans le peloton, car ils ne sont pas retransmis à la télé alors que c'est une information importante. Autre exemple, occasionnellement le classement ne reflète pas les vraies positions des pilotes. Les arrêts aux stands font que du coup des pilotes qui devraient être 15èmes se retrouvent 8e puisqu'ils ne sont pas encore arrêtés. Cela peut de temps en temps prêter à confusion.

2.2 Projet

Un outil de style compagnon sous forme d'application C# Windows Form qui récupère en temps réel les informations de la course et affiche les informations les plus importantes. Le but est non seulement de faciliter mon job, mais aussi faire en sorte d'améliorer la plus-value de mon travail en me permettant de fournir des commentaires qui ne sont pas disponibles pour le tout venant qui regarde simplement le flux RTS.

Exemples:

- Les pilotes qui sont proches (moins de 1-2 secondes qui sont donc en train de se battre).
- Les pilotes qui améliorent leur temps au tour et ceux qui perdent le plus de temps
- Le classement pondéré tenant compte des futurs arrêts au stand

Maintenant afficher différemment les infos, c'est sympa, mais cela serait encore mieux de traiter ces data et de permettre des petites prédictions.

Exemples :

- Prédire les arrêts aux stands en prenant en compte les baisses de performances des pneus
- Prédire le pneu que le pilote va chausser s'il rentre aux stands dans le prochain tour
- Prédire dans combien de tour tel pilote va rattraper tel autre pilote
- Prédire combien de temps le pilote va perdre dans les stands en fonctions des arrêts précédents

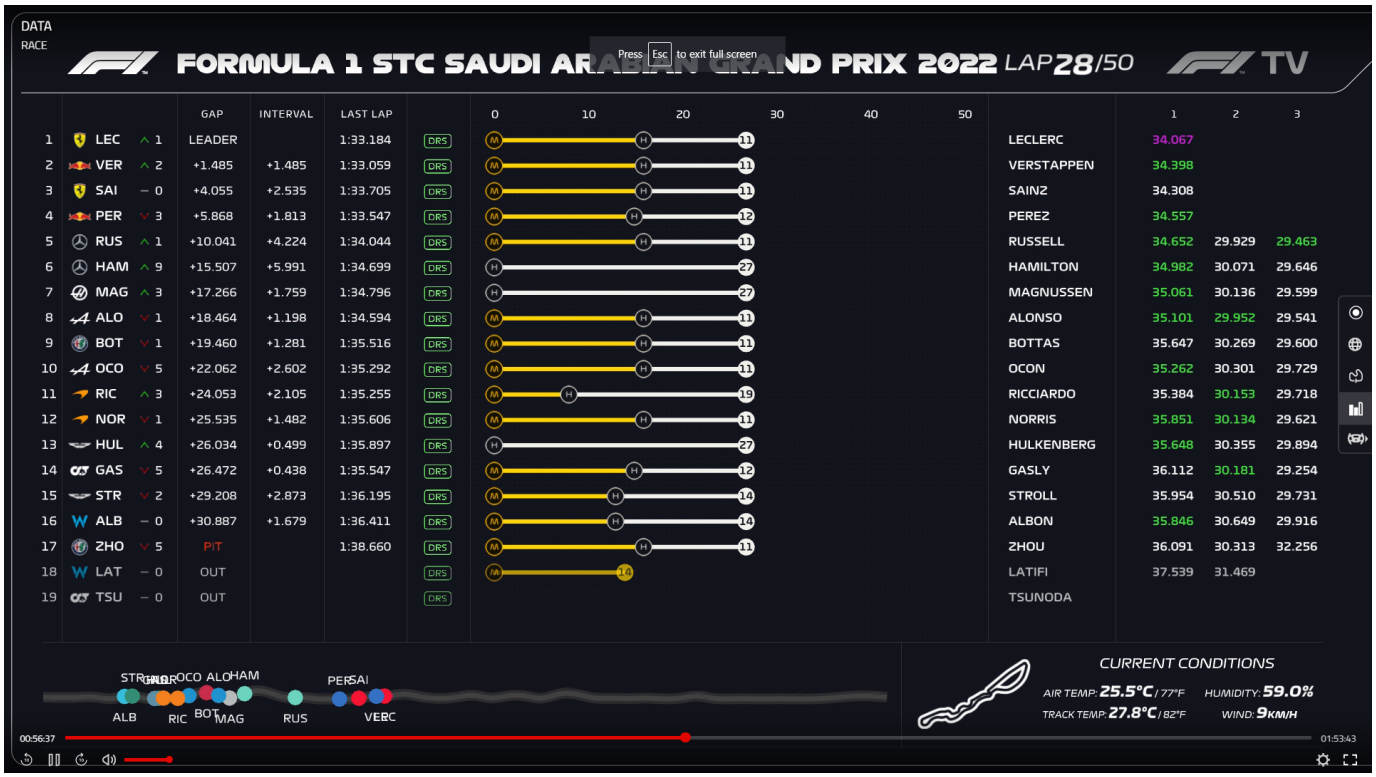
2.3 Réalisation

Malheureusement, la Formula 1 Management ne propose aucune API publique qui puisse nous permettre de faire ce projet "simplement". La raison la plus probable étant qu'Amazon avec son service AWS propose exactement ce genre de services pour le flux télévisé et il doit y avoir un contrat d'exclusivité.

Il existe des API "Pirates" faites par la communauté, le problème est qu'elles ne sont pas forcément des plus pratiques à utiliser.

Mais comme je possède un abonnement premium ++ à la F1TV, j'ai accès pour chaque grand prix à un flux vidéo nommé : DATA F1 CHANNEL

Qui ressemble à ça :



Donc la seule façon que je vois de récupérer ces données est de les prendre directement sur ce feed.

Même si le but final de l'application est de faire pleins de choses super avec les datas, le gros du projet va surtout être la récupération des données et leur stockage.

Les données viennent du flux vidéo et ainsi dans un premier temps, il va falloir récupérer d'une manière ou d'une autre des images qui viennent d'un grand prix en direct ou en rediffusion.

Ensuite, dans un second temps, il faut lire les informations directement sur l'image en utilisant une librairie prévue pour (exemple Tesseract) et vérifier l'intégrité de ces dernières pour qu'on puisse ensuite les stocker.

Dans un troisième temps, il faut stocker toutes ces données dans une forme qui permette d'aller facilement faire des requêtes de récupération et déjà préparer des méthodes qui permettent de récupérer des infos importantes (ex : la moyenne des cinq derniers tours, le temps moyen d'arrêt etc.) pour faciliter la dernière étape

Quand tout cela est fait, on peut ensuite s'amuser un peu avec les Data.

La dernière étape est donc l'affichage. L'idée est de créer une Windows Form qui contienne toutes ces informations dans un format beaucoup plus lisible et avec laquelle on pourrait interagir pour permettre de plus facilement commenter les Grands Prix. (exemple plus bas avec un croquis de ce à quoi l'application pourrait ressembler)

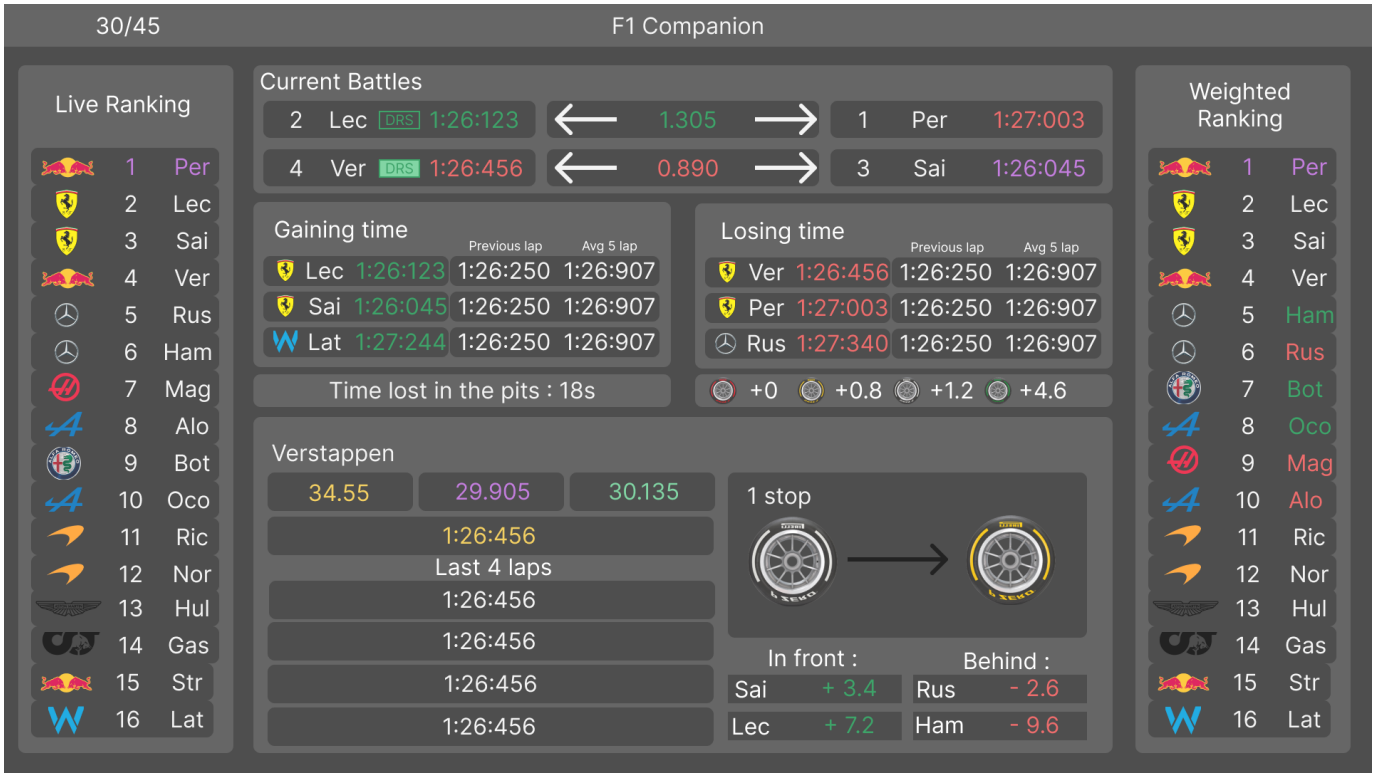
Voici la liste des données qui pourraient être affichées (Non contractuel, simplement des idées).

- Les pilotes qui sont proches (moins de 1-2 secondes qui sont donc en train de se battre).
- Les pilotes qui améliorent leur temps au tour et ceux qui perdent le plus de temps
- Le classement pondéré tenant compte des futurs arrêts au stand
- La moyenne de temps que les pilotes perdent dans les stands
- La performance moyenne des 5 types de pneus
- La moyenne de temps de chaque pilote sur le pneu actuel
- Le nombre de points que chaque pilote gagnerait selon sa position
- Le classement de la course

Voire même si c'est possible :

- Prédire les arrêts aux stands en prenant en compte les baisses de performances des pneus
- Prédire le pneu que le pilote va chausser s'il rentre aux stands dans le prochain tour
- Prédire dans combien de tour tel pilote va rattraper tel autre pilote
- Prédire combien de temps le pilote va perdre dans les stands en fonctions des arrêts précédents
- Prédire les temps au tour de chaque pilote selon l'usure des pneus

Voici un exemple d'interface possible pour une page :



2.4 Cas d'utilisation

*On va considérer que tous les user ont un abonnement F1 TV PRO

Un user veut récupérer les data :

- Il ouvre son navigateur et lance la page DATA de la F1 TV
- Il calibre la capture des data via le programme (pour la première utilisation).
- Il confirme que les données initiales sont bonnes (pour la première utilisation).
- Il regarde tranquille son Grand Prix

Le programme récupère les data :

- Il récupère des images depuis la F1TV
- Il utilise Tesseract (ou autre) pour en récupérer les infos.
- Il met ces infos dans un Objet Pilote, dans un Objet course avec un attribut tour pour hiérarchiser les data

Pour ce qui est de l'affichage, l'idée est de faire une application C# comme on l'a appris à l'école, mais avec assez de style pour qu'elle puisse être agréable à utiliser.

Quand le programme affiche les data :

- Il prend les données venant directement de la F1TV.
- Il affiche différemment les données pour permettre une meilleure lisibilité
- Il interprète avec des règles plutôt simples certaines data pour faire des miniprédictions ou aider à la lecture
- Il récupère des infos d'autres courses pour les comparer et proposer des prédictions plus intéressantes

2.5 Difficultés techniques

- Récupérer un flux vidéo plutôt propre malgré les contres mesures de la F1 TV pour en empêcher la lecture par un logiciel
- Si on doit passer par une capture d'écran, trouver un moyen de stocker les données de manière à prévoir que parfois un tour pourrait avoir plus de données qu'un autre, que le user peut mettre pause, ou même qu’il revienne en arrière.
- Développer des algorithmes pour récupérer les données comme les différents pneus utilisés ou l'activation du DRS ainsi que développer des moyens de nettoyer les résultats de l'OCR (Par exemple utiliser différents champs redondants pour comparer les résultats)
- Stocker les données sur une base pour les traiter plus tard tout en prévoyant un moyen de voir les stats live
- Développer des algorithmes de prédiction qui prennent en compte d'anciennes courses pour tenter de prédire des choses comme les arrêts aux stands par exemple.

3. Journal de bord

3.1 Mercredi 29 Mars 2023

Premier jour du travail de diplôme. Nous avons eu un briefing de mr Garcia et nous avons pu commencer à préparer le travail.

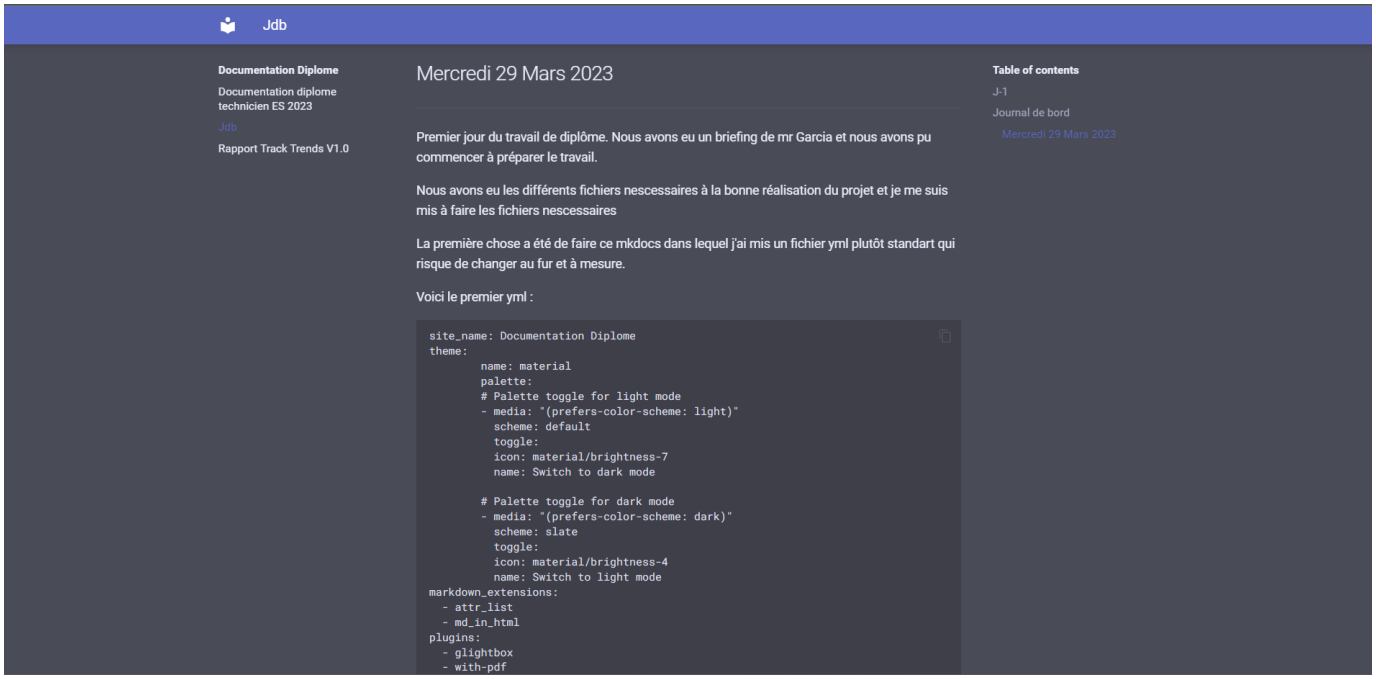
Nous avons eu les différents fichiers nescessaires à la bonne réalisation du projet et je me suis mis à faire les fichiers nescessaires

La première chose a été de faire ce mkdocs dans lequel j'ai mis un fichier yml plutôt standart qui risque de changer au fur et à mesure.

Voici le premier yml :

```
site_name: Documentation Diplome
theme:
        name: material
        palette:
        # Palette toggle for light mode
        - media: "(prefers-color-scheme: light)"
          scheme: default
          toggle:
          icon: material/brightness-7
          name: Switch to dark mode

        # Palette toggle for dark mode
        - media: "(prefers-color-scheme: dark)"
          scheme: slate
          toggle:
          icon: material/brightness-4
          name: Switch to light mode
markdown_extensions:
  - attr_list
  - md_in_html
plugins:
  - glightbox
  - with-pdf
```

Voici un example de à quoi ca ressemble en forme de site



Ensuite il m'a fallu faire une version plus à jour de mon cahier des charges car je n'y avait pas touché depuis novembre. J'ai envoyé un mail à mes enseignants pour qu'ils puissent y jeter un oeuil pour être sûr que je n'ai rien changé qui les dérangent.

Monsieur Jayr m'a demadé à l'occasion de lui faire un planning type Gantt alors je me suis mis à la tâche.

J'ai fait un planning prévisionnel et une légende les deux sont dispo dans le dossier planning de ce repertoire.

Ensuite je me suis mis à tout mettre sur git. A commencer par ce repertoire

Et c'est deja la fin de la journée ! Demain j'avance un peu sur la doc avec ce que je peux déja remplir et je finis de préparer ce dont j'ai besoin pour commencer à coder.

3.2 Jeudi 30 Mars 2023

Aujourd'hui selon le planning je dois me charger des dernirers préparatifs pour commencer correctement. J'ai fait exprès de prenre du temps pour ca au début pour ne pas me créer de soucis plus loin pendant le travail.

Je vais envoyer par mail le planning que j'ai fait à mes suiveurs.

Ensuite je vais m'attaquer au squelette de la docmentation. Je vais essayer de remplir tout ce que je peux remplir dans un premier temps car cela tout ca de fait pour plus tard quitte à modifier quelques aspects au fur et à mesure.

J'ai aussi désactivé mkdocs with pdf par ce que les résultats ne sont vraiment pas ceux que j'attends et cela ralentis énormément le déploiment.

J'ai aussi rassemblé mes croquis pour le poster :

"Croquis Poster 1" "Croquis Poster 2"

On peut voir que dans un premier temps j'ai tenté de faire un poster un peu plus stylisé et marketing. Cependant après avoir discuté avec Mr Garcia et différents profs dont un de l'HEPIA et ils m'ont indiqué que ce qui était attendu était moins du marketing qu'un diagramme de fonctionnement.

On peut voir sur les derniers posters que le coté technique ressort de plus en plus. Le but sera de faire une version encore plus technique ou on peut voir les différents fonctionnements de l'application avec les technologies utilisées.

Le défi cela va être de faire un joli poster qui soit en même temps vendeur et en même temps rempli techniquement.

Oh et j'ai eu un problème ou mon calvier et ma souris ne voulaient d'un coup plus fonctionner. Dans mon cas c'était un problème de power management des ports. J'ai eu le soucis sur mon pc fixe à la maison et sur mon pc portable également. En gros de ce que j'ai compris le soucis c'est que le pc croit que un port est trop solicité niveau puissance et du coup décide de couper l'alimentation du port USB.

J'ai pu règler le soucis en allant dans le device manager sous universal bus controller sous power management et en décochant la case qui indique que windows peut désactiver ce port.

Je ne conseille pas ce fix si vous avez des composants de mauvaise qualité car cela pourrait être une vraie alerte cependant le fait que mes composants sont plutôt haut de gamme et le fait que mon clavier et ma souris le fassent en même temps et que ils fonctionnaient très bien depuis plus de 4 ans me font penser que c'est juste une nouvelle mise a jour de windows qui est pénible.

Demain je vais pouvoir commencer à coder pour de bon.

3.3 Vendredi 31/03/2023

Aujourd'hui on s'occupe de la PT2 qui est la programmation de la récupèration des informations des images.

Je vais tester IronOcr

Source : https://www.c-sharpcorner.com/article/ocr-using-tesseract-in-C-Sharp/ Doc : https://ironsoftware.com/csharp/ocr/docs/ Examples : https://ironsoftware.com/csharp/ocr/examples/simple-csharp-ocr-tesseract/

Avant d'utiliser la librairie je me demande si je dois utiliser un peu de post processing pour aider à la reconnaissance.

Je peux soit utiliser l'image cropée directement :

Soit avec un filtre pour passer en noir et blanc laxiste

Soit avec un filtre pour passer en noir et blanc stricte

Il va falloir faire des tests avec tous les noms et les chiffres pour trouver le plus efficace.

Bon malheureusment Iron OCR semblait être une bonne alternative mais c'est une librairie privée qui demande une license pour être utilisée. Il va falloir trouver autre chose.

En utilisant la librairie "Tesseract" qui existe on peut faire de la reconnaissance de texte avec un code plutôt simple :

```
TesseractEngine engine = new TesseractEngine(tessDataFolder.FullName,"eng", EngineMode.Default);

var tessImage = Pix.LoadFromMemory(ImageToByte(sample));

Page page = engine.Process(tessImage);
string text = page.GetText();
```

Voici la methode ImageToByte : https://stackoverflow.com/questions/7350679/convert-a-bitmap-into-a-byte-array

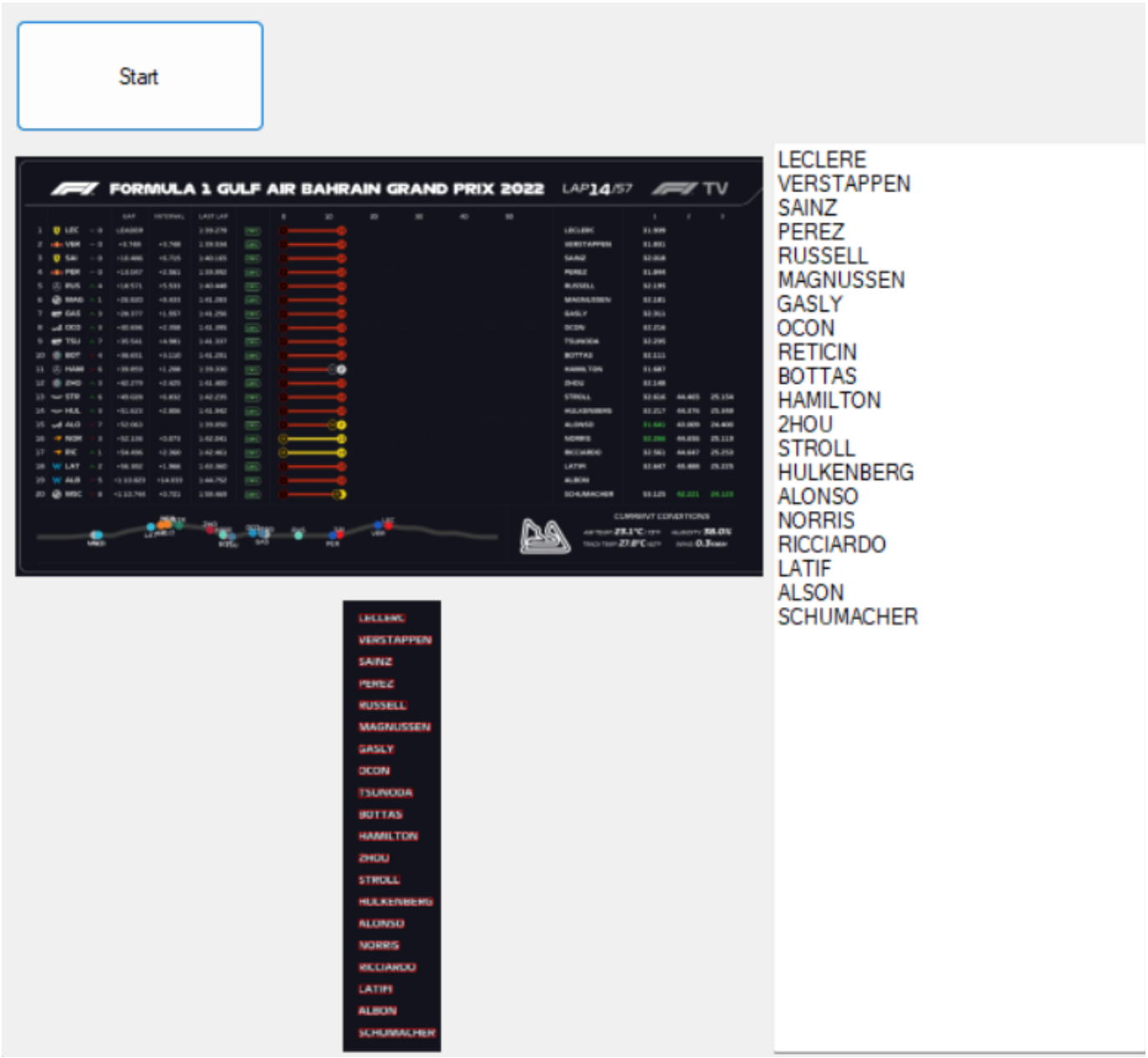
```
public static byte[] ImageToByte(Image img)
{
    using (var stream = new MemoryStream())
    {
        img.Save(stream, System.Drawing.Imaging.ImageFormat.Png);
            return stream.ToArray();
        }
}
```

Voici le code pour traiter plusieurs textes sur une seule image :

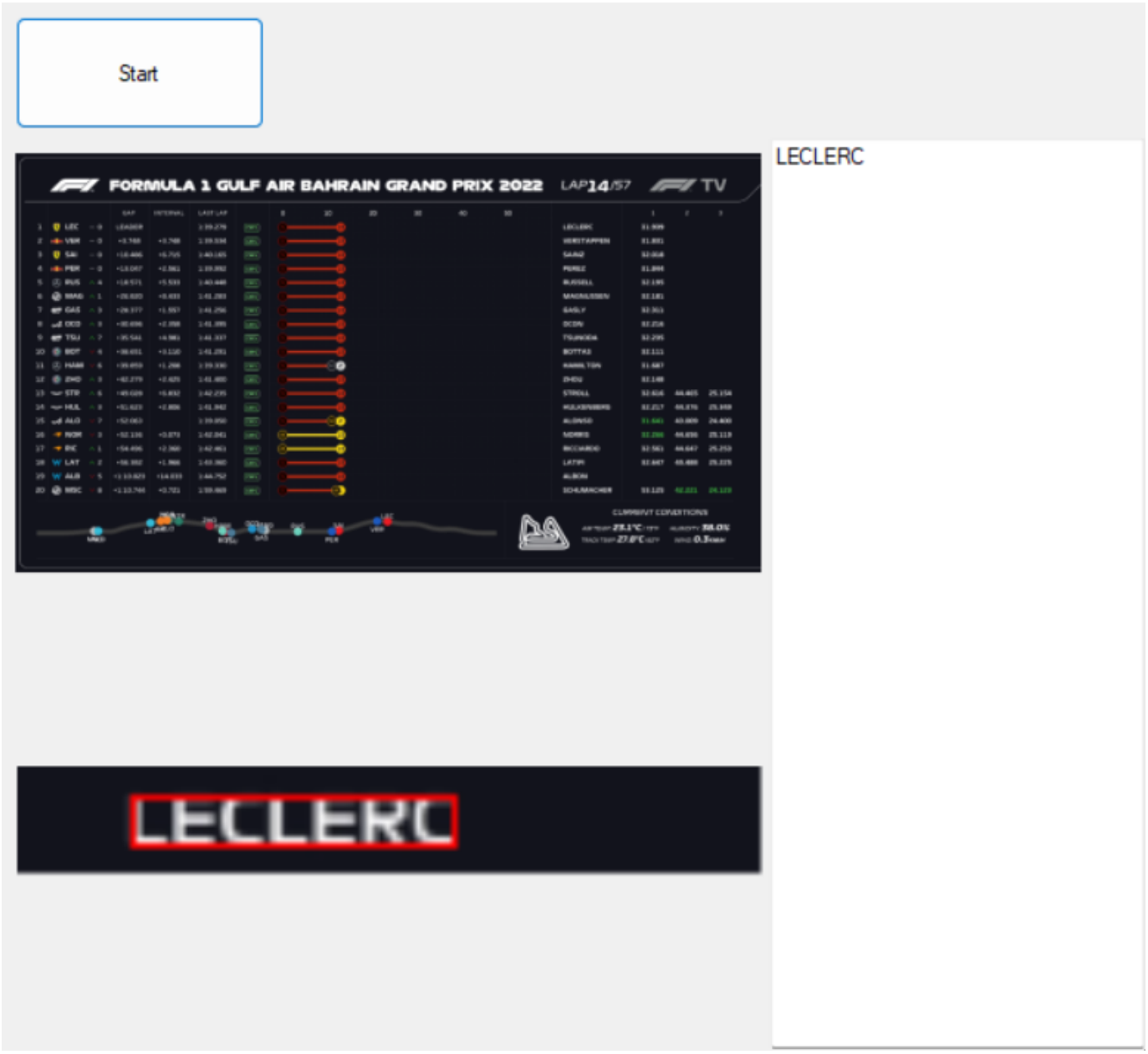
```
Page page = engine.Process(tessImage);
            // Get the iterator for the page layout
            using (var iter = page.GetIterator())
            {
                // Loop over the elements of the page layout
                iter.Begin();
                do
                {
                    // Declare a Rect variable to hold the bounding box
                    Rect boundingBox;

                    // Get the bounding box for the current element
                    if (iter.TryGetBoundingBox(PageIteratorLevel.Word, out boundingBox))
                    {
                        g.DrawRectangle(Pens.Red,new Rectangle(boundingBox.X1,boundingBox.Y1,boundingBox.Width,boundingBox.Height));
                    }

                    // Get the text for the current element
                    var text = iter.GetText(PageIteratorLevel.Word);
                    tbxResult.Text += text.ToUpper() + Environment.NewLine;
                } while (iter.Next(PageIteratorLevel.Word));
            }
```

Etonnament, avec plus de texte, des noms qui étaient autrefois mal reconnus sont parfaitement interprêtés.

Par exemple voici un exemple de reconnaisance de texte sur tous les pilotes :



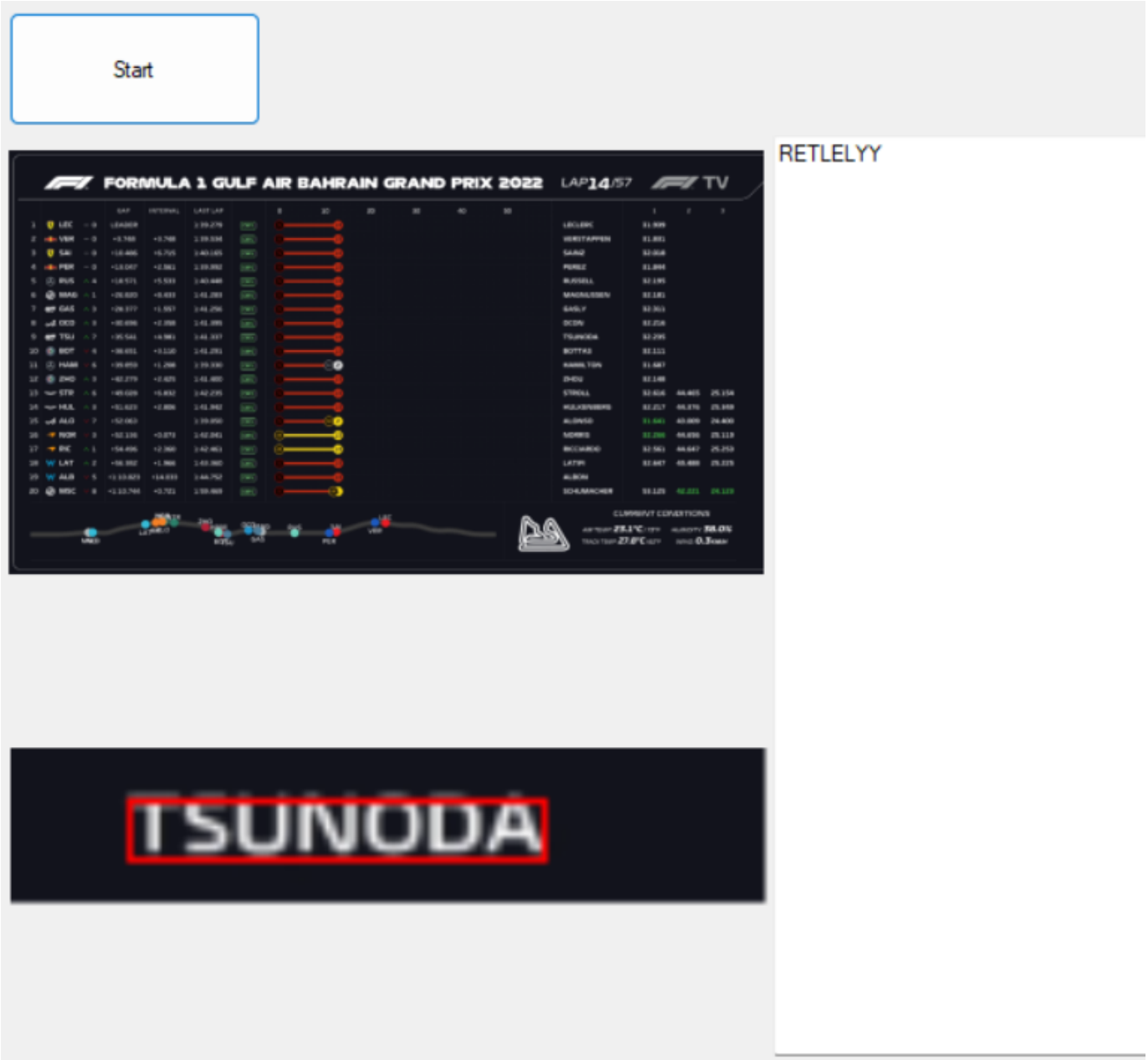
On voit que le nom Leclerc est mal reconnu. Mais voici ce que cela donne quand on prend une image qui ne contient que le nom Leclerc :



On voit ici que le nom Leclerc est très bien reconnu.

Dans le premier exemple on peut voir que Tsunoda est reconnu comme "Reticin" ce qui n'est pas exactement pareil (lol)

Et quand on isole le nom Tsunoda dans une image seule :



Il le lit "RETLELYY" ce qui n'est toujours pas exactement ca...

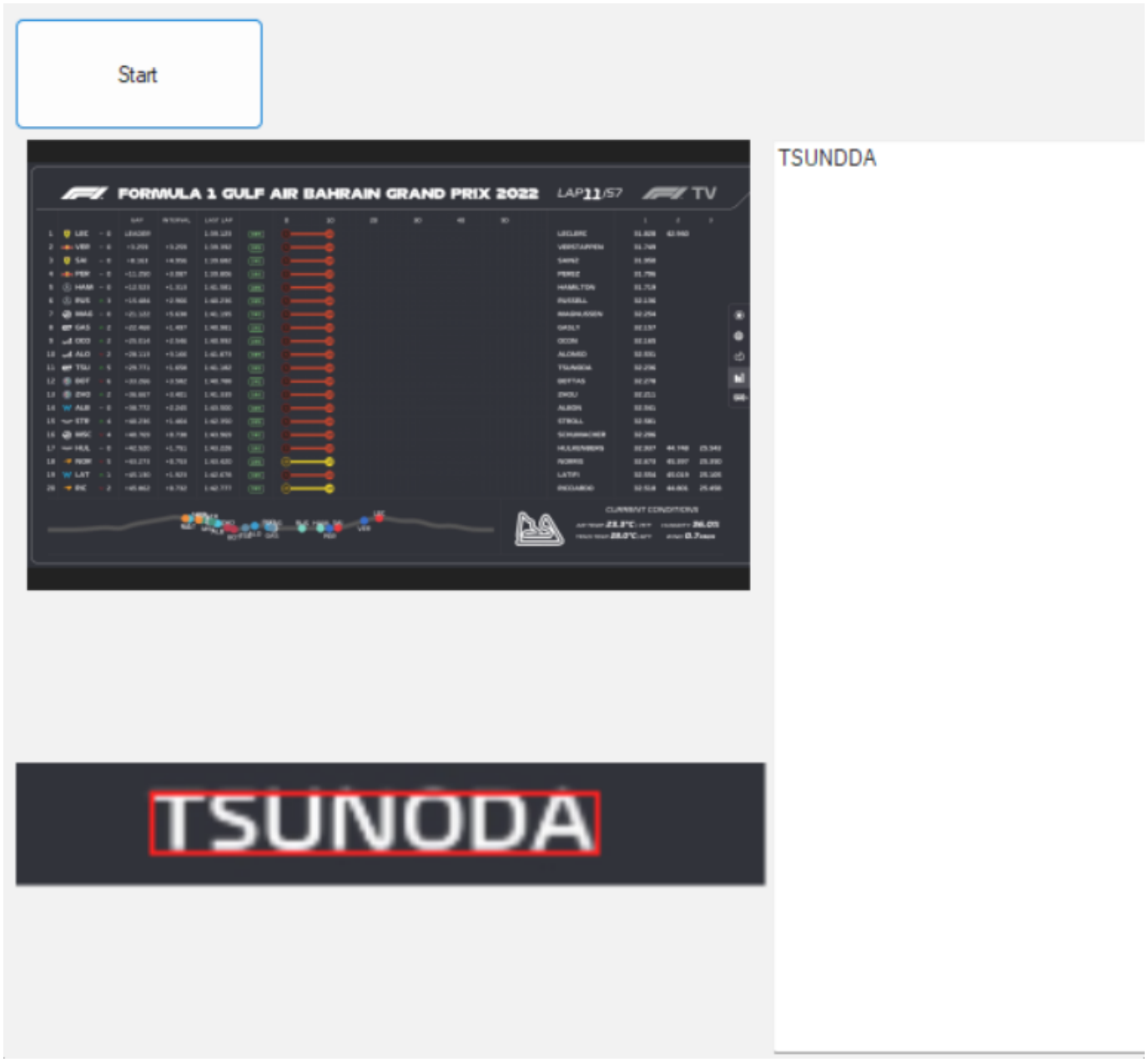
Une meilleure résolution pourrait peut-être résoudre le problème en partie.

Jusqu'ici les images étaient en presque 720P ce qui donne ceci :

Et j'ai lancé une récupèration d'images en 1080p pour récupèrer ceci :

On peut voir une certaine différence tout de même.

Et quand on lance la reconnaissance :



"Tsunoda n'est plus écrit "RETLELYY" mais "TSUNDDA" ce qui n'est pas parfait mais qui est déja beaucoup mieux.

J'ai essayé de mettre l'engine de Tesseract en mode "JPN" comme Tsunoda est un nom japonais mais sans succès j'ai le même résultat.

Comme la résolution est meilleure je me suis dit que peut être le filtre de passage en noir et blanc pourrait aider.

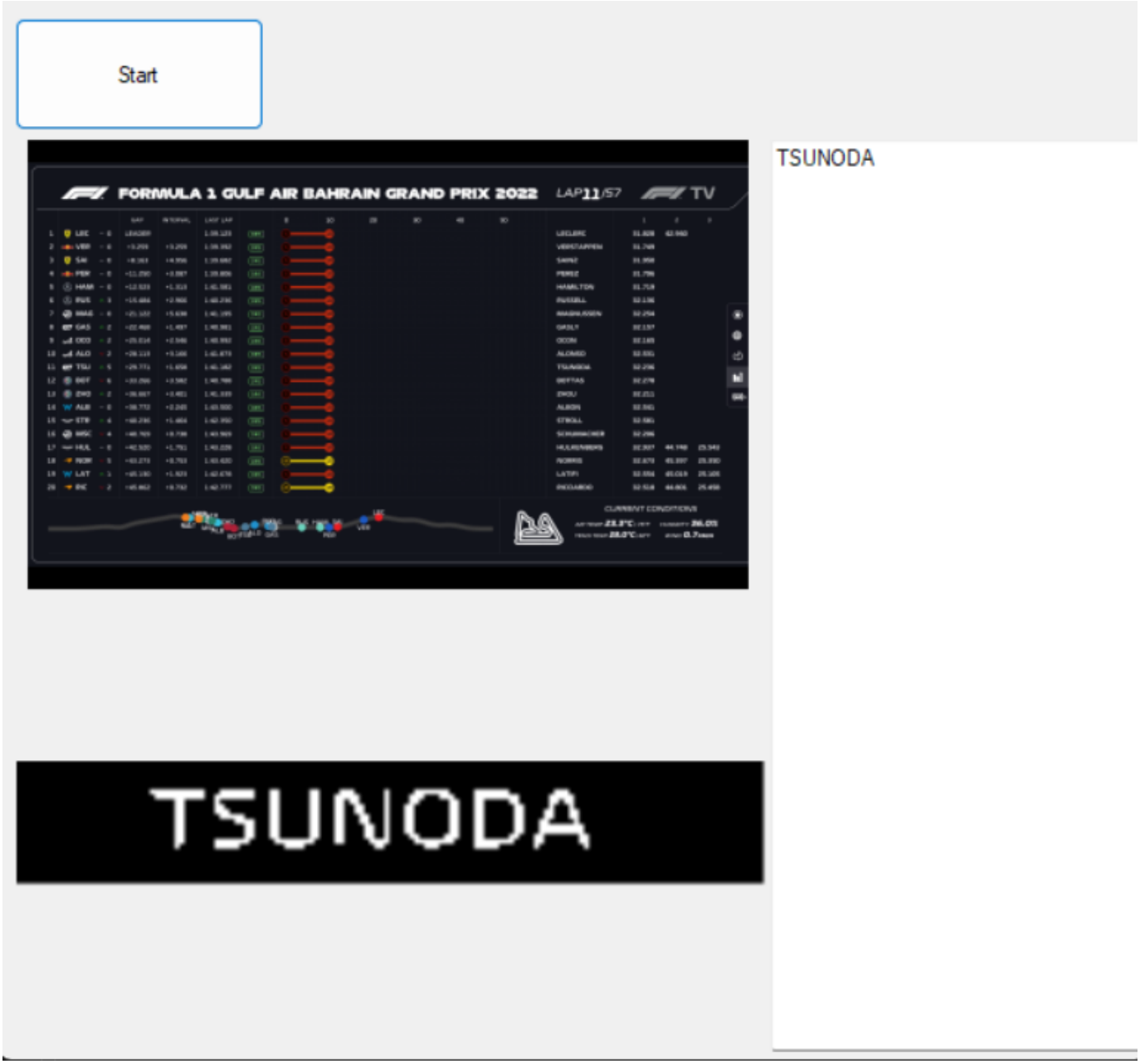
J'ai écrit cette petite methode pour convertir l'image en noir et blanc :

```
private static Bitmap ConvertToBlackAndWhite(Bitmap inputBmp)
        {
            const int BLACK_TO_WHITE_TRESHOLD = 200;
            Bitmap result = new Bitmap(inputBmp.Width, inputBmp.Height);

            for (int y = 0; y < inputBmp.Height; y++)
            {
                for (int x = 0; x < inputBmp.Width; x++)
                {
                    Color pixelColor = inputBmp.GetPixel(x,y);
                    if (pixelColor.R <= BLACK_TO_WHITE_TRESHOLD && pixelColor.G <= BLACK_TO_WHITE_TRESHOLD && pixelColor.B <= BLACK_TO_WHITE_TRESHOLD)
                    {
                        pixelColor = Color.FromArgb(0,0,0);
                    }
                    else
                    {
                        pixelColor = Color.FromArgb(255,255,255);
                    }
                    result.SetPixel(x,y,pixelColor);
                }
            }
            return result;
        }
```

Rien de bien dingue mais cela fonctionne et je peux jouer avec le BLACK_AND_WHITE_TRESHOLD pour changer son comportement.

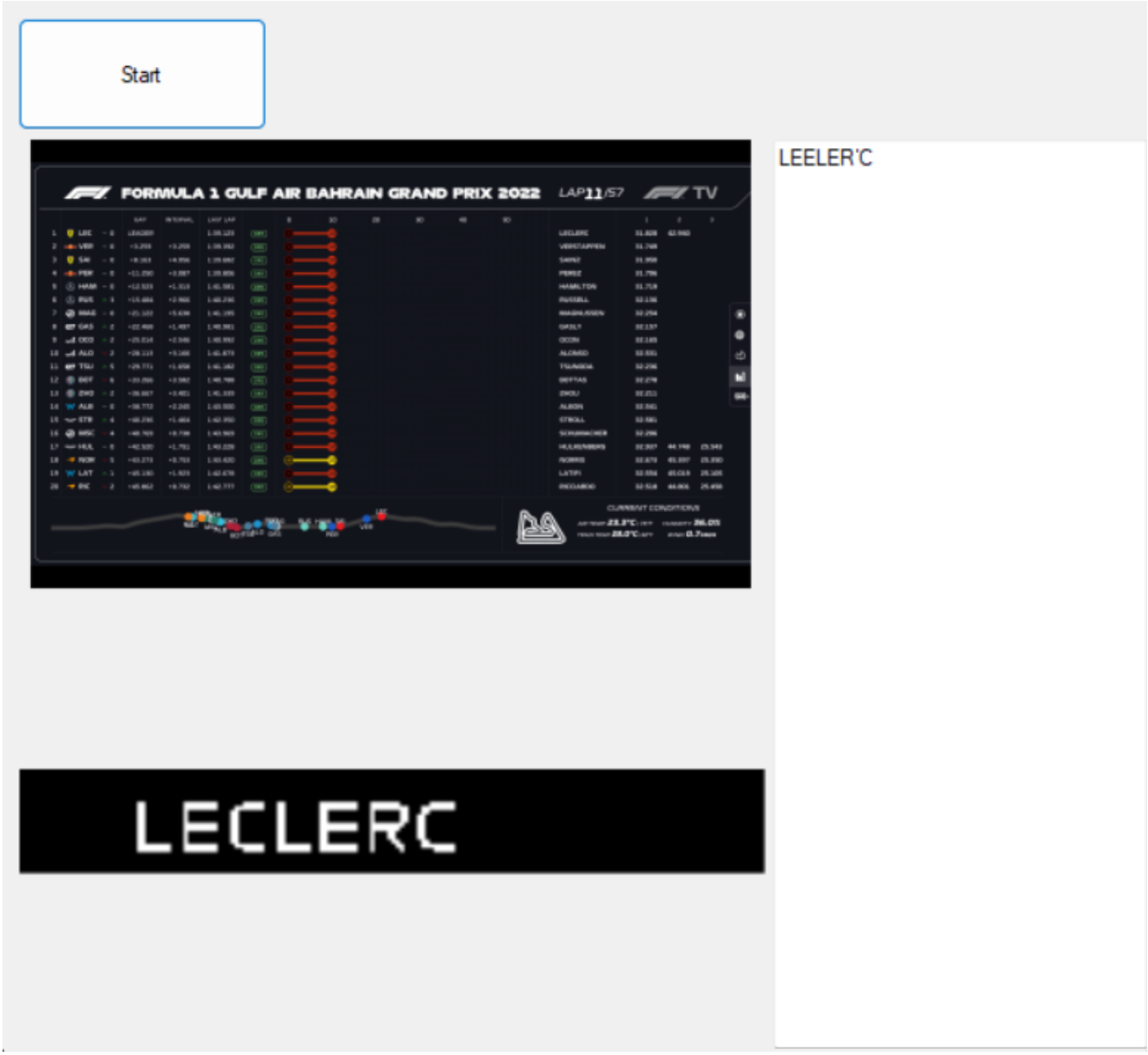
J'ai dabord testé avec un treshold de 100 et le programme a réussi à me sortir Tsunoda en deux mots ce qui était déja très encourageant.

Et après avoir augmenté le Treshold... Tada :



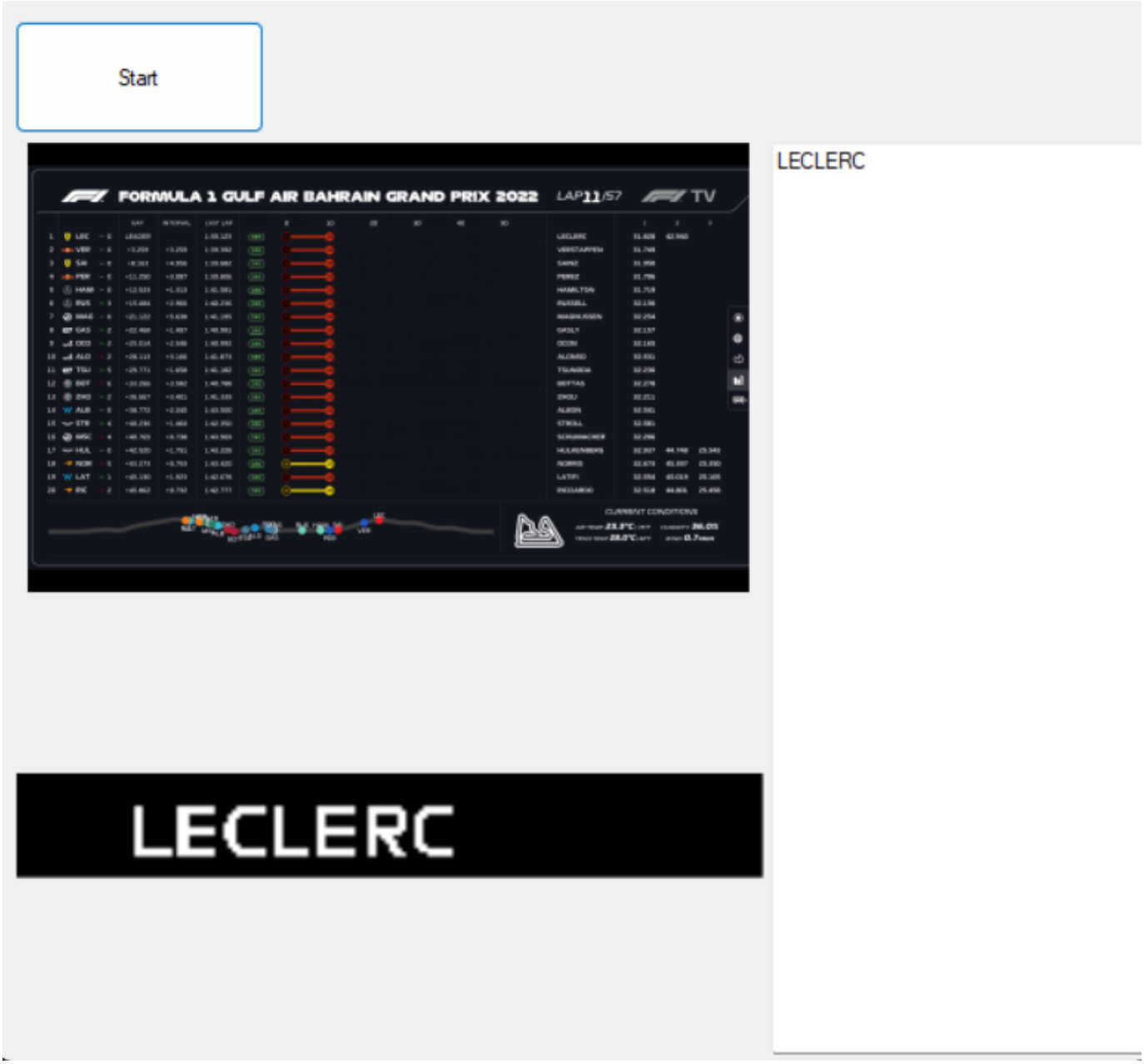
Le programme arrive bien à reconnaitre TSUNODA. Je pense que cette tactique ne fonctionnait pas avant car la resolution était trop faible et l'aliasing se mêlait trop avec le texte pour être utilisable.

Cependant cette technique ne fonctionne pas sur tous les noms. Par example avec Leclerc :



On récupère "Leeler'c" ce qui n'est pas bon du tout.

Mais en modulant le Treshold (ici à 150) On peut de nouveau voir Leclerc être reconnu correctement



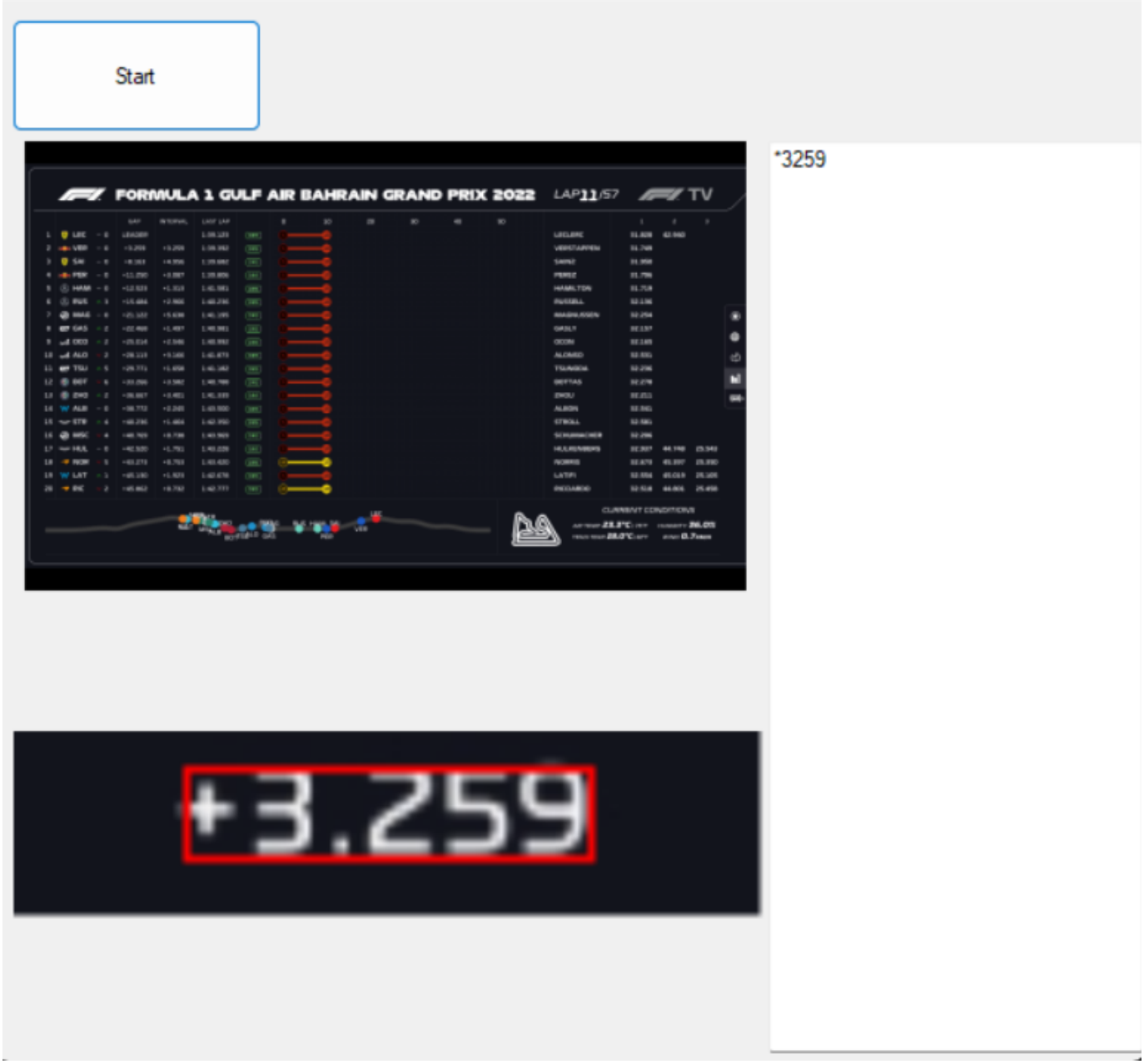
Je pense que pour avoir de bons résultats il va falloir faire un algo qui :

- Découpe l'image en autant de plus petites images pour avoir un mot par image.
- Teste voir si avec l'image originale un nom correspond à la liste de pilotes existant.
- Si cela ne marche pas, on applique le filtre en modulant le Treshold.
- Dans le cas ou on aurait pas un match parfait on fait un algo qui cherche le nom le plus proche qui existe dans la liste de noms donnés.

Seulement voila, il n'y a pas que des lettres que l'on veut récupèrer. On veut surtout pouvoir récupèrer les chiffres.

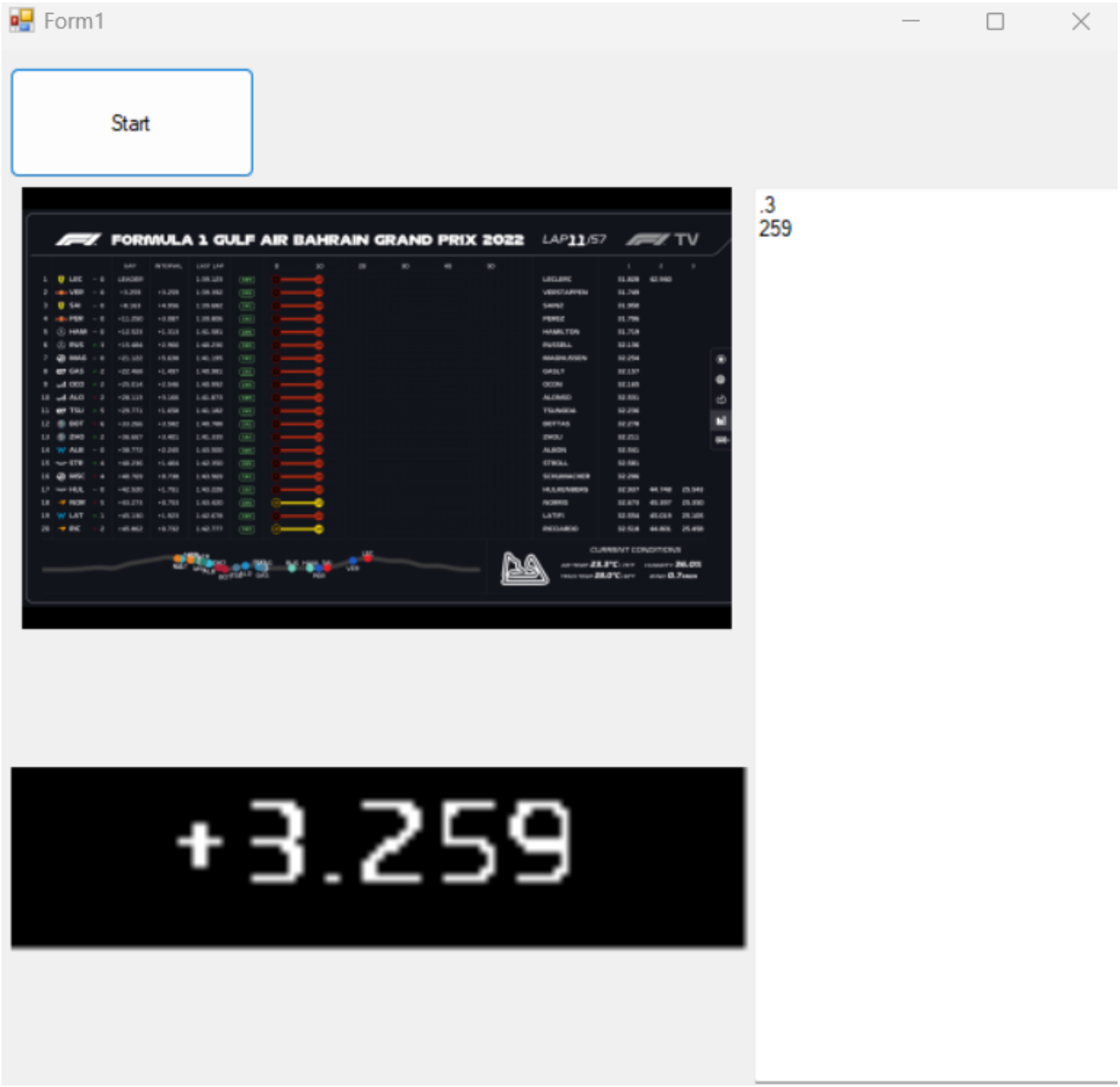
Pour les chiffres on va avoir des soucis également...

Si on essaie directement la même technique sans filtre on a des résultats comme celui ci :



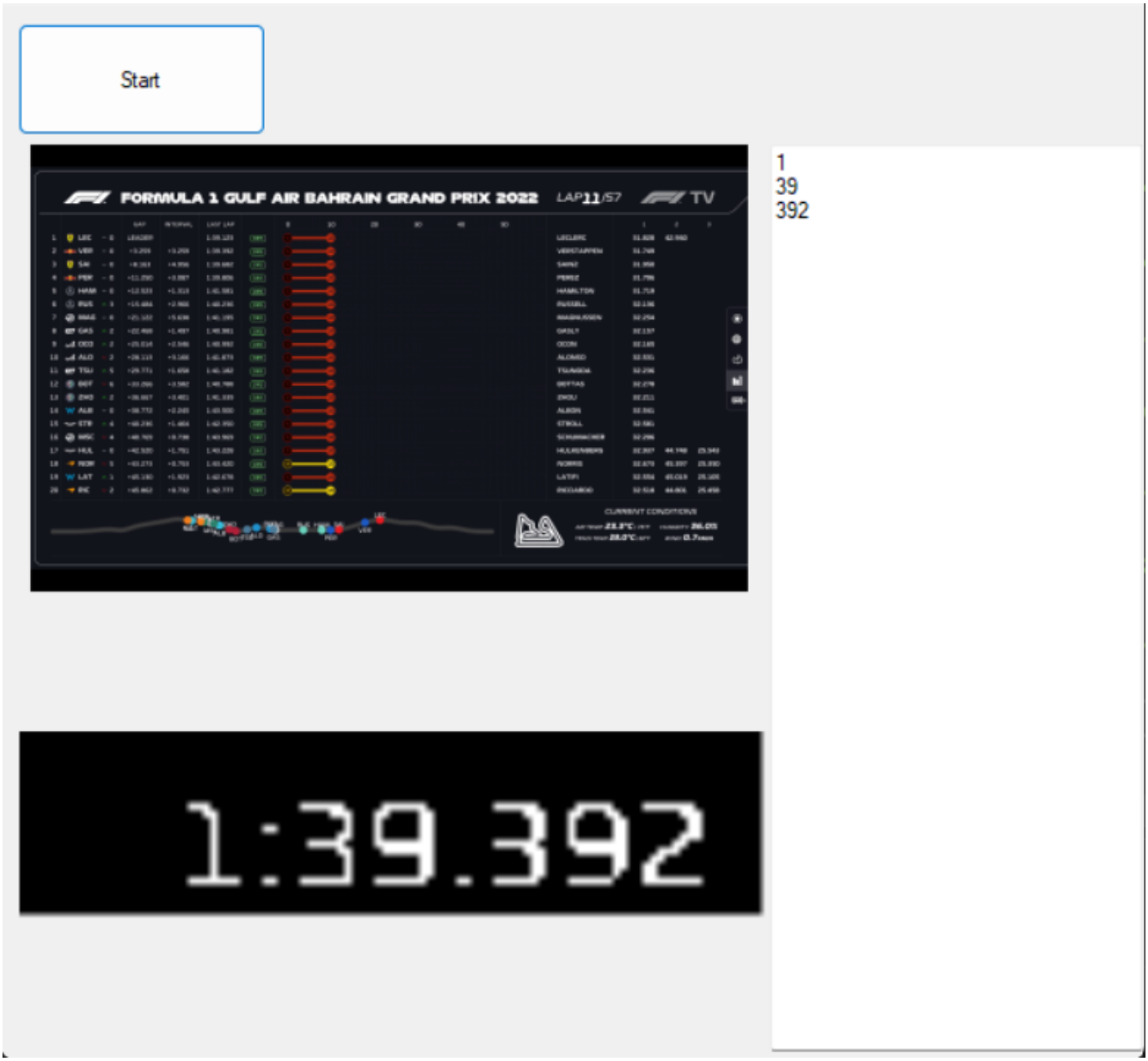
La virgule a tendeance à se barrer ce qui est particulièrement problématique. Cependant comme les chiffres ont beaucoup moins de possibilitées que les lettres et qu'il n'y a pas de problème de langue on devrait pouvoir travailler à faire des règlage que l'on pourra ensuite utiliser.

Avec un Treshold de 165 on arrive presque à quelque chose d'intéressant :



Le + n'est clairement pas compris mais ca n'est pas embêtant car c'est souvent redondant. On arrive cependant à isoler 3 et 259. Même si la virgule n'est pas comprise cela veut dire qu'il est tout de même possible de discriminer les secondes des milisecondes.

Maintenant avec un temps au tour :



On arrive sans rien changer aux paramêtres à isoler minutes secondes et milisecondes.

Il semble que la reconnaissance de chiffre soit bien plus efficace que la reconnaissance de lettres. Il va falloir faire un test à plus grande échelle avec plus d'image pour se rendre compte de la precision.

Demain ce qui serait bien cela serait que je fasse un jeu d'images avec des valeurs connues et que je fasse une batterie de tests pour voir à quel point je peux faire confiance à la reconnaissance des chiffres.

Automatiser un système de test de la sorte me sera très utile dans le futur pour vérifier la non regression de ma reconnaissance de texte quand je tenterai d'y faire des changements.

Je suis toujours curieux cependant de voir comment le programme se débrouille avec les nombres de tours qui se trouvent dans les icones de pneus.

3.4 Lundi 3 Avril

Aujourd'hui on va faire un programme qui permet de créer un dataset qui permette de tester le programme de reconnaissance.

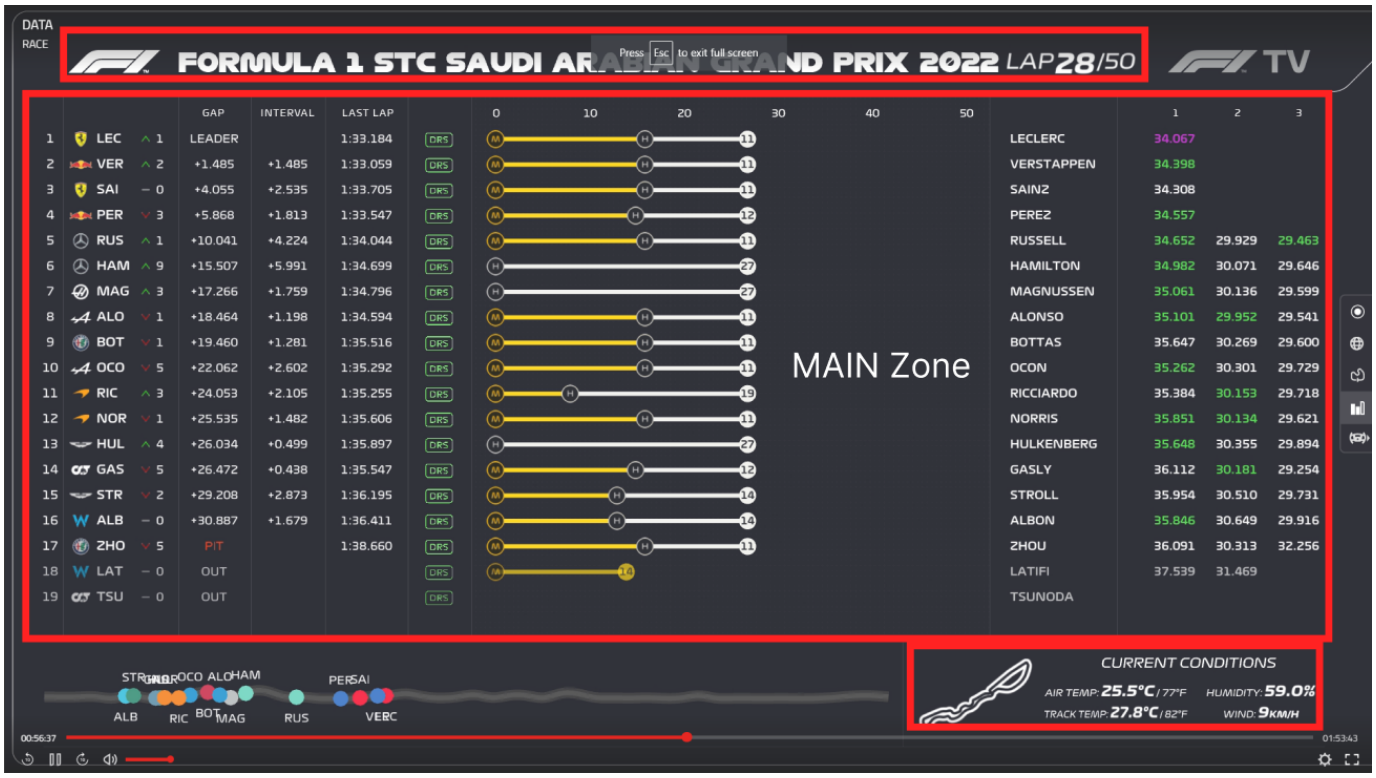
Je pense que le meilleur moyen de faire serait un programme qui crée le dataset et qui ensuite peut tester différentes methodes de reconnaissance.

Par la même occasion je peux développer la technologie qui va permettre de découper une image en 20 lignes ce qui me servira ensuite pour la reconnaissance.

Je me rend compte que pour faire un programme de tests je dois déja avoir une idée de la structure de mon programme.

Pour le moment je réflechis à un système de "Zones" et de "Windows". L'idée serait que une Zone est juste une sous partie d'image qui peut encore être décomposé tandis que chaque Window contient une ou plusieurs informations à récupèrer.

J'ai essayé de découper l'image pour que cela soit plus clair :

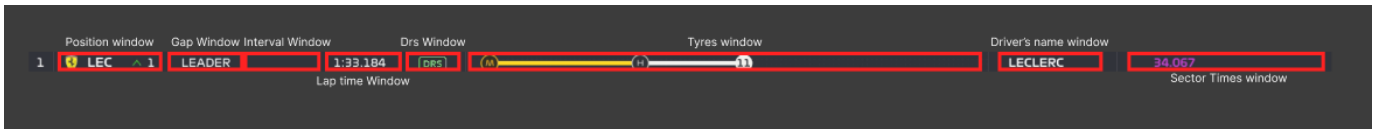


Ici on peut voir que l'image est découpée en plusieurs grandes zones. Dans un premier temps on ne s'occupe que de la première.

Ensuite :

On peut voir la que cette Main zone serait elle même décomposée en plusieurs plus petites zones.

Et ensuite chacunes de ces petites zones :



Sera décomposée en plusieurs windows qui elles sont des zones qui contiennent de l'information.

En gros on aurait trois types de zone :

- Les zones qui contiennent d'autres zones
- Les zones qui contiennent des Windows
- Les Windows

Cependant en y réflechissant on pourrait tout à fait avoir seulement des zones et des windows en faisant en sorte que les windows peuvent avoir une liste de windows et une liste de zones.

Une zone serait composée de :

- Une image de départ
- Un rectangle qui la positionne sur cette dernière
- Une liste de zones (potentiellement vide)
- Une liste de windows (potentiellement vide)
- Une methode qui permet de récupèrer une image de la zone
- Une methode qui permet de lancer la reconnaissance sur chaque window

Une window serait composée de :

- Une image de départ (cela peut être l'image cropée de la zone parente peu importe)
- Un rectangle qui la positionne sur cette dernière
- Une methode qui permet de récupérer un image de la window
- Une methode qui permet de lancer la reconnaisance sur l'image (Chaque type de zone doit l'implémenter)

Dans chaque window on peut imaginer que la methode qui fait la reconnaissance au lieu de retourner un objet qui peut contenir nimporte quel type d'information peut envoyer ce qu'elle vient de récupèrer dans une base de donnée ou un objet.

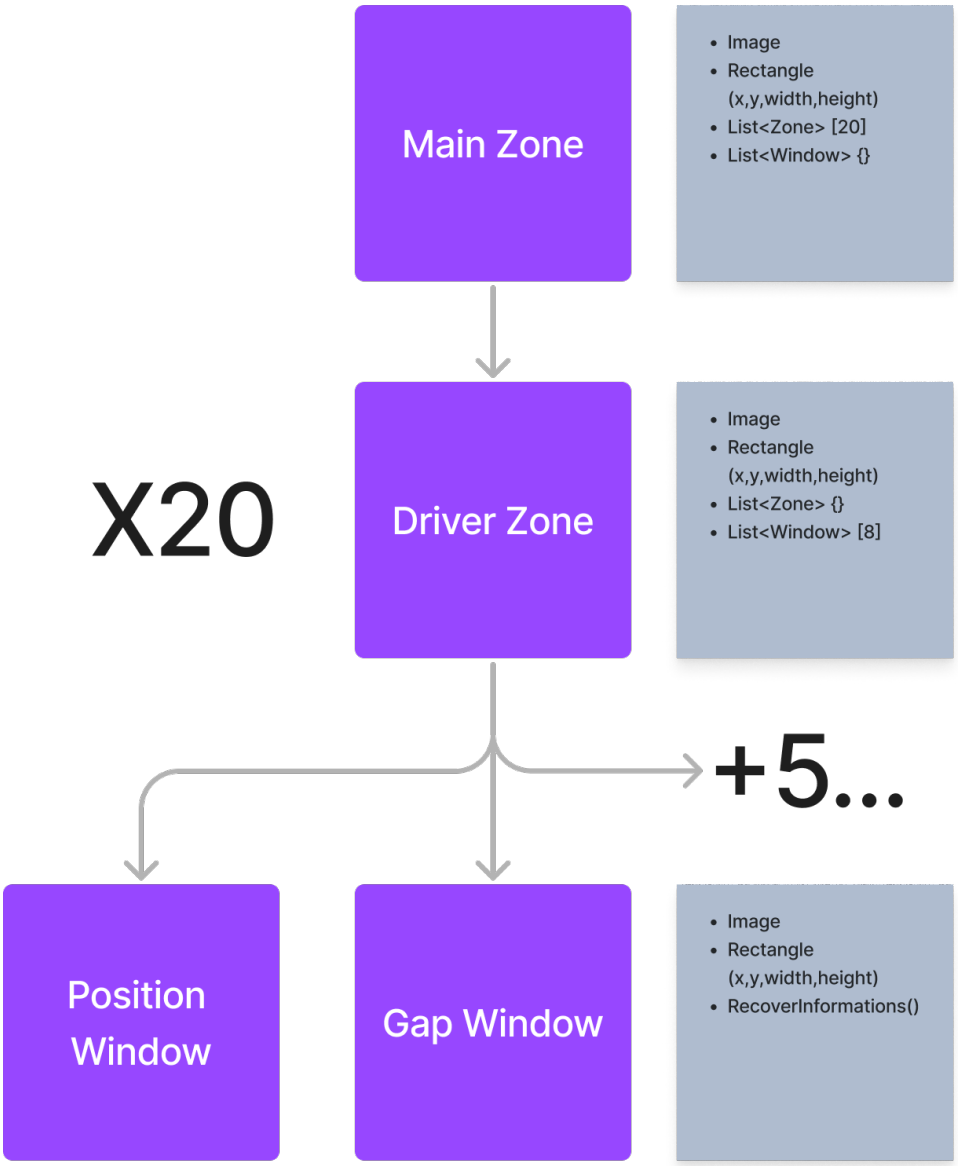
Par exemple une Zone de pilote pourrait très bien contenir un objet pilote et le donner à ses windows qui rempliraient ce même objet.

C'est une reflexion plus stockage que OCR mais c'est intéressant pour savoir ce que fait une window des données qu'elle récupère.

Dans un premier temps je pense que les windows vont simplement écrire dans un fichier ce qu'elles trouvent chacunes dans le format qu'elles veulent.

Pour comprendre pourquoi je me prend la tête il faut savoir que chaque window peut avoir accès à pleins d'informations différentes. On pourrait dire qu'elles retournent toutes une string sauf que si ca marche pour un temps au tour ou pour un nom de pilote, cela ne marche pas forcément pour un type de pneu ou un DRS ouver. Comme chaque window a plusieurs types de data elle devra elle même se charger de comment la traiter ET de la stocker.

Voila un diagramme qui résume comment je vois l'implémentation dans un premier temps :



Voici comment se présente le squellette d'une Zone :

```
public class Zone
    {
        private Bitmap FullImage;
        private List<Zone> Zones;
        private List<Window> Windows;

        private Rectangle _bounds;
        public Rectangle Bounds { get => _bounds; private set => _bounds = value; }
```

```
        public Bitmap ZoneImage
        {
            get
            {
                Bitmap sample = new Bitmap(Bounds.Width, Bounds.Height);
                Graphics g = Graphics.FromImage(sample);
                g.DrawImage(FullImage, new Rectangle(0, 0, sample.Width, sample.Height), Bounds, GraphicsUnit.Pixel);
                return sample;
            }
        }

        public Zone(Image fullImage, Rectangle bounds)
        {
            FullImage = (Bitmap)fullImage;
            Init(bounds);
        }
        public Zone(Bitmap fullImage, Rectangle bounds)
        {
            FullImage = fullImage;
            Init(bounds);
        }
        private void Init(Rectangle bounds)
        {
            Bounds = bounds;
            Zones = new List<Zone>();
            Windows = new List<Window>();
        }
        public void AddZone(Rectangle bounds)
        {
            if(Fits(bounds))
                Zones.Add(new Zone(ZoneImage,bounds));
        }
        public void AddWindow(Rectangle bounds)
        {
            if (Fits(bounds))
                Windows.Add(new Window(ZoneImage,bounds));
        }
        private bool Fits(Rectangle inputRectangle)
        {
            if (inputRectangle.X + inputRectangle.Width > Bounds.Width || inputRectangle.Y + inputRectangle.Height > Bounds.Height || inputRectangle.X < 0 ||
inputRectangle.Y < 0)
            {
                return false;
            }
            else
            {
                return true;
            }
        }
    }
```

Le but est ensuite de créer différent types de Zones.

Par exemple la MainZone devra découper son contenu en 20 parties égales pour tenter de chopper les 20 pilotes. Il serait cool de trouver un moyen de calibrer automatiquement.

C'est peut-être possible de calibrer avec de la reconnaissance de texte, on peut essayer de lancer la reconnaissance et voir ou on trouve du texte avec un peu de chance cela pourrait donner les positions et avec ca on peut peut-être determiner des lignes.

Et voici le squelette d'une window générique

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Drawing;

namespace OCR_tester
{
    public class Window
    {
        private Bitmap FullImage;

        private Rectangle _bounds;
        public Rectangle Bounds { get => _bounds; private set => _bounds = value; }

        public Bitmap WindowImage
        {
            get
            {
                Bitmap sample = new Bitmap(Bounds.Width, Bounds.Height);
                Graphics g = Graphics.FromImage(sample);
                g.DrawImage(FullImage, new Rectangle(0, 0, sample.Width, sample.Height), Bounds, GraphicsUnit.Pixel);
                return sample;
            }
        }
        public Window(Bitmap fullImage, Rectangle bounds)
        {
```

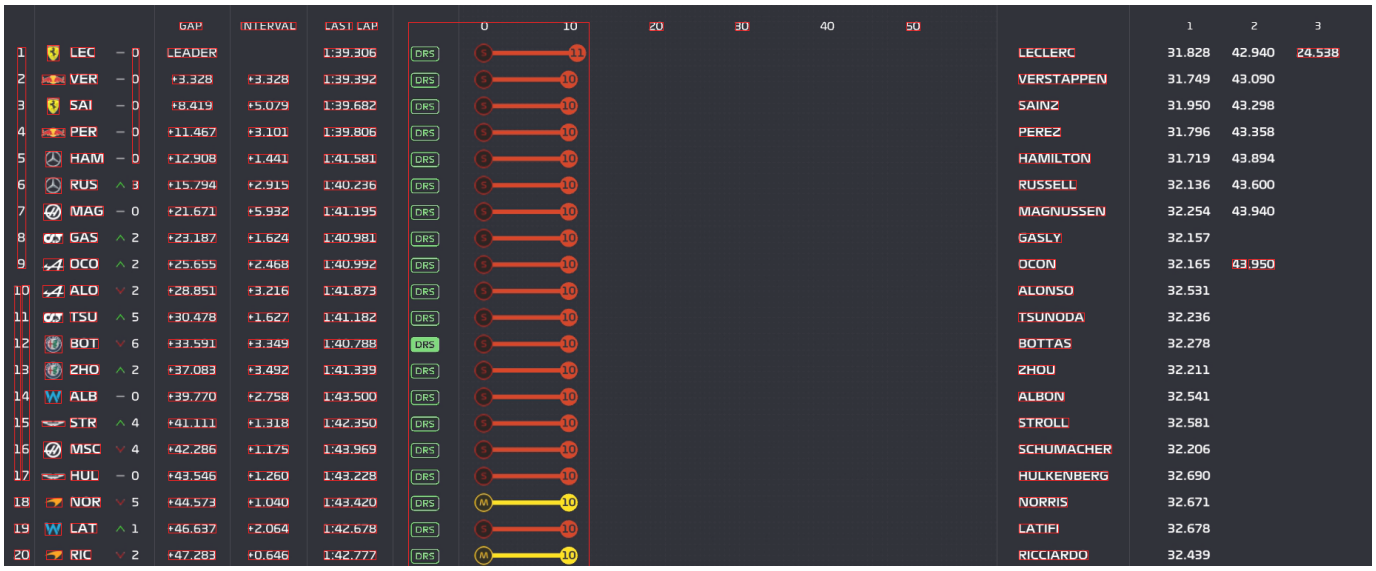
```
            FullImage = fullImage;
            Bounds = bounds;
        }
        public virtual void RecoverInformations()
        {
            //Each Window type will have to implement its own way to recover the informations stored in the Window Image
        }
    }
}
```

Chaque Window pourra ainsi elle même implémenter la récupèration d'informations. La facon de les retourner/stocker est encore un peu floue.

Par exemple pour un temps au tour on peut imaginer que il fait une petite vérification dans l'objet pilote et dans le tableau des tours si il n'y a pas deja une valeur et si il n'y en a pas une alors il peut l'ajouter.

Maintenant je vais essayer de créer une Main window qui se calibre toute seule.

Alors après avoir bien galèré avec l'interface pour permettre au user de cliquer sur la form pour voir les zones qu'il crée, j'ai pu créer un zone qui fait les dimensions de MainZone et j'ai pu lancer la reconnaissance sur l'image et voir ou il trouve du texte :

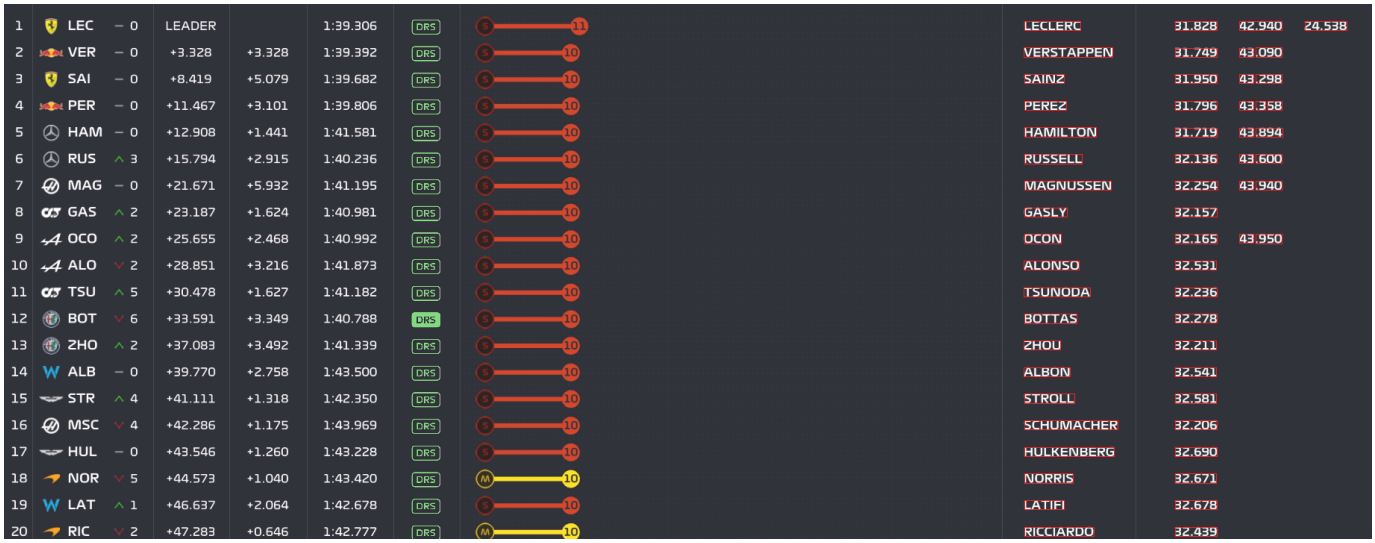


Maintenant il faut que je nettoie la liste de rectangle pour exclure ceux qui sont trop grands pour être sur une seule ligne, ceux qui indiquent le nombre de tour en haut et ceux qui n'ont pas d'intérêts. On pourra ensuite isoler les lignes et créer une liste d'images.

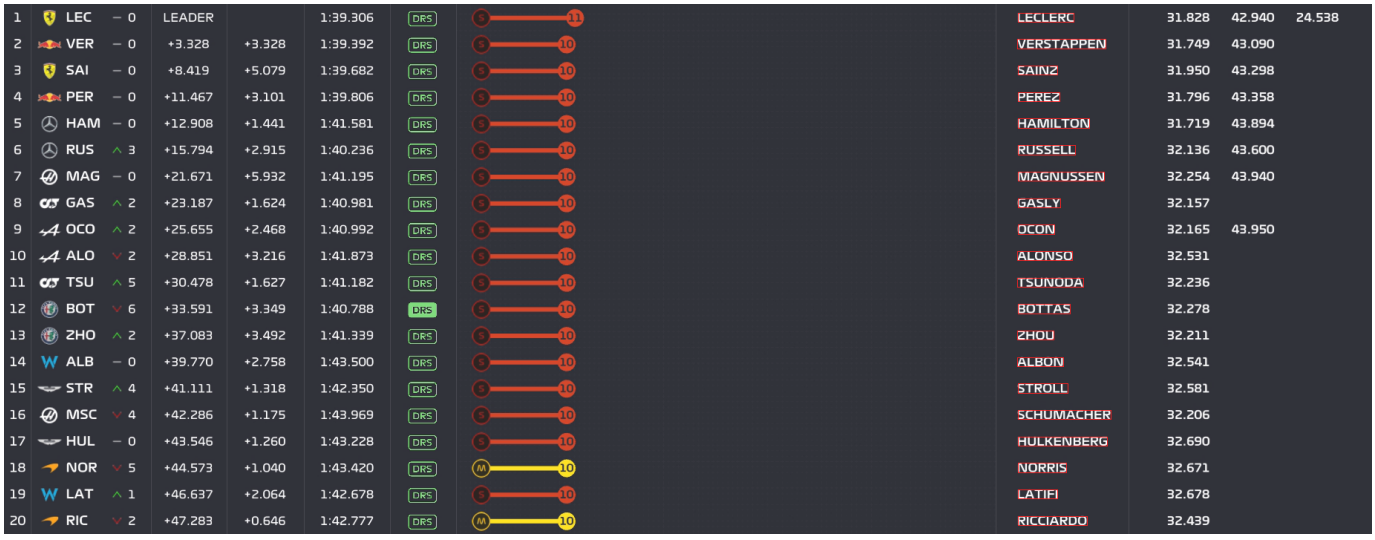
Pour ce qui est de la ligne qui contient les "Gap interval last lap" et des chiffres sur les tours pour les pneus etc je vais juste demander à l'utilisateur de ne pas les prendre dans la screenshot. Comme ils contiennent des mots qui peuvent être utilisés plus loin dans les data je ne peux pas les blacklister et faire un système qui s'occupe de les enlever si ils existent selon le position y me prendrait trop de temps pour rien.

Après avoir filtré un peu les resultats et enlevé les zones beaucoup trop grandes, on se retrouve déja plus qu'avec ca :

Comme on peut le voir, du côté gauche de l'image on a beaucoup de choses reconnues mais avec beaucoup de tailles différentes ce qui n'est pas idéal. Alors j'ajoute un filtre qui permet de ne selectionner que les data sur la droite.



Maintenant il devrait être possible de faire un algorythme qui ne prend que un seul carré par ligne.

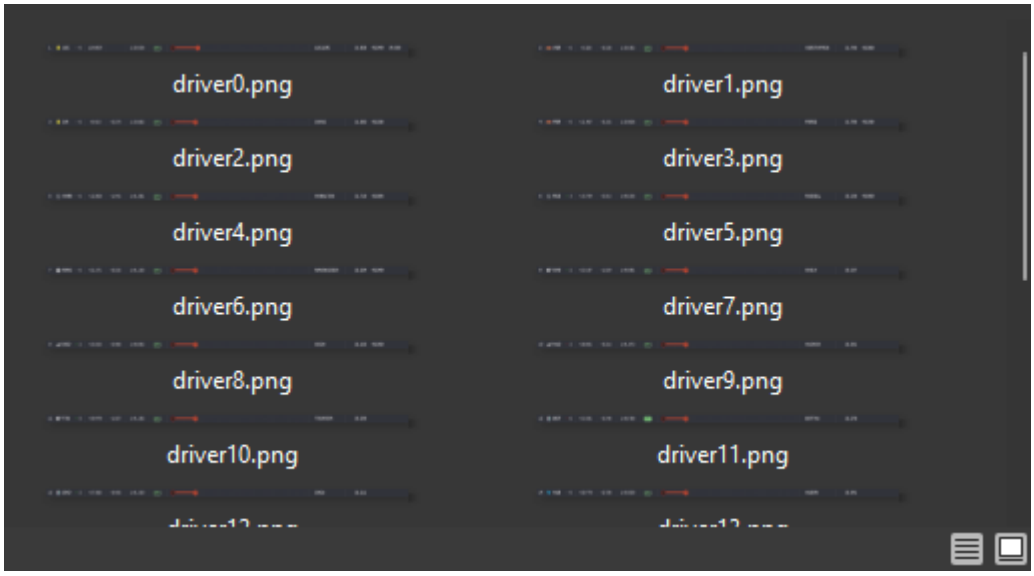


Maintenant que on sait ou se trouve chaque ligne on peut faire un petit traitement et découper l'image en plusieurs windows.

Et voila :

1	LEC	– 0	LEADER		1:39.306	DRS	S 11	LECLERC	31.828	42.940	24.538
2	VER	– 0	+3.328	+3.328	1:39.392	DRS	S 10	VERSTAPPEN	31.749	43.090	
3	SAI	– 0	+8.419	+5.079	1:39.682	DRS	S 10	SAINZ	31.950	43.298	
4	PER	– 0	+11.467	+3.101	1:39.806	DRS	S 10	PEREZ	31.796	43.358	
5	HAM	– 0	+12.908	+1.441	1:41.581	DRS	S 10	HAMILTON	31.719	43.894	
6	RUS	^ 3	+15.794	+2.915	1:40.236	DRS	S 10	RUSSELL	32.136	43.600	
7	MAG	– 0	+21.671	+5.932	1:41.195	DRS	S 10	MAGNUSSEN	32.254	43.940	
8	GAS	^ 2	+23.187	+1.624	1:40.981	DRS	S 10	GASLY	32.157		
9	OCO	^ 2	+25.655	+2.468	1:40.992	DRS	S 10	OCON	32.165	43.950	
10	ALO	v 2	+28.851	+3.216	1:41.873	DRS	S 10	ALONSO	32.531		
11	TSU	^ 5	+30.478	+1.627	1:41.182	DRS	S 10	TSUNODA	32.236		
12	BOT	v 6	+33.591	+3.349	1:40.788	DRS	S 10	BOTTAS	32.278		
13	ZHO	^ 2	+37.083	+3.492	1:41.339	DRS	S 10	ZHOU	32.211		
14	ALB	– 0	+39.770	+2.758	1:43.500	DRS	S 10	ALBON	32.541		
15	STR	^ 4	+41.111	+1.318	1:42.350	DRS	S 10	STROLL	32.581		
16	MSC	v 4	+42.286	+1.175	1:43.969	DRS	S 10	SCHUMACHER	32.206		
17	HUL	– 0	+43.546	+1.260	1:43.228	DRS	S 10	HULKENBERG	32.690		
18	NOR	v 5	+44.573	+1.040	1:43.420	DRS	M 10	NORRIS	32.671		
19	LAT	^ 1	+46.637	+2.064	1:42.678	DRS	S 10	LATIFI	32.678		
20	RIC	v 2	+47.283	+0.646	1:42.777	DRS	M 10	RICCIARDO	32.439		

Maintenant le programme peut créer des zones pour chaque pilote



3	SAI	– 0	+8.419	+5.079	1:39.682	DRS	S 10	SAINZ	31.950	43.298

Maintenant il faut que j'implémente un système un peu similaire pour créer des windows.

Voici la methode que j'ai créé pour l'autocalibration :

```
public void AutoCalibrate()
        {
            List<Rectangle> detectedText = new List<Rectangle>();
            Zones = new List<Zone>();

            TesseractEngine engine = new TesseractEngine(Window.tessDataFolder.FullName, "eng", EngineMode.Default);
            Image image = ZoneImage;
            var tessImage = Pix.LoadFromMemory(Window.ImageToByte(image));

            Page page = engine.Process(tessImage);
            using (var iter = page.GetIterator())
            {
                iter.Begin();
                do
                {
                    Rect boundingBox;
                    if (iter.TryGetBoundingBox(PageIteratorLevel.Word, out boundingBox))
                    {
                        //var text = iter.GetText(PageIteratorLevel.Word).ToUpper();
                        //We remove all the rectangles that are definitely too big
                        if (boundingBox.Height < image.Height / NUMBER_OF_DRIVERS) {
```

```
                        //Now we add a filter to only get the boxes in the right because they are much more reliable in size
                        if (boundingBox.X1 > image.Width / 2)
                        {
                            //Now we check if an other square box has been found roughly in the same y axis
                            bool match = false;
                            //The tolerance is roughly half the size that a window will be
                            int tolerance = (image.Height / NUMBER_OF_DRIVERS) / 2;

                            foreach (Rectangle rect in detectedText)
                            {
                                if (rect.Y > boundingBox.Y1 - tolerance && rect.Y < boundingBox.Y1 + tolerance)
                                {
                                    //There already is a rectangle in this line
                                    match = true;
                                }
                            }
                            //if nothing matched we can add it
                            if(!match)
                                detectedText.Add(new Rectangle(boundingBox.X1, boundingBox.Y1, boundingBox.Width, boundingBox.Height));
                        }
                    }
                }
            } while (iter.Next(PageIteratorLevel.Word));
        }
        foreach (Rectangle Rectangle in detectedText)
        {
            Rectangle windowRectangle;
            Size windowSize = new Size(image.Width, image.Height / NUMBER_OF_DRIVERS);
            Point windowLocation = new Point(0, (Rectangle.Y + Rectangle.Height / 2) - windowSize.Height / 2);
            windowRectangle = new Rectangle(windowLocation, windowSize);

            Zones.Add(new Zone(ZoneImage, windowRectangle));
        }
    }
```

Ca peut paraitre pas énorme comme code mais pour tout mettre en place ca demande quand même pas mal de reflexion.

J'ai du clean un peu le code que j'avais fait pour permettre la selection de zones et ajouter la possibilité d'ajouter des windows sur une zone.

J'ai juste quelques difficultées à les ajouter correctement, j'ai un offset tout pourri qui se met tout le temps

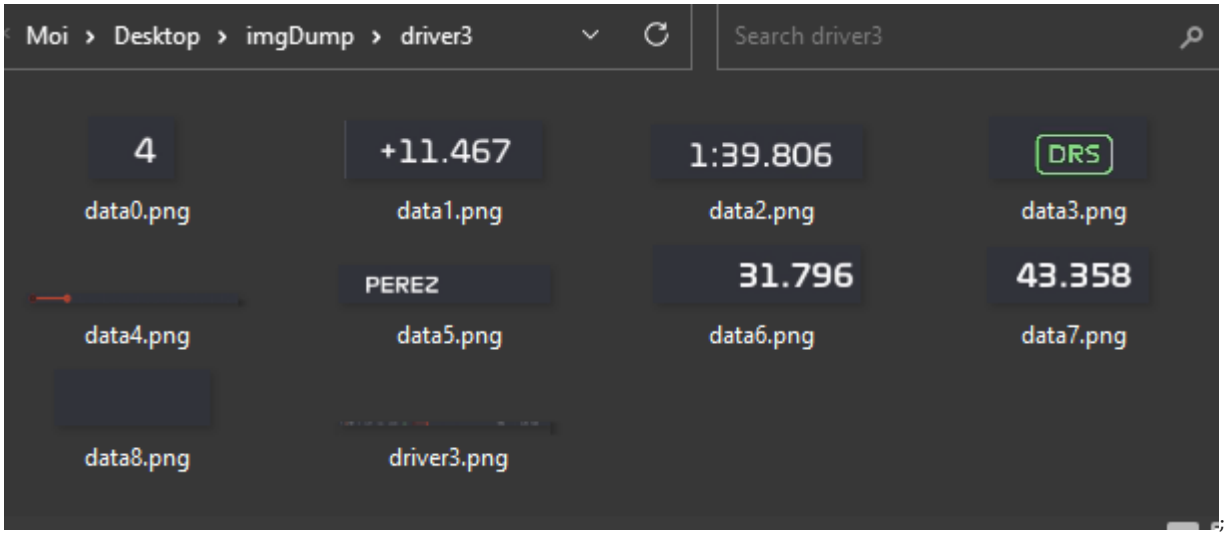
Cela doit être un soucis lors de la detection de clic qui met un offset en trop. C'est vraiment pénible en tout cas.

Certes c'est moins fun de devoir manuellement indiquer ou sont les windows sur une ligne de pilote, mais je ne vois vraiment pas comment faire cela automatiquement. Le but c'est de faire une configuration qui puisse être sauvegardée comme ca pas besoin d'à chaques fois le refaire.

C'est bon ! J'avais juste oublié de changer le calcul d'offset entre le code de la zone et de la window. Note pour plus tard, il serait peut-être judicieux de faire quelque chose pour la vue, les windows et les Zones ont le même exact comportement pour la vue ce qui fait dupliquer du code.

Mais au moins maintenant ca fonctionne :

Et le programme va directement créer un dossier par pilote avec toutes les images de chaque Data le concernant :



Et c'est tout pour aujourd'hui je pense. Ce qui serait cool demain c'est que je puisse stocker d'une manière ou d'une autre ces fichiers de calibration et que je puisse les transfèrer vers le programme qui va s'occuper de décoder et commencer gentillement à décoder les différents types de data.

Note pour quand je ferai les tests. Je pense que la meilleure idée serait que je prenne pleins de photos du style et que je les mette dans un fichier CSV ou JSON avec leur contenu. Et ensuite je le fais passer en tests pour calculer la prescision de mon algo de décodage.

Pour le moment on est plutôt dans les clouts niveau planning.

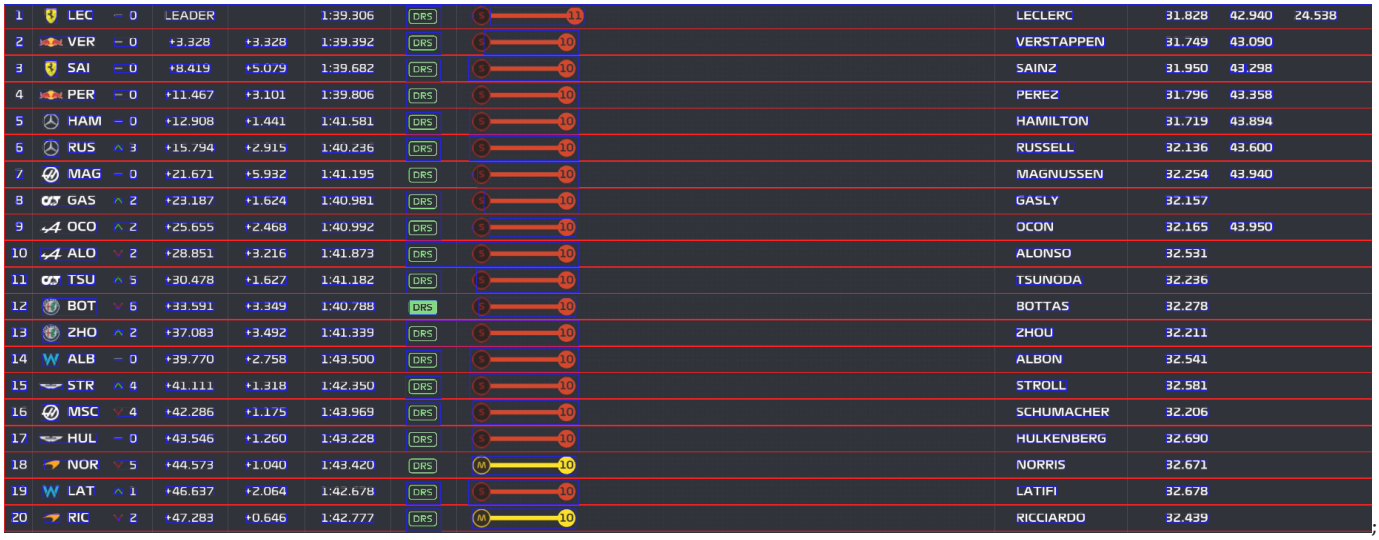
3.5 Mardi 4 Avril

Aujourd'hui je suis scensé plutôt bosser sur l'interpretation des données, mais une idée m'a taraudé l'esprit toute la nuit. Est-ce que je ne pourrais pas quand même essayer de décomposer la zone de pilote directment comme pour la Main zone.

Pour ce faire j'ai tenté de faire comme pour la main zone c'est à dire lancer la reconnaissance pour savoir ou étaient tous les champs de données mais malheureusement je ne pense pas que cela va être possible.

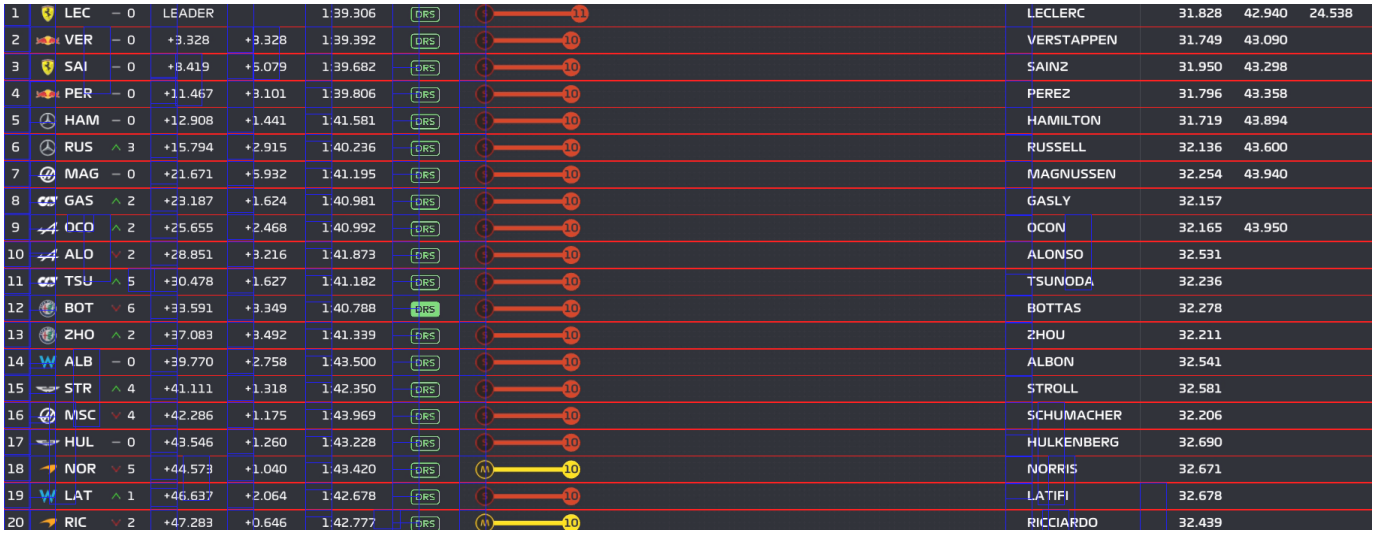
En effet non seulement ici les champs sont de tailles très variées, mais en plus la reconnaissance n'arrive pas à en récupèrer le même nombre sur chaque ligne ce qui risque d'être complexe à utiliser ensuite.

La preuve :



Cependant tout n'est pas perdu ! Il y a peut-être un moyen qui serait mieux en tous points. Le soucis avec ce type de reconnaissance c'est qu'on utilise beaucoup de ressources inutiles. On peut peut-être hard coder la valeur des diviseurs et les utiliser pour créer des zones.

Ok alors visiblement c'est un problème car il semble y avoir d'autres pixels de cette couleur dans l'image (Qui l'aurait cru lol)



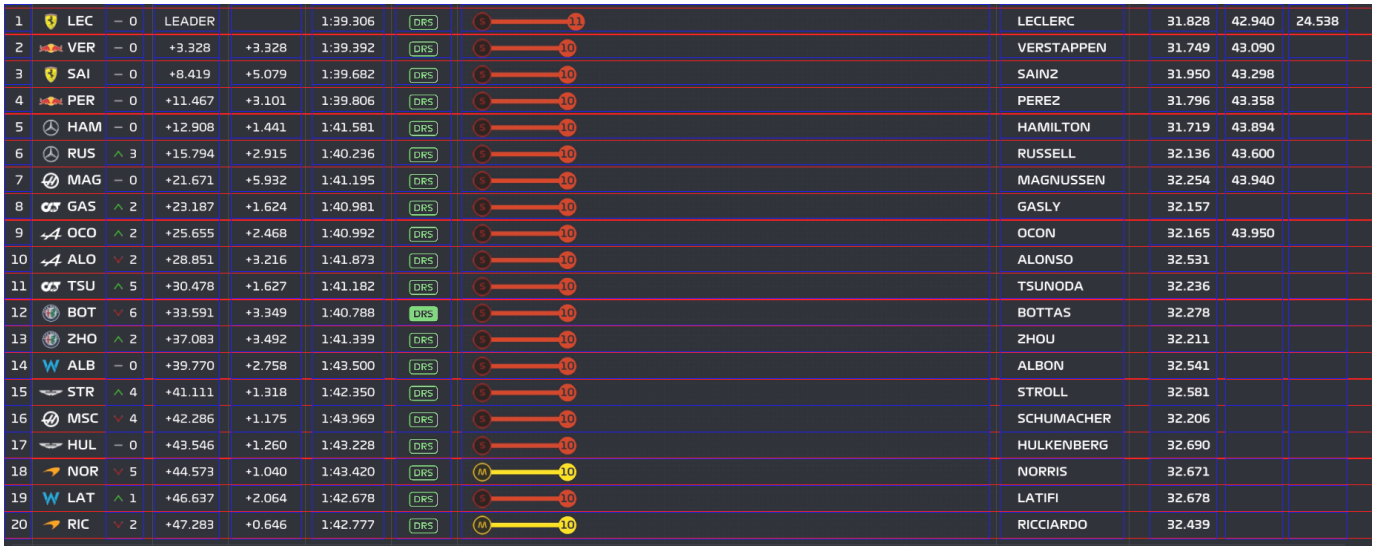
J'a tenté de réduire la tolérance mais le soucis c'est que c'est soit trop soit pas assez

Dernière tentative, j'ai essayé de prendre plusieurs pixels en hauteur pour chaque incrément de X et en faire la moyenne, et même comme ca, impossible de trouver de manière efficace les zones. Je pense que je vais donc revert tous mes changements pour revenir à la version ou on les choisissait manuellement.

Pas mal de temps perdu mais bon c'est comme ca ca arrive

Bon j'ai fait un revert mais j'ai ajouté une feature importante. Les zones font la largeur indiquée par l'utilisateur mais elles font la hauteur max comme ca toutes les window font la même hauteur et ca permet à l'utilisateur de ne pas forcément être ultra précis dans sa selection.

Ce qui nous donne :



1	LEC	– 0	LEADER		1:39.306	DRS	LECLERC	31.828	42.940	24.538
2	VER	– 0	+3.328	+3.328	1:39.392	DRS	VERSTAPPEN	31.749	43.090	
3	SAI	– 0	+8.419	+5.079	1:39.682	DRS	SAINZ	31.950	43.298	
4	PER	– 0	+11.467	+3.101	1:39.806	DRS	PEREZ	31.796	43.358	
5	HAM	– 0	+12.908	+1.441	1:41.581	DRS	HAMILTON	31.719	43.894	
6	RUS	^ 3	+15.794	+2.915	1:40.236	DRS	RUSSELL	32.136	43.600	
7	MAG	– 0	+21.671	+5.932	1:41.195	DRS	MAGNUSSEN	32.254	43.940	
8	GAS	^ 2	+23.187	+1.624	1:40.981	DRS	GASLY	32.157		
9	OCO	^ 2	+25.655	+2.468	1:40.992	DRS	OCON	32.165	43.950	
10	ALO	v 2	+28.851	+3.216	1:41.873	DRS	ALONSO	32.531		
11	TSU	^ 5	+30.478	+1.627	1:41.182	DRS	TSUNODA	32.236		
12	BOT	v 6	+33.591	+3.349	1:40.788	DRS	BOTTAS	32.278		
13	ZHO	^ 2	+37.083	+3.492	1:41.339	DRS	ZHOU	32.211		
14	ALB	– 0	+39.770	+2.758	1:43.500	DRS	ALBON	32.541		
15	STR	^ 4	+41.111	+1.318	1:42.350	DRS	STROLL	32.581		
16	MSC	v 4	+42.286	+1.175	1:43.969	DRS	SCHUMACHER	32.206		
17	HUL	– 0	+43.546	+1.260	1:43.228	DRS	HULKENBERG	32.690		
18	NOR	v 5	+44.573	+1.040	1:43.420	DRS	NORRIS	32.671		
19	LAT	^ 1	+46.637	+2.064	1:42.678	DRS	LATIFI	32.678		
20	RIC	v 2	+47.283	+0.646	1:42.777	DRS	RICCIARDO	32.439		

Maintenant je dirais que les deux prochaines choses à faire seraient de stocker ces zones dans un fichier JSON ou autre pour que la calibration puisse être envoyée directement dans le logiciel de reconnaissance et ensuite de faire une calibration sur des images qui font la taille qu'on aura pendant les Grands Prix. Pour le moment elles sont au format 16:10 qui est le format d'écrant de mon laptop.

Pour le stockage j'imagine un fichier qui donne des indications assez simples qui permettent de reconstruire le total des zones quand il est importé plutot que d'écrire les coordonnées en dur pour chacunes.

Chaque Grande zone va implémenter une methode qui s'occupe de mettre tous ses enfants dans un fichier.

```
{
    "MainZone":{
        "x": 10,
        "y": 20,
        "width": 1450,
        "height": 1340,
        "DriverZone":{
            "x": 0,
            "y": 23,
            "height": 25,
            "Windows":[
                {
                    "DriverPositionWindow":{
                        "x": 0,
                        "y": 0,
                        "width": 35
                    }
                },
                {
                    "DriverPositionChangesWindow":{
                        "x": 0,
                        "y": 0,
                        "width":45
                    }
                }
            ]
        }
    }
}
```

C'est le résultat auquel j'aimerais arriver. Mais pour y arriver il faut encore que je crée les différents types de window.

Cela veut dire que je dois décider quelles informations je vais récupèrer de la page.

Par exemple je vais conserver la position du pilote mais au final les changements de positions sont difficiles à lire et sont redondants. Si je garde un historique des positions des pilotes je peux calculer moi même les changements.

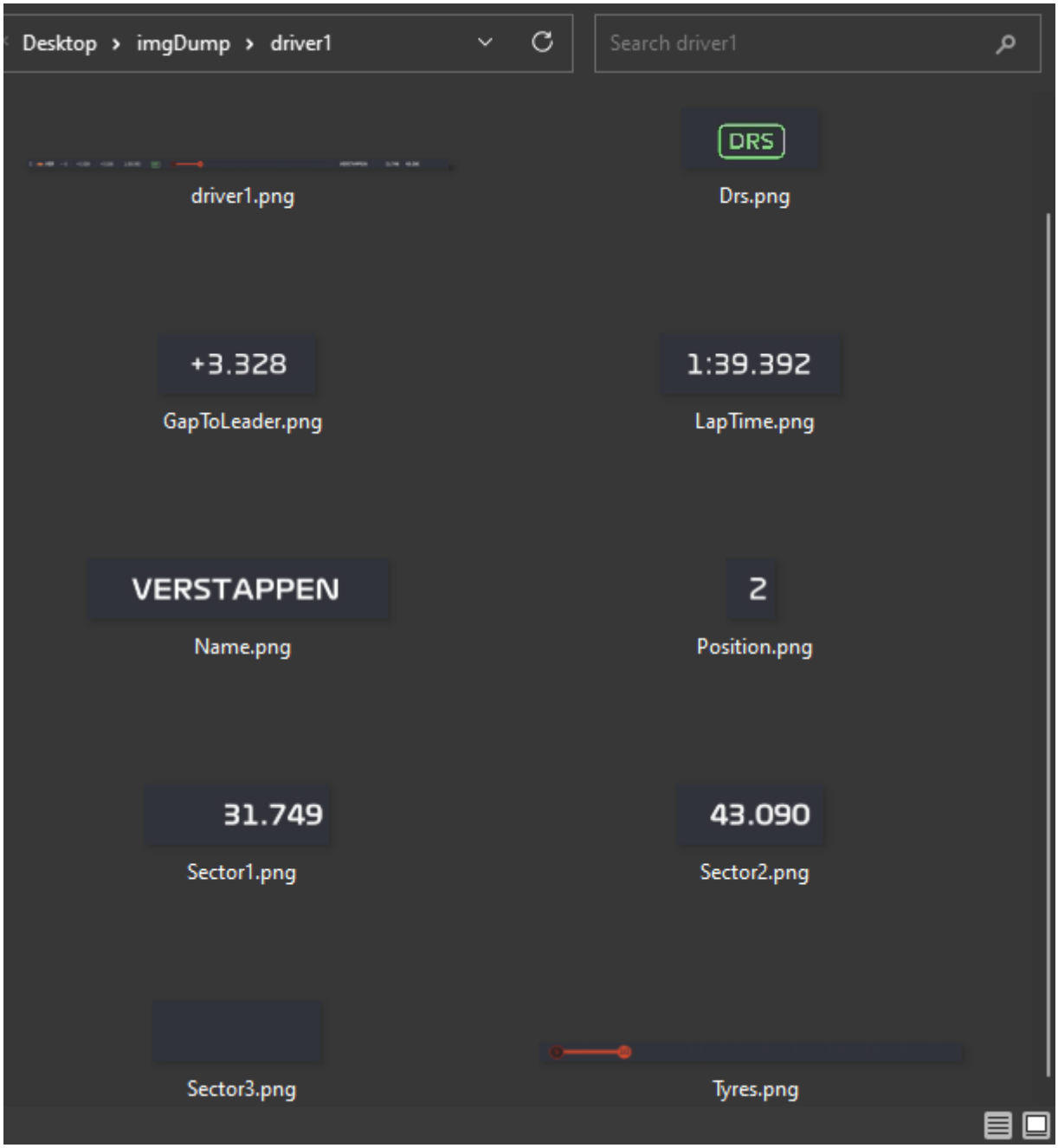
Pareil pour gap avec la voiture devant. Je pense que je vais juste garder l'information des écarts absolus et ensuite je pourrai toujours calculer la différence entre les pilotes.

Ca peut paraître bête car cela rajoute du calcul mais en réalité le calcul de l'OCR est extrêmement gourmand alors il faut que j'évite le plus possible d'y faire recours. Il est bien plus rapide de calculer les écarts que d'essayer de reconnaitre le texte et le convertir en chiffre.

J'ai visiblement ajouté un bug dans mon code. Maintenant tous les pilotes ont la même image quand on les selectionne. Mais visiblement ca n'était pas le cas avant car j'avais pu prendres des images de chaque pilote.

J'ai passé 3 minutes à fixer un bug stupide j'ai un peu envie de brûler ma place de travail... Mais bon au moins maintenant cela fonctionne !

Toutes les images sont récupèrées et ont un format correct avec le bon nom :



Avec un peu de code très moche j'ai pu créer un fichier JSON qui contient les différentes infos. Cependant en exportant TOUT on se retrouve avec un fichier de 1200 lignes ce qui n'est pas optimal.

Mais quand on regarde, il devrait être possible de faire un fichier qui ne contient que les infos d'un seul pilote car ensuite il y a simplement un offset à appliquer sur la zone et les windows.

Je vais donc pouvoir commencer enfin le logiciel de décodage qui prend en entrée un fichier JSON comme celui ci qui a été génèré avec le programme de calibration.

```
{
    "Main": {
        "x": 40,
        "y": 230,
        "width": 1845,
        "height": 719,
        "Zones": [
            {
                "DriverZone": {
                    "x": 0,
                    "y": 3,
                    "width": 1845,
                    "height": 35,
                    "Windows": [
                        {
                            "Position": {
                                "x": 2,
                                "y": 0,
                                "width": 32
                            },
                            "GapToLeader": {
                                "x": 204,
                                "y": 0,
                                "width": 96
                            },
                            "LapTime": {
                                "x": 413,
                                "y": 0,
                                "width": 105
                            },
                            "Drs": {
                                "x": 526,
                                "y": 0,
                                "width": 81
                            },
                            etc...
                        }
                    ]
                }
            }
        ]
    }
}
```

Dans le futur il faudrait ajouter d'autres choses comme par exemple les différents pilotes présents sur le Grand Prix et ce genre d'infos.

Quoique je vais l'ajouter déja maintenant et plus tard je mettrai en place la feature acessible depuis l'interface. Mais le hardcoder maintenant me permet déja de mieux coder l'autre côté.

Ce programme n'est en aucun cas terminé et je vais devoir travailler encore un peu dessus pour qu'il soit utilisable correctement mais au moins il fonctionne à peu près.

Exemple du json avec les noms de pilotes:

```
{
    "Main": {
        "x": 37,
        "y": 238,
        "width": 1851,
        "height": 713,
        "Zones": [
            {
                "DriverZone": {
                    "x": 0,
                    "y": -5,
                    "width": 1851,
                    "height": 35
                }
            }
        ]
    },
    "Drivers": [
        "Leclerc",
        "Verstappen",
        etc...
    ]
}
```

Maintenant je vais m'attaquer au décodage.

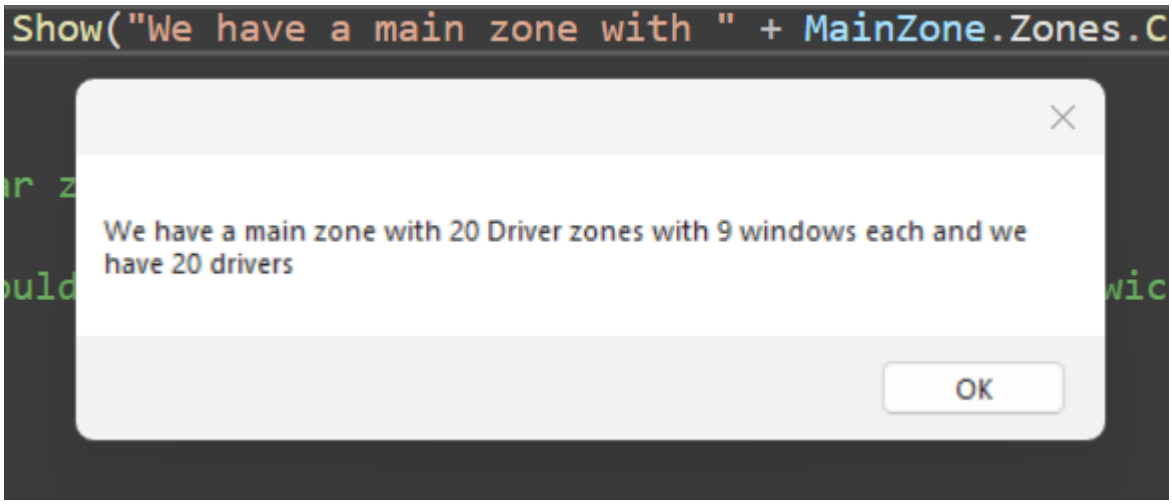
Demain je dois finir le décodage du JSON et je dois commencer à implémenter la reconnaissance des textes. Voire même des pneus etc si j'y arrive.

3.6 Mercredi 5 Avril

Bon la il faut vraiment que je finisse assez vite la lecture du JSON et la reconstruction des zones pour commencer la reconnaissance.

J'ai pris beaucoup de temps à faire le programme de calibration mais je pense que c'est essentiel de prendre ce temps maintenant. (BTW il faudra quand même retourner faire une plus jolie version par ce que la ca marche mais c'est tout)

Bon après pas mal de boulot je pense avoir réussi. Dans le nouveau programme on arrive à récupèrer les différentes zones :

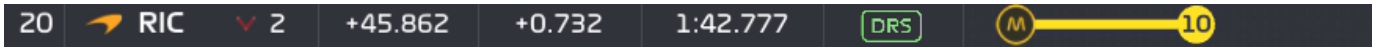


Un conseil de notre professeur M.Bonvin a été de créer des Releases de versions qui ne fonctionnent pas ou pas très bien. J'ai donc publié une première release de l'OCR_TEST qui fonctionne vite fait.

J'ai seulement un petit soucis, comme je recrée complêtement la structure des driver zones avec seulement la première, il y a un petit décalage car entre les zones il y avait un gap.

Ce qui fait que si la première zone est parfaitement centrée :

La vingtième ne l'est plus exactement :

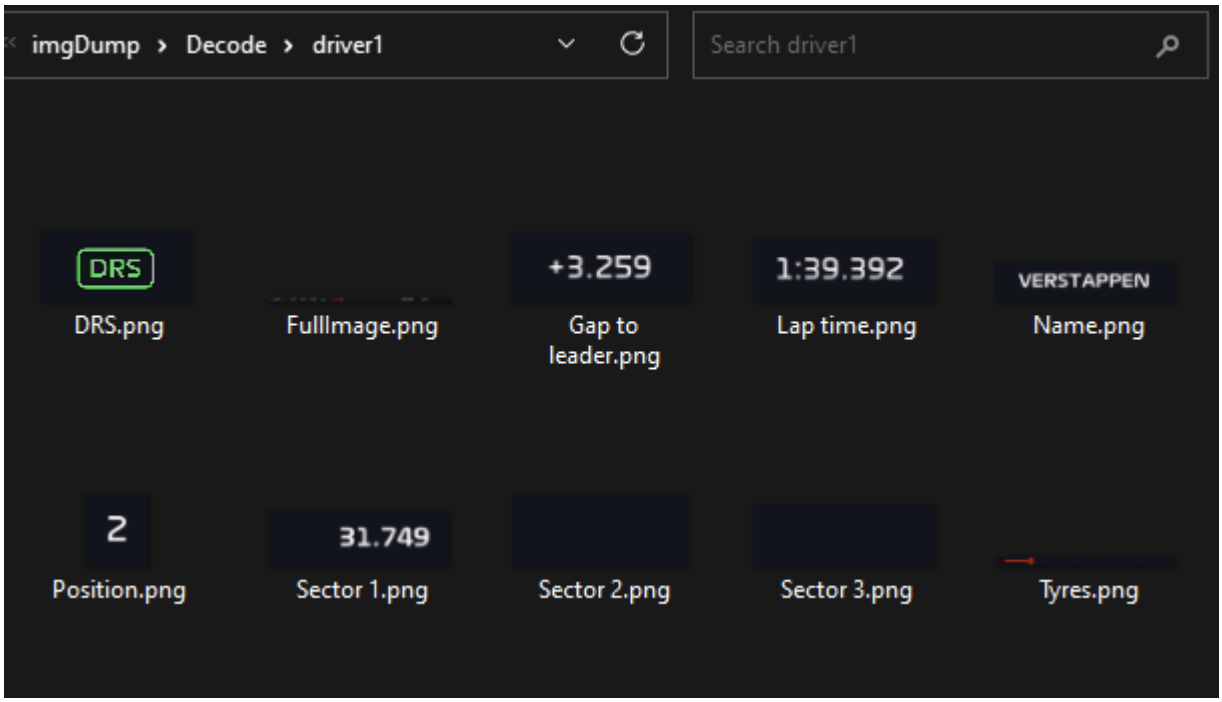


Pour ca j'ai essayé de mettre un espacement arbitraire mais c'est complexe. Je vais plutôt tenter de faire une différence entre la taille de la zone complête et de la taille additionnée de toutes les fenêtre et diviser le resultat entre toutes les fenêtres.

Ca n'est pas parfait mais au moins maintenant les données ne touchent plus les bords de la fenêtre.

Et voila !

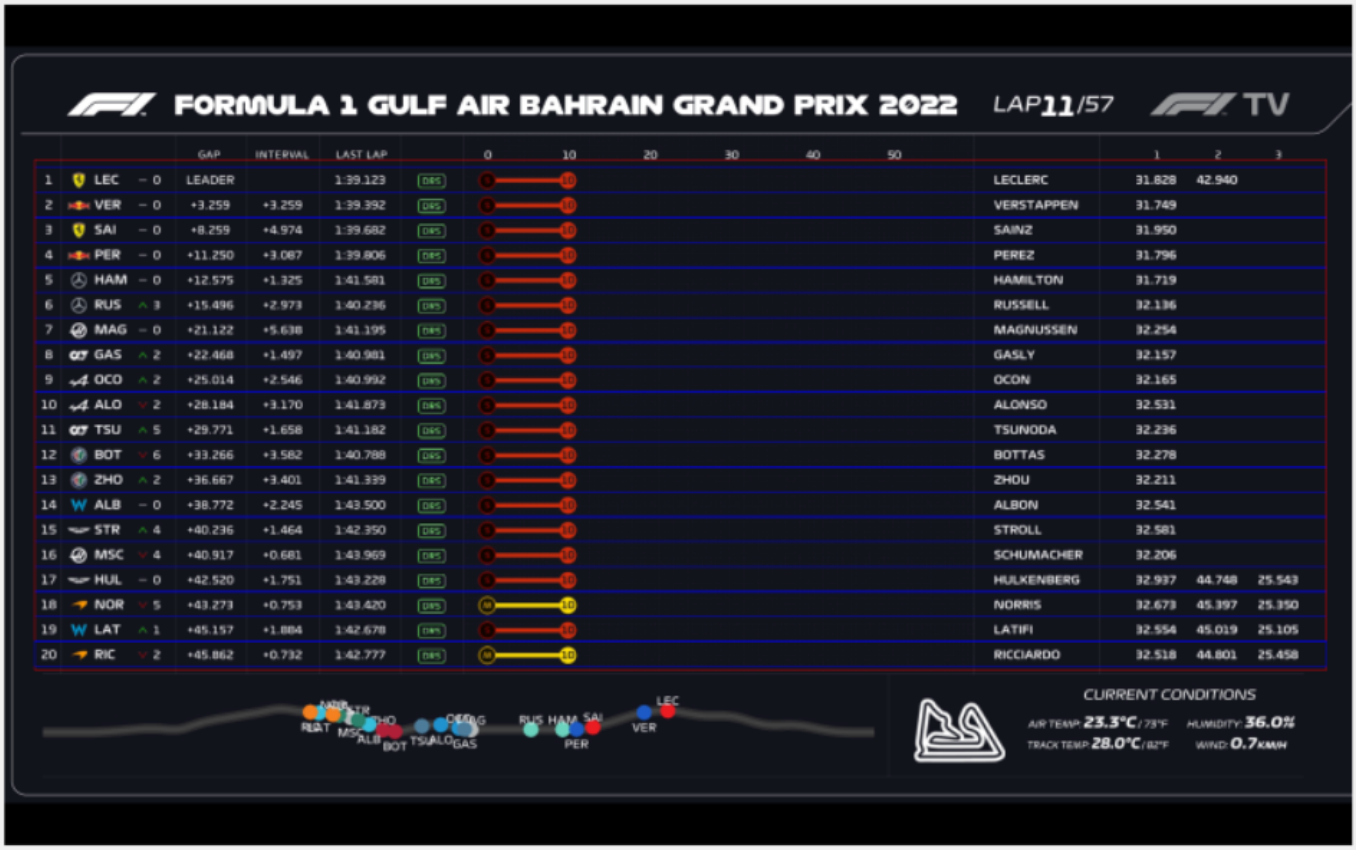
Maintenant avec le fichier de configuration en Json on arrive à récupèrer toutes les infos comme si elles avaient été envoyées directement depuis l'app de calibration mais sans le processing time !



On peut donc ENFIN passer au décodage de ces FICHUES données.

Je vais pouvoir implémenter ce que j'ai fait dans le projet de test de décodage.

Grâce à mon découpage initial qui m'a pris du temps à implémenter on a enfin un truc qui marche même si je n'ai implémenté que la reconnaissance de noms.



Si on se rappelle du système de window et de zones dans le diagramme plus haut, c'est assez facile de comprendre comment je m'y suis pris.

En gros on des listes et des listes de listes de zones, c'est la partie un peu plus technique car il y a des zones qui peuvent contenir d'autres zones etc.

Je vais commencer par la reconnaissance de noms.

Voici le tableau de pilotes de 2023

```
"Drivers": [
        "Leclerc",
        "Verstappen",
        "Hamilton",
        "Alonso",
        "Russel",
        "Gasly",
        "Stroll",
        "Sainz",
        "Hulkenberg",
        "Norris",
        "Tsunoda",
        "Piastri",
        "Zhou",
        "Ocon",
        "Magnussen",
        "Perez",
        "Sargeant",
```

```
        "De Vries",
        "Bottas",
        "Albon"
    ]
```

ET voici le tableau de pilotes de 2022 :

```
"Drivers": [
        "Leclerc",
        "Verstappen",
        "Sainz",
        "Perez",
        "Hamilton",
        "Russel",
        "Magnussen",
        "Gasly",
        "Ocon",
        "Alonso",
        "Tsunoda",
        "Bottas",
        "Zhou",
        "Albon",
        "Stroll",
        "Schumacher",
        "Hulkenberg",
        "Norris",
        "Latifi",
        "Ricciardo"
    ]
```

Je les notes ici car J'ai souvent besoin de changer selon le dataset que j'utilise.

Dans le futur je ferai sûrement un grand dataset qui prend des pilotes de reserves et des pilotes juniors pour que dans le cas ou un pilote est remplacé dans l'année il n'y a pas besoin de tout recalibrer avec l'application.

Après une discussion avec M.Bonvin j'ai décidé de tester 3 valeurs de convertion en noir et blanc et si je ne trouve pas un match exact je prend le nom le plus proche.

Pour trouver la string la plus proche je pense que je vais utiliser quelque chose qui s'appelle la technique de Levenshtein. De ce que j'ai compris c'est un algorythme qui permet de donner une metric de différence entre deux strings.

Bon et évidemment il ne faut pas se tromper dans la liste des pilotes GENRE NE PAS OUBLIER QUE GEORGE RUSSELL COMPORTE DEUX WFNEWIEWV DE "L" A LA FIN DE SON NOM CE QUI POURRAIT ENGRANGER 2H DE DEBUGGING POUR RIEN ASK ME HOW I KNOW joker laugh

J'ai vraiment un soucis avec Tsunoda, Il a trop tendeance à le confondre avec "TSUNDDA" et pour des raisons obscures, quand j'applique l'algorythme de Levenshtein le plus proche n'est pas "Tsunoda" mais "Sainz" iniuvbwdiucbiubisc POURQUOI !!??!!

Je pense que cela demande moins de changements de lettres enfin bon c'est quand même pas idéal. Il va falloir que je trouve un moyen de le repondérer. C'est dommage par ce que cela marche super avec Alonso Verstappen et Albon.

J'ai un peu modifié la methode et j'ai fait en sorte d'envoyer tous les noms en majuscules en me disant que cela pourrait réduire le nombre de changements. Et ca a marché !! Cela va sûrement demander plus de tests pour être bien sûr que tout fonctionne nikel, cependant pour le moment ca marche parfaitement avec les pilotes de 2022.

Pour ce qui est de la reconnaissance de chiffres, j'ai déja fait une partie du boulot le premier jour alors je vais juste reprendre à partir de là.

Je récupère une string de ce type "1:35.123" le soucis c'est que les : se transforment parfois en . ou inversement mais bon ca devrait pas être trop dur à gèrer.

Il faut que je transforme cette string en nombre de milisecondes (Du moins je pense que c'est le meilleur moyen pour ensuite pouvoir facilement comparer et stocker l'information).

Cela fait que 1:35:123 en milisecondes donne :

- 1 * 1000 * 60 => 60'000
- 35 * 1000 => 35'000
- 123 => 123

Total : 60'000+35'000+123 => 95'123ms

Et pour l'affichage :

- Minutes = ms / 60'000
- secondes = (ms - (minutes/60'000))/1000
- ms = ms - ((minutes 60'000) + (secondes * 1000))

Et on se retrouve avec 1:35:123

Maintenant après un peu de temps pour nettoyer la string etc on se retrouve avec un résultat comme le suivant :

```
Position : 0
Gap to leader : 0:0:0
Lap time : 2:15:123
DRS : False
Tyre : Undefined laps with the tyre : 0
Driver name : LECLERC
Sector 1 : 0:31:323
Sector 2 : 0:42:340
Sector 3 : 0:0:0
```

Evidemment pareil pour les autres pilotes Et je me rend compte que j'ai encore tout cassé car le laptime ne devrait pas être 2:15 mais 1:35...

Voila après une heure de debugging et des ajouts pour nettoyer les chaines on se retrouve avec :

```
Position : 0
Gap to leader : 0:0:0
Lap time : 1:35:123
DRS : False
Tyre : Undefined laps with the tyre : 0
Driver name : LECLERC
Sector 1 : 0:31:323
Sector 2 : 0:42:340
Sector 3 : 0:0:0
```

Note: le traitement commence à devenir long, il serait peut-être intéressant d'utiliser un seul Tesseract Engine ou de voir ce qui prend autant de temps, on dépasse la seconde de traitement ce qui est un peu ma limite.

Après on peut toujours tester de rajouter du multicore processing mais c'est pour une autre fois.

Demain je m'occupe de règler les soucis que j'ai avec la prescision de ces temps au tour et j'éspère pouvoir m'occuper aussi de la position des pneus et du DRS. J'aimerais finir tout ca cette semaine.

3.7 Jeudi 6 Avril

Une idée m'est passée par la tête pendant que je dormais, dans la liste des pilotes, quand ils sont à plus d'un tour de retard avec le leader (Ce qui arrive normalement dans presque tous les Grand Prix) on a pas des minutes mais une string qui montre "+1 Lap" ou "+2Laps" ce qui est évidemment un problème. Je pense qu'une bonne facon d'envoyer l'info serait de retourner -1 -2etc... à la place des milisecondes, mais encore faut-il detecter le nombre de tours

Je devrais être en train de commencer la documentation de commment tout ce que j'ai fait fonctionne. Cependant je ne me vois pas faire ca tant que je n'ai pas au moins récupèré toutes les infos au moins un peu proprement. Cela veut dire que je commence officiellement à prendre du retard. (Sachant que si je finis tout aujourd'hui une journée de doc suffira largement le terme est un peu exagèré mais bon)

Bon pour la reconnaissance des temps c'est spécial... Le filtre semble ne pas changer grand chose ce qui est problématique et ca n'est vraiment pas fiable.

Voici quelques expemples avec un treshold de 100:

Cette image est comprise comme "11ZSD"

Cette image est comprise comme "42340"

Et celle ci "ZZAEB"...

Ce qui... n'est pas bon du tout...

J'ai essayé de trouver un fichier d'entrainement spécifiquement fait pour les digits. J'ai essayé de blacklister les chars non voulus pour tenter d'obliger Tesseract à trouver des chiffres.

Avec la première option, les résultats ne sont pas meilleurs voire pires. Avec la seconde option c'est déja pas mal mieux mais on perd complètement la possibilité de detecter les mots comme "LEADER" ou "LAP" et de toute facon ca n'est pas parfait.

Le soucis c'est que si je n'ai pas des données fiables c'est juste impossible de faire des calculs et de l'affichage correct...

Il faut absolument que je trouve une solution.

J'ai essayé d'utiliser de l'interpolation our augmenter la taille de l'image et ensuite appliquer mon filtre pour retirer le flou mais sans succès...

Pourtant la on se retrouve avec des images plutôt claires :

Ici le programme trouve "44301"

Et ici "A5151"...

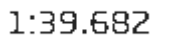
On a toujours les mêmes problèmes.

Bon je suis allé me renseigner sur l'OCR et je me suis dit que j'allais tenter de faire les choses proprement.

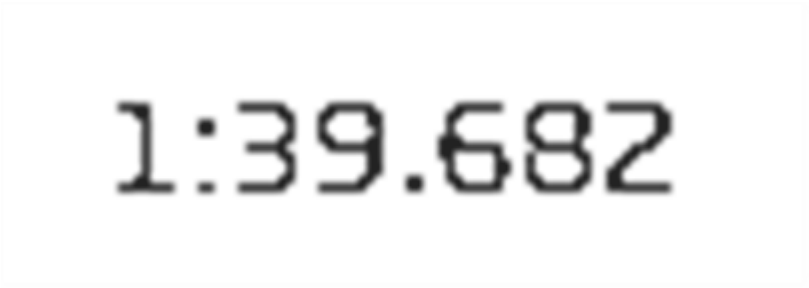
Je vais faire passer plusieurs étapes de postProcessing avant de donner l'image à Tesseract.

- GrayScale
- Tresholding
- InvertColors
- Scaling
- Dilatation

Ce qui donne :



;



;

Ce qui ne change : Roulement de tambour RIEN kjd viuwvuirnvoirenbf

Tout ca pour rien...

C'EST BON !!!

Bon en fait au final le problème était une mauvaise configuration de Tesseract. Je vais devoir un peu nettoyer tout ca. Mais avec les changements de l'image on a des résultats BEAUCOUP plus précis et potentiellement utilisables.

La je vais devoir faire un serieux travail de nettoyage et simplification de mon code par ce que la c'est vraiment un chantier vu le nombre de choses que j'ai du essayer.

J'ai du aussi beaucoup modifier la gestion de l'image ce qui donne :

Et la on a des résultats qui sont vraiment bons.

J'ai pu ajouter assez facilement la detection de position comme c'est simplement un chiffre.

On se retrouve maintenant avec ce genre de retours :

```
Position : 1
Gap to leader : 8:33:51
Lap time : 2:19:123
DRS : False
Tyre : Undefined laps with the tyre : 0
Driver name : LECLERC
Sector 1 : 0:31:828
Sector 2 : 0:42:940
Sector 3 : 0:0:0
Position : 2
Gap to leader : 0:3:259
Lap time : 23:12:392
DRS : False
Tyre : Undefined laps with the tyre : 0
Driver name : VERSTAPPEN
Sector 1 : 0:38:119
Sector 2 : 0:0:0
Sector 3 : 0:0:0
```

Il ne manque plus que l'implémentation de la reconnaissance du DRS et des Pneus

Et non... je viens de me rendre compte que mon programme a encore cassé car le tap time ne peut pas être 23 min lol.

J'ai un nouveau magnifique problème... Les points et les deux points sont interprêtés comme des chiffres ... Give me a F** break...

J'ai du mal à comprendre pourquoi ils ne sont détectés comme tels que maintenant.

Bon alors il semblerait les temps au tour aie besoin d'un ordre très précis pour fonctionner.

- Grayscale
- InvertColors
- Tresholding
- Resize * 2
- Resize * 2

Et la on a des résultats un peu mieux.

Bon demain il faut absolument que je me charge de règler tous ces problèmes et que je commence la reconnaissance des pneus et de DRS par ce que je commence à être en retard.

3.8 Vendredi 6 Avril 2023

Alors aujourd'hui c'est le dernier jour avant de commencer à être en retard pour de bon.

J'ai réussi à règler le problème des temps au tour, des gaps, et des secteurs. Dans le processus j'ai cassé la detection de position mais ca devrait pas être TROP compliqué.

Et voila ... Après seulement plus de dix heures de galère, si on donne cette image au programme et le bon JSON le programme nous retourne :

```
Position : 1
Gap to leader : 0:05:059
Lap time : 1:39:123
DRS : False
Tyre : Undefined laps with the tyre : 0
Driver name : LECLERC
Sector 1 : 0:31:828
Sector 2 : 0:42:940
Sector 3 : 0:00:000
Position : 2
Gap to leader : 0:03:259
Lap time : 1:39:392
DRS : False
Tyre : Undefined laps with the tyre : 0
Driver name : VERSTAPPEN
```

```
Sector 1 : 0:31:749
Sector 2 : 0:00:000
Sector 3 : 0:00:000
```

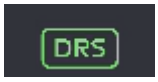
Evidemment le GapToLeader est faux sur leclerc car il est leader mais bon ca je pourrai toujours Hardcoder que le premier a jamais de GapToLeader.

Bon j'ai eu beaucoup de soucis que je ne vais pas mentionner ici car ce sont simplement des soucis de logique de programmation pour trouver un DRS ouvert ou non.

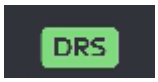
Au final la technique que j'utilise et qui marche plutôt bien pour le DRS est que je prend la première image de DRS et je la déclare comme valeur étalon d'un DRS non actif, en effet dans 99% des cas le leader n'a pas de DRS (cela peut arriver alors il faudra donc juste verifier que les pilotes sont bien à moins de deux secondes les uns des autres pour confirmer).

Ensuite cette valeur étalon je la calcule en fonction du nombre de pixels verts dans l'image et si il y a plus de 30% de pixels verts en plus c'est que le DRS est activé ex:

Ceci est un DRS fermé:



Ceci est un DRS ouvert:



Cela marche à peu près tout le temps mais dans le pire des cas on peut toujours verifier que les pilotes sont bien proches pour detecter les potentiels rares cas de faux positifs.

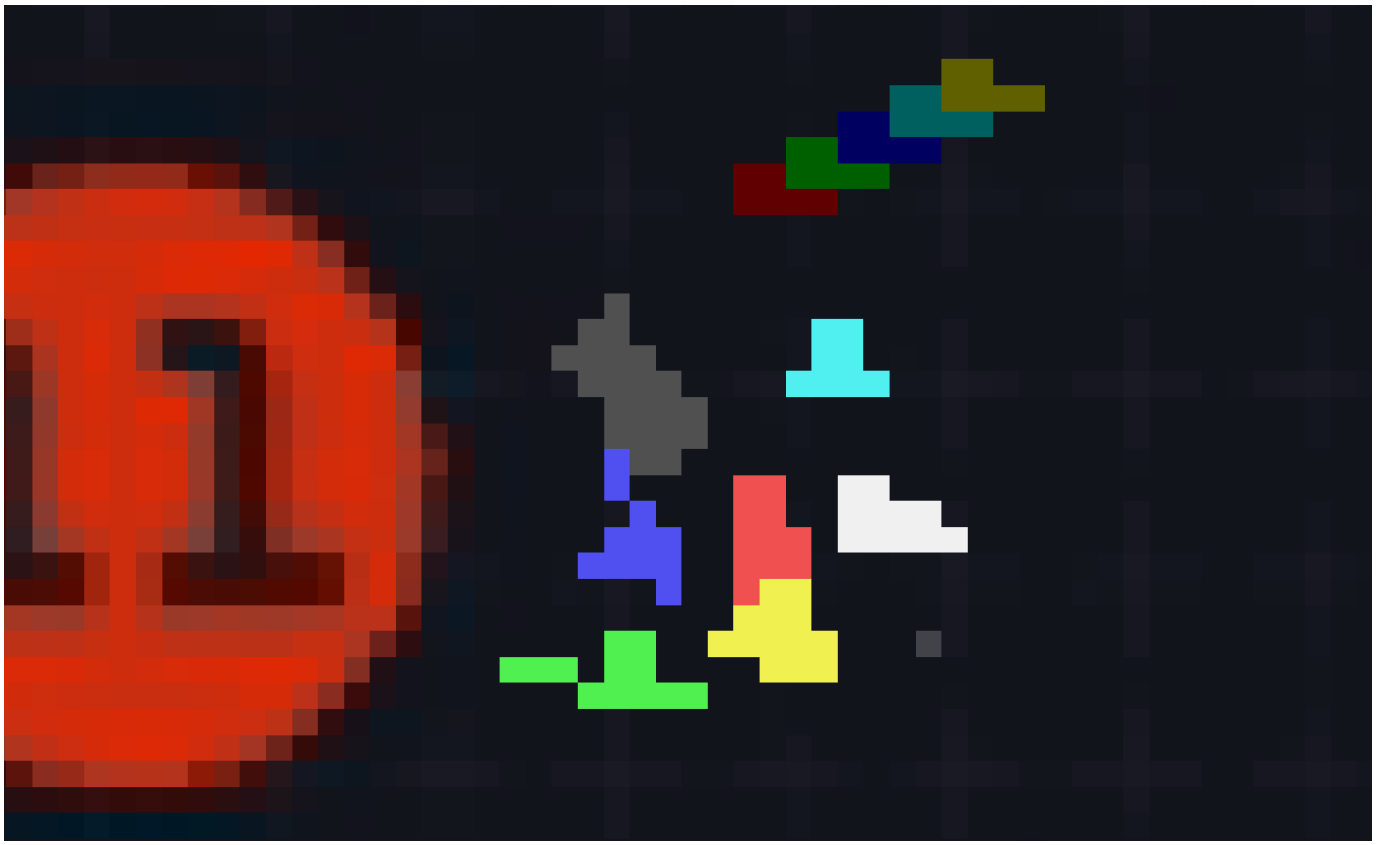
J'ai pu augmenter les performances en utilisant un seul engine pour tout le monde et en arrêtant d'utiliser GetPixel et SetPixel qui sont simplement des horreurs à utiliser. Mais elles ne sont pas encore bonnes

Le soucis avec la detection de pneus cependant, c'est qu'il n'est pas possible d'utiliser la reconnaissance pour savoir ou regarder la couleur car cela ne marcherait pas. Je ne peux pas faire trop de post processing car je dois conserver la couleur Je ne peux pas hardocder un endroit ou aller regarder car cela évolue tout le long du Grand Prix.

Bref c'est la galère. En y réflechissant je me suis dit qu'une bonne idée pourrait être de partir de la droite de la zone du pneu en regardant au milieu de la hauteur. Puis continuer vers la gauche jusqu'à ce que je rencontre une couleur différente. Je pourrai ensuite faire une zone un peu vers la gauche qui devrait contenir les infos du pneu et sur laquelle il sera possible de faire de le reconnaissance de couleur et de la reconnaissance de chiffres.

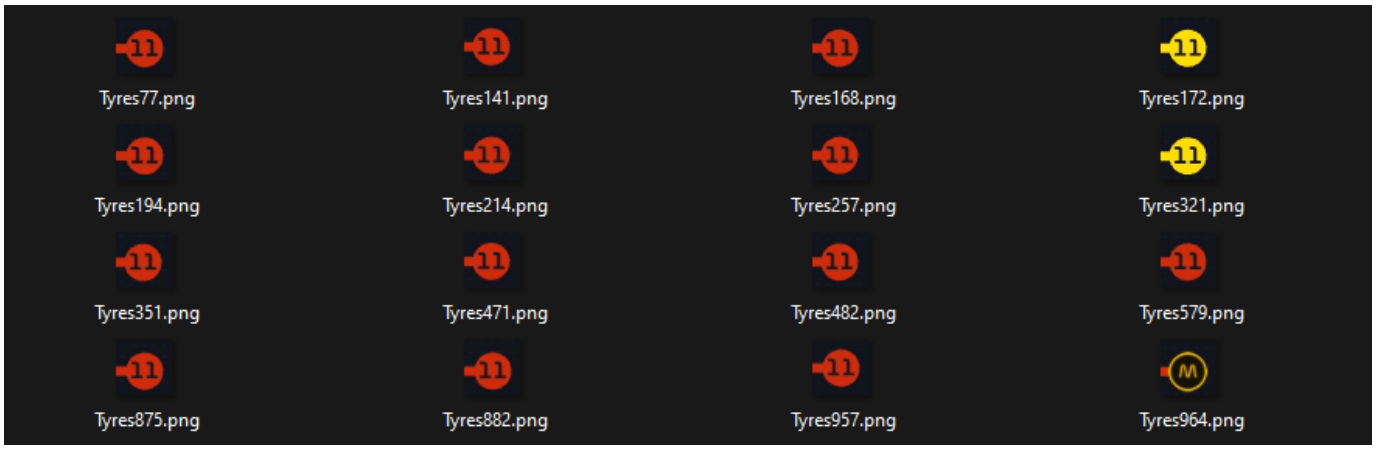
J'ai déterminé que le background n'était jamais plus clair que #505050 et que donc nimporte quelle couleur qui aurait plus que 50 dans un seul des channels serait considèrée comme une couleur cassant le background

Pour arriver à cette conclusion je me suis amusé un peu avec les couleurs pour jouer avec les limites de mon algorythme :



Et je crois que j'ai eu une bonne idée, avec une petite methode bien faite on arrive à de supers résultats :

```
private Rectangle FindTyreZone()
        {
            Bitmap bmp = WindowImage;
            int currentPosition = bmp.Width;
            int height = bmp.Height / 2;
            Color limitColor = Color.FromArgb(0x50,0x50,0x50);
            Color currentColor = Color.FromArgb(0,0,0);

            Size newWindowSize = new Size(bmp.Height,bmp.Height);

            while(currentColor.R <= limitColor.R && currentColor.G <= limitColor.G && currentColor.B <= limitColor.B && currentPosition > 0)
            {
                currentPosition--;
                currentColor = bmp.GetPixel(currentPosition,height);
            }

            //Its here to let the new window include a little bit of the right side
            int offset = Convert.ToInt32((float)newWindowSize.Width / 100f * 20f);
            int CorrectedX = currentPosition - (newWindowSize.Width - offset);
            if (CorrectedX <= 0)
                return new Rectangle(0,0,newWindowSize.Width,newWindowSize.Height);

            return new Rectangle(CorrectedX,0,newWindowSize.Width,newWindowSize.Height);
        }
```



Maintenant cela devrait être beaucoup plus simple de trouver la couleur générale et le nombre de tours.

Donc ce que je fais c'est que je fais une reconnaissance de texte sur l'image réduite.

Si je trouve une lettre c'est facile Ca me donne le type de pneu et ca me dit que c'est le premier tour avec.

Si c'est un nombre alors je fais la moyenne de toutes les couleurs de l'image et je prend la couleur de pneu la plus proche.

Voici les différentes couleurs de pneus :

- SOFT : #FF0000
- MEDIUM : #f5bf00
- HARD : #d9d8d4
- INTER : #00a42e
- WET : #2760a6

Les couleurs de pneus peuvent changer de temps à autres, par exemple cette règle de pneus est arrivée en 2019 et avant il y avait beaucoup plus de couleurs mais dans une volonté de rendre le sport plus facile à comprendre à la télé cela a été simplifié. Je ne pense pas que cela va changer dans les années qui viennent alors tout est hardcodé.

Je pense que j'ai des soucis avec la detection de texte et de couleur car ma zone est trop grande.

Alors bon j'écrit ces lignes apres des heures de tests.

Il semble que la principale difficulté avec ces pneus c'est que les chiffres ou lettres sont minuscules. Il est donc extrêmement difficile de faire une reconnaissance ne serait-ce qu'un peu fiable..

Je fais de mon mieux pour tenter de règler le soucis cependant c'est vraiment complexe.

Je commence à devenir fou, je tente tout et nimporte quoi pour permettre à mon algo de fonctionner et même quand je fais du post processing comme pas possible il me retourne toujours nimporte quoi...

Ici le programme va trouver '5i'...

En fait c'est complexe d'expliquer tout ce que je fais car je change tout en boucle en essayant et en ratant ce qui prend des heures.

Pour aujourd'hui j'abandonne je vais simplement rentrer chez moi et y réflechir cette nuit mais je ne vois pas comment mieux faire la...

C'est terrible par ce que je sens que je ne suis pas bien loin.

3.9 Vacances

Bon je vais un peu laisser de côté la detection de chiffres pour me pencher un peu plus sur la détection de couleur. Par ce que techniquement si j'arrive à toujours parfaitement la detecter alors je pourrais me passer des chiffres car ils sont redondant si je construit un historique de pneus.

J'ai réussi à fix mon problème de mauvaise detection de couleur de pneus. Du moins je crois.

Seulement j'ai quand même un souci, les fenêtres de pneus avec une lettre n'ont pas assez de couleur pour être détectés. Je vais donc essayer de detecter les cinq lettres possibles et si il ne trouve pas alors je pourrai tenter de detecter les chiffres sans lettres ce qui devrait grandement aider.

Le but est encore une fois de réduire les possibilités de Tesseract. Je me rend de plus en plus compte que le plus important c'est de réduire le scope le plus possible. Moins il y a de mots et lettres et de chiffres possibles meilleure sera la reconnaissance.

Bon ca ne veut toujours pas marcher maintenant le 11 est interprêté comme trois I ou comme un M... J'en ai marre sans rire c'est vraiment pénible.

Alors j'écrit ces lignes deux jours plus tard et me rend compte avec horreur que toutes mes modifications sur ce journal de bord n'ont pas été auvegardée... yess..

Bon pour faire simple, j'ai réussi à rendre la detection de couleurs bien plus efficace en réduisant la taille de l'image et en ne prenant pas en compte les couleurs que l'on détecte comme étant partie intégrante du background.

Par exemple quand on a une image comme celle ci :

qui contient un background alors que ci dessous, on l'a enlevé.

La différence est ténue mais elle permet de grandement améliorer la prescision de la reconnaissance de couleurs.

Pour ce qui est du nombre de tours je me suis rendu compte que cela n'était déja pas très utile car avec l'historique on devrait pouvoir le déduire. Mais bon pour la forme je me suis dit que cela serait quand même une bonne idée de vérifier avec la reconnaissance. J'étais quasi certain que le soucis était le fait que l'on voie le contour du logo de pneu qui faisait que la reconnaissance avait du mal. Et j'avais raison ! En les enlevant (Ce qui n'a pas été simple) J'ai pu avoir des chiffres beaucoup plus proches de la réalité.

En même temps je ne vois pas bien comment j'aurais pu faire mieux :

Je suis quand même assez fier de voir que j'ai réussi à part de l'image que on peut voir un peu plus haut et automatiquement la transormer en celle ci-dessus.

J'ai donc pu retirer le round autour du chiffre et cela m'a permit de pouvoir dézoomer un peu et c'est avec ca que les lettres ont pu être mieux reconnues :

Maintenant je pense qu'il ne reste "plus qu'à" nettoyer un peu tout ce code qui traine de partout pour tout faire fonctionner et implémenter un peu de parrallel processing ainsi que de l'asynchrone pour ne pas bloquer le reste du programme.

Par ce qu'il faut savoir que en l'étât, le programme met 25 secondes à démarrer et consomme presque 2GB de Ram. Certes cela ne veut pas dire que la reconnaissance à elle seule prend 25 secondes car au démarrage il y a aussi la lecture du fichier de config et la création des window etc..

En réalité la partie strictement OCR prend dans les 12s si on en croit la fonction stopWatch de C#.

Et quand on change d'image la reconnaissance prend 9s.

Dans tout les cas c'est BEAUCOUP trop.

J'aurais eu comme objectif de faire une reconnaissance toutes les secondes. Je ne sais pas bien si cela va être possible mais en tout cas le but va être de s'en rapprocher.

Pour être plus exact et permettre une comparaison, voici les stats exactes

Avec un fichier d'images vide :

- Loading - 11.8s
- Splitting d'images - 90ms
- OCR - 12.5s

Avec un fichier d'images plein :

- Loading - 10.8s
- Splitting d'images - 80ms
- Ocr - 11.6s

En passant d'une image à l'autre :

- Loading - NaN
- Splitting d'images - 50ms
- Ocr - 8.8s

Donc on peut voir que les deux endroits ou le programme prend le plus de temps c'est au premier démarrage quand il faut lire le fichier et setup les windows etc...

Et l'OCR qui prend un temps fou.

Ce qui est pratique c'est que les presque 2gb de ram sont utilisés que au lancement et ensuite l'application n'en utilise que quelques centaines de mb.

Le processeur lui tourne entre 10 et 20% ce qui ne va pas durer :)

Je vais m'occuper dabord du loading.

J'ai essayé d'utiliser un Parrallel.For au moment de la création des windows, le problème c'est que visiblement les objets windows sont beaucoup trops complexes et utilisent trop de ressources partagées pour être vraiment thread Safe. J'éspère que je n'aurais pas trop de soucis avec ca qu'en j'en viendrai à l'optimisation de l'OCR...

Ce qui me rend fou c'est que cette boucle toute nulle prend plus de dix secondes à s'executer et je ne comprend pas bien pourquoi.

```
for (int i = 0; i < NUMBER_OF_DRIVERS; i++)
                    {
                        Point tmpPos = new Point(0, FirstZonePosition.Y + i * FirstZoneSize.Height - Convert.ToInt32(i * offset) /*- (i*
(FirstZoneSize.Height / 32))*/);
                        Zone newDriverZone = new Zone(MainZoneImage, new Rectangle(tmpPos, FirstZoneSize));
                        Bitmap zoneImg = newDriverZone.ZoneImage;

                        newDriverZone.AddWindow(new DriverPositionWindow(zoneImg, new Rectangle(driverPositionPosition, driverPositionArea)));
                        newDriverZone.AddWindow(new DriverGapToLeaderWindow(zoneImg, new Rectangle(driverGapToLeaderPosition, driverGapToLeaderArea)));
                        newDriverZone.AddWindow(new DriverLapTimeWindow(zoneImg, new Rectangle(driverLapTimePosition, driverLapTimeArea)));
                        newDriverZone.AddWindow(new DriverDrsWindow(zoneImg, new Rectangle(driverDrsPosition, driverDrsArea)));
                        newDriverZone.AddWindow(new DriverTyresWindow(zoneImg, new Rectangle(driverTyresPosition, driverTyresArea)));
                        newDriverZone.AddWindow(new DriverNameWindow(zoneImg, new Rectangle(driverNamePosition, driverNameArea)));
                        newDriverZone.AddWindow(new DriverSector1Window(zoneImg, new Rectangle(driverSector1Position, driverSector1Area)));
                        newDriverZone.AddWindow(new DriverSector2Window(zoneImg, new Rectangle(driverSector2Position, driverSector2Area)));
                        newDriverZone.AddWindow(new DriverSector3Window(zoneImg, new Rectangle(driverSector3Position, driverSector3Area)));

                        MainZone.AddZone(newDriverZone);
                    }
```

Alors que Zone.AddWindow c'est simplement :

```
public virtual void AddWindow(Window window)
        {
            Windows.Add(window);
        }
```

Et windows est simplement une liste. Donc ca ne peut pas être ca qui prend du temps.

Et les windows que je créé ont ca comme code :

```
public DriverPositionWindow(Bitmap image, Rectangle bounds) : base(image, bounds)
        {
            Name = "Position";
        }
```

Sachant que le constructeur de base d'une Window c'est :

```
public Window(Bitmap image, Rectangle bounds)
        {
            Image = image;
            Bounds = bounds;

            Engine = new TesseractEngine(TESS_DATA_FOLDER.FullName, "eng", EngineMode.Default);
            Engine.DefaultPageSegMode = PageSegMode.SingleLine;
        }
```

Sachant que TesseractEngine est en statique et que donc il ne devrait... OHLLALALALALALALALALA je suis un imbécile...

J'ai juste à changer ce constructeur avec ca:

```
if (Engine == null)
            {
                Engine = new TesseractEngine(TESS_DATA_FOLDER.FullName, "eng", EngineMode.Default);
                Engine.DefaultPageSegMode = PageSegMode.SingleLine;
            }
```

ET le loading ne prend plus que 2-300 ms...

Bon c'est une très belle amélioration pour pas très chèr mais bon c'est un peu bête...

Bon je pense que 2-300ms c'est une durée correcte surtout que ca n'est appelé qu'une fois pour le lancement. On peut passer à la suite maintenant.

Alors il y a un grand soucis avec la parallellisation de l'OCR... Tesseract n'est pas par défaut une classe "Thread safe" ce qui veut dire que je ne peut utiliser de parallell.Foreach sur mes windows pour accèlèrer le traitement drastiquement.

Je pourrais par exemple avoir une instance de Tesseract par window sauf que cela fait 20 pilotes * 9 windows chacuns ce qui donne 180 instances ce qui n'est tout simplement pas raisonnable.

Je vais donc essayer de voir avec l'utilisation de methodes asynchrones qui me permettraient de faire un genre de flux tendu de reconnaissance. J'avoue que la je navigue un peu à vue, je me base sur différentes infos que je trouve sur des sites un peu perdus et sur chatGPT, j'espère que j'arriverai à trouver une solution car 10 secondes de reconnaissance c'est vraiment beaucoup trop.

Alors le soucis avec un Engine unique entre toutes les windows c'est qu'il n'est pas possible de process plusieurs images à la fois.

Je vais donc retirer l'engine unique pour voir si en créer un par window me permet de passer en multithreading.

La grande question sera : Est-ce que les ressources supplémentaires que vont prendre la création de tous ces engines va compenser entièrement le temps gagné avec la paralellisation.

Pour stocker les données dans un premier temps je vais créer un objet DriverData. Ce qu'il y a de pratique avec ca, c'est que je pourrais ajouter du code de vérification de certaines données directement dedans avant de les donner à la suite du programme.

Et on peut même imaginer une implémentation d'une liste de DriverData pour avoir l'historique.

Ce qui serait cool ca serait de grouper toutes ces data avec un numéro de tour. Placer ensuite la liste de Data dans une DB serait ainsi super simple. Mais il va falloir savoir quoi mettre, quelles infos sont redondantes et prendre en compte le fait que un tour affiché sur la page de la F1TV n'est accompli que par certains des premiers pilotes. D'autres pilotes peuvent être dans des tours précédents si ils ont du retard.

Il faudra réfléchir à cela quand je viendrai au modèle.

Bon pour y arriver j'ai du faire de gros changements et le résultat n'est peut-être pas aussi cool que ce que j'aurais voulut...

Voici un petit point sur les performances maintenant J'ai également désactivé le dump d'images. Pour le moment j'ai tout mis en commentaire mais cela pourrait être intéressant de faire en sorte de pouvoir l'activer en changeant une ou deux variables

Au démarrage :

- Loading - 113ms
- Splitting d'images - 14ms
- Ocr - 7s

En passant d'une image à l'autre :

- Loading - 113ms
- Splitting d'images - 13ms
- Ocr - 5s

Alors clairement les stats montrent qu'il y a eu un changement mesurable mais bon je pensais pouvoir en gagner un peu plus...

Je soupconne la création d'engines d'être à l'origine de ces performances presque décevantes.

Autre soucis, il semble que plus je change d'image plus la detection est lente et plus je consomme de RAM.

Il va falloir que je travaille encore un peu.

J'ai tenté de mettre un stopwatch sur une des créations d'engine Tesseract et le résultat me parait fou... Plus d'une seconde c'est dingue.

J'ai testé dans d'autres endroits du code et effectivement il semble que la création d'un engine prenne entre une et deux secondes ce qui est une ETERNITEE what !

Donc il faut optimiser tout ca.

Une idée serait de décomposer le threading mais cela me demanderait un gros refactor et je n'ai pas envie d'en refaire un la...

Sinon, une fois qu'ils sont créés ils ne prennent pas de temps du tout. Créer une fois tous les engines et ensuite les utiliser pourrait être une bonne idée. Cela prendrait longtemps au load mais ensuite les reconnaissances devraient être super rapides.

Ok alors ca c'est déja plus ce à quoi je m'attendais ! On est de nouveau à plus de 10s de loading time mais on est descendu à deux secondes par OCR. (Bon autre soucis, l'utilisation de la RAM est ridicule plus de 2gb mais ce qui m'inquiète c'est que j'ai l'impression qu'elle augmente plus on change d'image)

J'ai règlé (en partie) le soucis en obligeant le GC (Garbage Collector) à collecter après chaque detection. même après 50 detections l'utilisation de la ram se stabilize autour des 2GB.

Bon en paralellisant la création des Engines le soucis c'est que cela demande d'allouer beaucoup trop de mémoire d'un coup alors le programme se fige pendant genre cinq secondes avant de tout créer. Du coup même si la création est plus rapide, on se retrouve avec un temps total plus long... Je pense que l'on va devoir se contenter de ces dix secondes.

Bon la j'allais tenter de faire la documentation mais je viens de me rendre compte que la detection de temps au tour est pas vraiment encore idéale...

J'ai réussi à changer un petit peu le programme de reconnaissance pour rendre la reconnaissance un peu meilleure mais cela a drastiquement augmenté le temps requis pour décoder... On arrive à 3.5 secondes.

Je vais tenter de rajouter un peu de parralell processing sur les boucles de traitement voir si cela peut aider.

Alors effectivement cela aide pas mal, on arrive maintenant à faire une detection presque tout le temps en dessous de la seconde.

Et j'ai aussi du changer un peu le fonctionnement de la detection des Temps au tour.

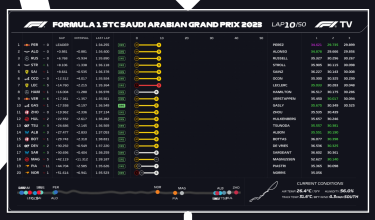
Et voila je pense que je vais m'arrêter la pour la partie décodage. Je ne pense pas que je peux facilement faire mieux que ca et il faut que j'avance dans d'autres parties du projet.

Je vais pouvoir commencer à documenter un peu toute la partie OCR. Il faut que je prenne le temps de le faire bien car c'est la partie la plus intéressante du projet et ou je pense que j'aurai le plus essayé de choses qui vallent le coup dêtres racontées.

J'ai aussi passé pas mal de temps sur le poster du projet. J'avais fait des croquis au crayon de ce à quoi je pensais, cependant après de longues discussions avec M.Garcia ils n'étaient pas forcément très bons car ils ne représentent pas assez bien le fonctionnement du projet et sont un peu trop marketings.

Du coup j'ai fait une première version au propre :

Track Trends



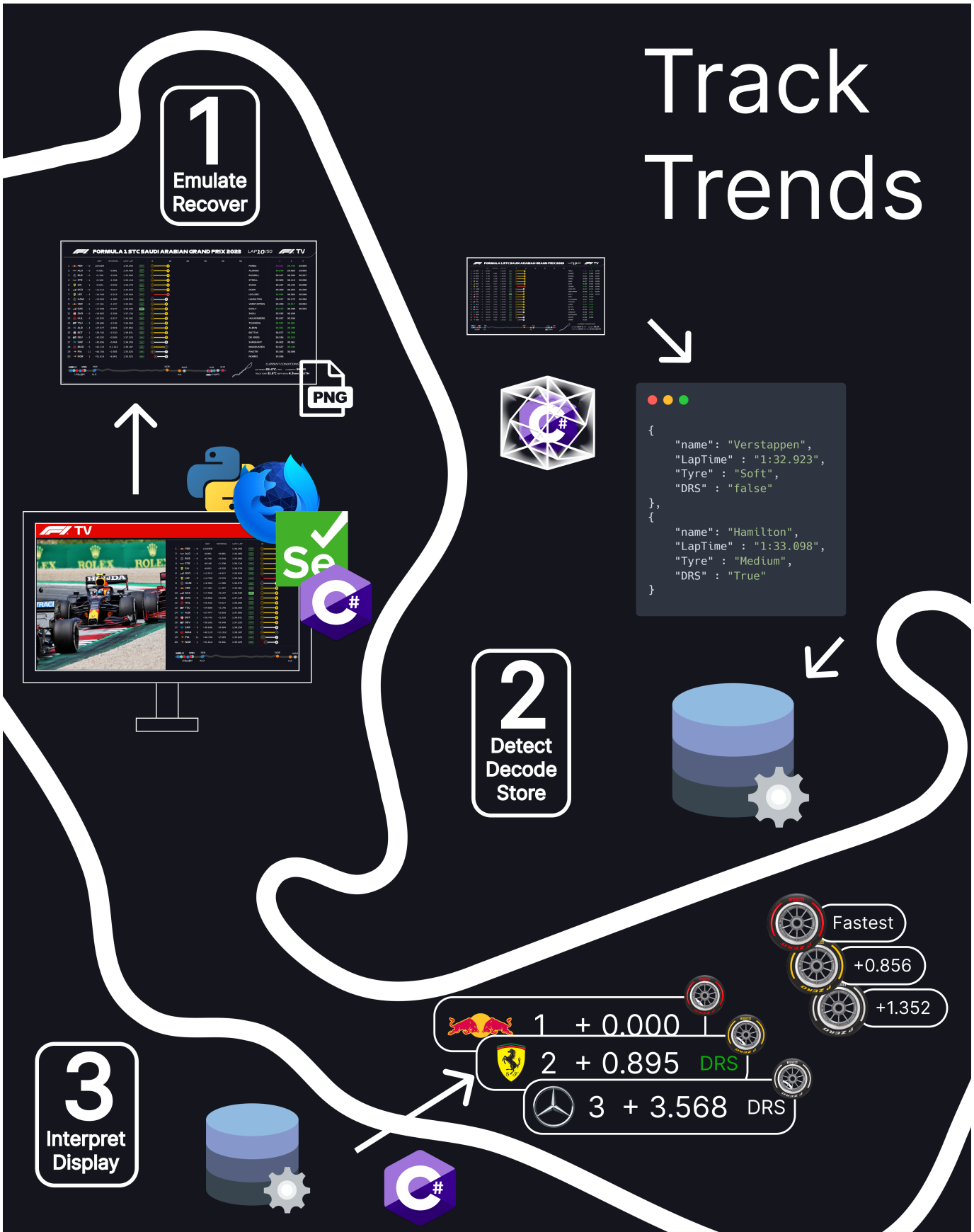
{Verstappen;1:32:923;Tyre:Soft;Drs:false}
{Leclerc;1:32:923;Tyre:Medium;Drs:true}

Maxime Rohmer
Travail de diplôme Tech II 2023

Headless Firefox · Csharp · Python · MySQL · Tesseract OCR

Mais je n'étais pas forcément content du résultat et il manquait des choses je trouve comme par exemple l'utilisation de Selenium.

J'ai donc repassé des heures à faire une seconde version :


Track
Trends
1
Emulate
Recover
PNG
{
"name": "Verstappen",
"LapTime" : "1:32.923",
"Tyre" : "Soft",
"DRS" : "false"
},
{
"name": "Hamilton",
"LapTime" : "1:33.098",
"Tyre" : "Medium",
"DRS" : "True"
}
2
Detect
Decode
Store
Fastest
+0.856
+1.352
1 + 0.000
2 + 0.895 DRS
3 + 3.568 DRS
3
Interpret
Display
Maxime Rohmer
Diploma work Tech II 2023
Tesseract OCR
CSharp
Selenium
Python
Headless Firefox

La police d'écriture n'est pas encore la bonne mais cela va venir. Mais je préfère déja beaucoup cette version à la première.

Je ne sais pas encore si la version finale sera une version plus travaillée de ce poster ou complêtement autre chose mais pour l'instant je suis à peu près content de cette version.

Je le trouve un tout petit peu trop brouillon ou avec trop d'infos mais il m'a été de nombreuses fois reproché de ne pas assez montrer le fonctionnememt interne et je ne peux pas faire plus simple.

L'ajout des nombres pour compartimenter le projet ajoute de la structure mais je me demande si cela suffit.

Maintenant que je suis à peu près content de mon code pour l'OCR je vais commencer sa documentation. (Uniquement son fonctionnement interne pas comment s'en servir car cela va changer)

Bon j'ai créé u nouveau projet selenium mais même avec les bonnes libraries je n'arrivais pas à faire fonctionner firefox j'avais toujours une erreur "OpenQA.Selenium.WebDriverException: 'Cannot start the driver service on http://localhost:51481/'" et j'ai pu règler le problème en téléchargeant directement le gecko driver depuis le git https://github.com/mozilla/geckodriver/releases et utiliser le fichier directement dans le service :

```
var service = FirefoxDriverService.CreateDefaultService(AppDomain.CurrentDomain.BaseDirectory+@"geckodriver-v0.27.0-win32\geckodriver.exe");

FirefoxOptions options = new FirefoxOptions();

var driver = new FirefoxDriver(service,options);
```

Le seul problème c'est que du coup il faut tout le temps déplacer le fichier dans le dossier bin si je clone le projet. Il faudra faire un installeur dans la version finale qui s'occupe de tout je pense.

Je me suis dit que j'allais garder la doc pour le retour des vacances quand j'aurai un bureau un clavier et un setup complet un peu propres.

Bon il va falloir que je parle de la récupération de cookie. J'ai déja pu travailler lors d'un poc sur la meilleure facon de prendres des screenshots de la F1TV :

- Avoir une page chrome ouverte avec le feed en plein écran et un programme qui prend des captures d'écrans.
- Avoir une caméra qui prend en photo l'écran au cas ou chrome et Firefox empêchent la prise de captures d'écrans.
- Récupèrer directement le feed en faisant du reverse engeneering de la plateforme.
- Simuler un chrome en background qui prenne des screenshot sans qu'on aie à le voir.

Dans toutes ces options, je dirais que la pire était celle de la caméra qui filme l'écran, mais à l'époque c'était encore envisageable comme solution de dernier recours. Le soucis de cette solution c'est l'horreur que serait la partie OCR avec une image de très mauvaise qualité.

Une autre option qui m'aurait vraiment embêté aurait été de devoir garder une page de Chrome ou Firefox ouverte quelque part sur un écran pour que le programme puisse prendres des captures d'écrans. C'est de loin l'option la plus simple et la plus logique mais elle possède pour moi de très gros points noirs :

- On ne peut pas certifier l'intégrité des données car l'utilisateur a le contrôle total sur le feed. Il peut mettre pause, avancer, reculer, tout casser sans faire exprès en ouvrant autre chose sur son ordi qui se mette pile devant. Bref c'est un peu bancale.
- Et surtout on bloque une partie non significative de l'écran de l'utilisateur avec des infos redondantes. Et je peux vous dire que quand je commente la F1 j'ai besoin de beaucoup d'informations et que chaque centimètre d'écran est cruciaé ! Alors avoir un écran complet bloqué est juste un point bloquant qui m'empêcherait d'utiliser l'app aussi bonne soit-elle dans ses prédictions.

Mais bon si aucune autre methode ne fonctionne ce qui est bien c'est que celle la est plutôt simple à mettre en place.

Ensuite reverse engeneer le feed serait l'option la plus classe, cependant c'est la plus complexe et la plus bancale au niveau légal haha. L'idée serait de récupèrer le lien vers le broadcast général et de comprendre comment il fonctionne pour le décoder nous même pendant un Grand Prix. Seuls soucis :

- Il n'est pas possible de faire des tests en dehors des periodes de Grand Prix (Et je rappelle que c'est des périodes ou je travaille en plus)
- Difficile de faire un système qui marche pareil pour les rediffusions et les lives. (En effet les liens des rediff sont beaucoup plus simple à récupèrer mais ne fonctionnent pas du tout pareil et pour tester l'app il est essentiel de pouvoir s'entrainer sur des anciens Grand Prix)
- Dernier GROS soucis, je ne sais tout simplement pas faire ca lol. Je ne sais pas comment faire. Peut-être que avec des profs qui m'aident et chat gpt ainsi qu'internet je pourrais potentiellement négocier un truc mais c'est hautement improbable et cela serait une perte de temps folle si je n'y arrive pas.

Dernière option que je trouve la plus séduisante. Simuler une instance de Chrome ou de Firefox (Le soucis avec chrome c'est qu'il implémente l'utilisation de DRM dans les vidéos qui fait qu'il est très difficile de passer outre la sécurité avec un bot) pour ensuite prendre des captures d'écrans automatiquement. Cette solutions offre pleins d'avantages :

- Pas de place prise sur l'écran
- L'intégrité des données est assurée car c'est le programme qui décide d'ou partir et de si il met pause ou non
- C'est une option complexe mais beaucoup moins que le reverse engeneering
- Elle permet de ne demander presque aucun input de la part de l'utilisateur.

Mais elle pose quelques problématiques :

- Comment se connecter automatiquement sans être detecté par un Bot et sans demander à l'utilisateur ses identifiants (Pour des raisons évidentes qui sont : QUI VA METTRE SES IDENTIFIANTS SUR UNE VIEILLE APP COMME LA MIENNE??)
- Comment faire en sorte que le programme prenne les meilleures captures dans la meilleure qualité et en plein écran.

Mais j'ai décidé de partir sur cette option.

Pour ce faire j'utilise Selenium. J'ai pu tester Puppetteer Sharp et même si dans un premier temps j'ai pu avancer asez vite, malheureusement il y a des bugs qui rendent son utilisation impossible dans notre contexte.

J'ai donc décidé de tout faire en utilisant un portage de Selenium dans mon programme.

Voici un exemple de code qui va ouvrir FireFox et qui va lancer un RickRoll

```
var service = FirefoxDriverService.CreateDefaultService(AppDomain.CurrentDomain.BaseDirectory+@"geckodriver-v0.27.0-win32\geckodriver.exe");
            service.Host = "127.0.0.1";
            service.Port = 5555;

            FirefoxOptions options = new FirefoxOptions();
            options.AddArgument("--disable-headless");

            var driver = new FirefoxDriver(service,options);

            driver.Navigate().GoToUrl("https://www.youtube.com/watch?v=dQw4w9WgXcQ&autoplay=1&mute=1");
```

Dans cet exemple on désactive le "Headless" pour qu'on puisse voir ce que fait l'app car sinon tout est invisible. Alors dans les faits la vidéo youtube ne se lance pas du tout car il y a des pubs et des prompts de cookies que l'on doit accepter etc... ce qui montre les différents challenges que l'on va devoir surmonter pour vraiment faire ce que l'on veut.

Mais un petit détail extrêmement important, la F1TV est un programme payant un peu comme netflix. Ce qui veut dire que pour accèder au contenu il faut être connecté. Sauf que une instance de firefox créé par Selenium est comme une page de naviguation privée, ce qui veut dire que si on va sur la page de la F1TV on est pas connectés.

Je pourrais tout à fait demander à l'utilisateur de me donner ses identifiants pour que j'aille ensuite automatiquement me connecter sauf que cela pose deux soucis:

- Personne ne voudra mettre ses identifiants sur mon programme
- La page de login de la F1TV a été protègée avec la meilleure technologie de detection de bots que je connaisse. Presque aucun site n'arrive à me detecter sauf eux ! Donc c'est tout simplement impossible d'utiliser cette technique.

Ensuite je me suis rappelé que ce que la page stocke pour me permettre de rester connecté ce sont des cookies. Et si je mets le bon cookies dans Selenium alors je serai connecté.

Dans un premier temps je voulais faire un système ou l'utilisateur irait prendre dans son chrome son cookie et le copie colle dans mon programme mais c'est immonde.

C'est alors que vient la partie récupèration de cookies !

Tous les cookies de chrome sont stockés dans une base de données SQLITE. On pourrait se dire Banco il suffit d'aller dedans et de retrouver tous les cookies et se connecter. Sauf que, pas bêtes, les équipes de chrome ont décidé que c'était une bonne idée d'encoder les cookies pour que tout le monde ne puisse pas venir y mettre son nez... En effet les cookies peuvent contenir des informations importantes.

Cela fait que pour utiliser ces cookies il faut pouvoir les décoder. Mon hypothèse a été que si ces cookies peuvent être lus par Chrome même hors connexion, c'est que la clé de décodage existe sur l'appareil et qu'il suffit de la trouver. ET C'EST LE CAS! Après pas mal de recherches j'ai pu voir que la clé de décodage existe bel et bien et qu'il suffit de la décoder en utilisant la librairie DPAPI pour la lire.

Avec cette clé on peut ensuite décoder les cookies et leurs valeurs ce qui veut dire qu'il est théoriquement possible d'automatiser le processus sans que l'utilisateur n'aie rien à faire.

J'ai décidé de faire la partie récupèration en python pour deux raison :

- Je n'arrivais pas à trouver une bonne implémentation de DPAPI en C# qui me permettait de décoder la clé.
- Il existe beaucoup plus de documentation en Python pour ce qui est de la cryptographie et donc si Chrome change de fonctionnement il sera beaucoup plus simple de changer cette partie en particulier sans avoir à recompiler le code C#.

J'ai donc avec l'aide d'internet et de ChatGPT créé ce script :

```
def get_master_key():
    with open(
        os.getenv("localappdata") + "\\Google\\Chrome\\User Data\\Local State", "r"
    ) as f:
        local_state = f.read()
        local_state = json.loads(local_state)
    master_key = base64.b64decode(local_state["os_crypt"]["encrypted_key"])
    master_key = master_key[5:]  # removing DPAPI
    master_key = win32crypt.CryptUnprotectData(master_key, None, None, None, 0)[1]
    print("MASTER KEY :")
    print(master_key)
    print(len(master_key))
    return master_key

def decrypt_payload(cipher, payload):
    return cipher.decrypt(payload)

def generate_cipher(aes_key, iv):
    return AES.new(aes_key, AES.MODE_GCM, iv)

def decrypt_password(buff, master_key):
    try:
        iv = buff[3:15]
        payload = buff[15:]
        cipher = generate_cipher(master_key, iv)
        decrypted_pass = decrypt_payload(cipher, payload)
        decrypted_pass = decrypted_pass[:-16].decode()  # remove suffix bytes
        return decrypted_pass
    except Exception:
        # print("Probably saved password from Chrome version older than v80\n")
        # print(str(e))
        return "Chrome < 80"


master_key = get_master_key()

cookies_path = Path(
    os.getenv("localappdata") + "\\Google\\Chrome\\User Data\\Default\\Network\\Cookies"
)

if not cookies_path.exists():
    raise ValueError("Cookies file not found")

with sqlite3.connect(cookies_path) as connection:
    connection.row_factory = sqlite3.Row
    cursor = connection.cursor()
    cursor.execute("SELECT * FROM cookies")

    with open('cookies.csv', 'a', newline='') as csvfile:
        fieldnames = ['host_key', 'name', 'value', 'path', 'expires_utc', 'is_secure', 'is_httponly']
```

```
        writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

        if csvfile.tell() == 0:
            writer.writeheader()

        for row in cursor.fetchall():
            decrypted_value = decrypt_password(row["encrypted_value"], master_key)
            writer.writerow({
                'host_key': row["host_key"],
                'name': row["name"],
                'value': decrypted_value,
                'path': row["path"],
                'expires_utc': row["expires_utc"],
                'is_secure': row["is_secure"],
                'is_httponly': row["is_httponly"]
            })

print("Finished CSV")
```

Ce programme va faire tout ce que j'ai expliqué et va ensuite stocker les résultats dans un CSV pour qu'il soit facile d'y accèder depuis le C#.

Alors oui cela pose certaines questions de sécurité. Car en effet je prend tous les cookies, les décode et les stocke. Ce qui veut dire que je pourrais tout à fait envoyer ces données quelque part, par exemple un compte Netflix, et me rincer.

Si je devais rendre le projet ouvert au public je pense qu'il faudra que cela soit mentionné clairement et que le projet soit open source pour que les utilisateurs puissent verifier que je ne fais pas ca.

Maintenant de l'autre côté j'ai juste à lire le CSV et le tour est joué !

(Trouver cette solution m'a pris une semaine de vacances à l'époque)

Bon j'ai réussi à faire le programme se connecter et naviguer etc..

Par contre quelque chose que j'ai voulu ajouter et qui m'a pris pas mal de temps c'est de faire en sorte de pouvoir selectionner la qualité.

Pour changer la qualité du feed il faut cliquer sur settings et ensuite prendre le menu deroulant et selectioner 1080p. Le soucis c'est le que la value du select est jamais la même.

Elle commence toujours pas "1080_" mais ensuite ca peut être "1080_45930285" ou "1080_56801" la suite est apparemment random.

J'ai donc du utiliser ce code pour le selectioner quand même :

```
IWebElement settingsButton = driver.FindElement(By.ClassName("bmpui-ui-settingstogglebutton"));
                settingsButton.Click();
                IWebElement selectElement = driver.FindElement(By.ClassName("bmpui-ui-videoqualityselectbox"));
                SelectElement select = new SelectElement(selectElement);
                IWebElement selectOption = selectElement.FindElement(By.CssSelector("option[value^='1080_']"));
                selectOption.Click();
```

Sauf que pour que cela marche je dois avant cliquer sur le bouton des settings le problème c'est qu'il est invisible alors on doit le faire apparaitre.

J'ai tenté de le faire aparaitre en bougeant la souris, en cliquant à un endroit précis, impossible de le faire marcher correctement.

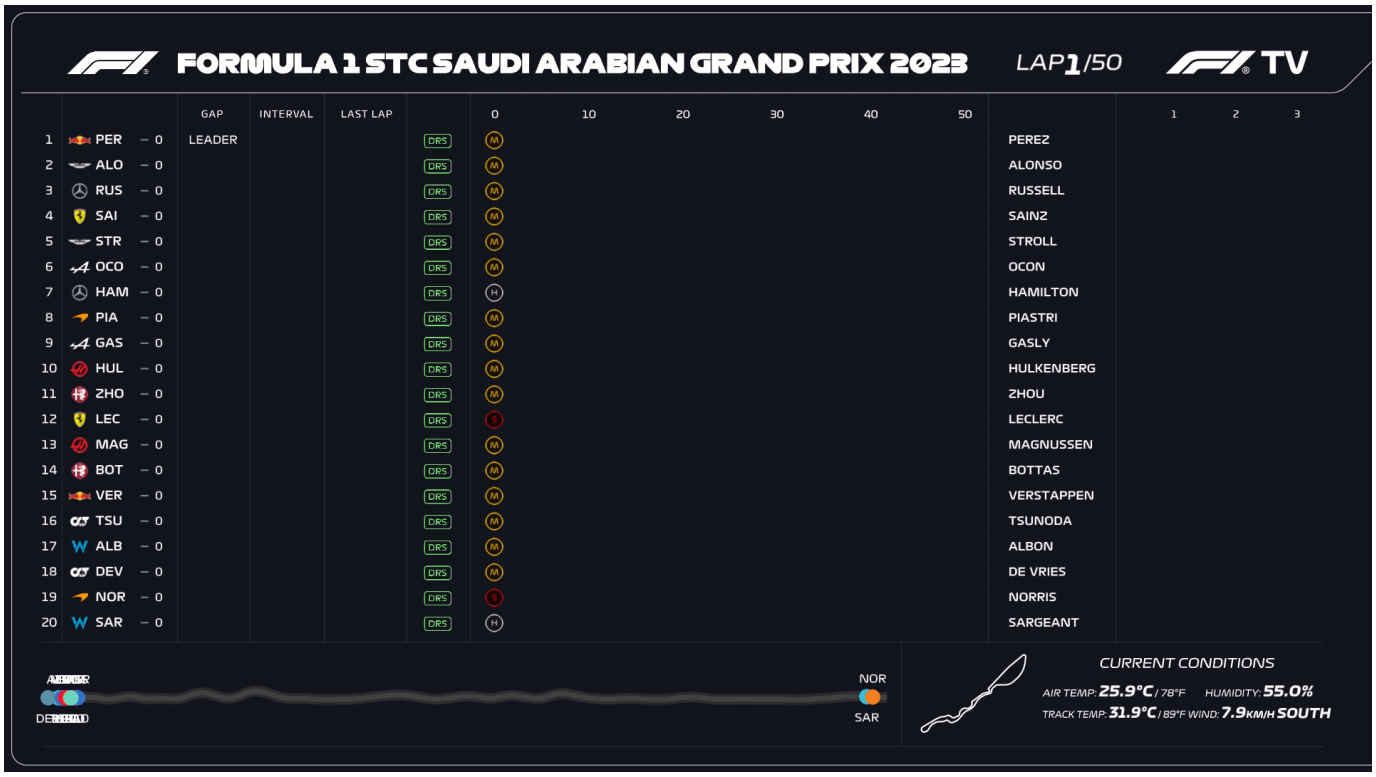
Puis j'ai eu l'idée de mettre pause en envoyant un appui sur la touche Espace et ca a permit de découvrir le bouton et permettre qu'on clique dessus.

Ca peut paraitre tout bête mais rien que ca, ca m'a pris un temps considérable.

Bon pour ce qui est du timecode de la vidéo. Je pense qu'il serait trop complexe de faire en sorte que selenium change le slider de progression de la vidéo. Alors j'ai fait quelques tests et apparemment, si on quitte la F1TV sur un timecode de la vidéo que on donne au programme, comme il récupère tous les cookies de la F1TV il commencera de la.

Donc si on veut utiliser le programme avec des Grand Prix ayant déja eu lieu, on peut le faire, seulement il faudra juste au préalable avoir choisit le bon timecode dans le page de la F1TV avant de le lancer.

Ce qui est intéressant c'est que la page de la F1TV ressemble à ca au départ :



Je pense qu'une bonne idée serait de dire au programme que c'est la grille de départ et ensuite dès qu'il détecte un secteur il sait que la course a commencé.

3.10 Lundi 24 Avril 2023

Aujourd'hui c'est jour de documentation.

J'ai pas mal travaillé pendant les vacances mais je n'ai pas encore pu faire de vraie documentation correcte du fonctionnement. Du coup je vais m'en charger aujourd'hui et peut-être un peu demain.

Ok normalement je ne devrais faire que de la documentation mais je ne peux pas passer à coté de ca... Le problème que j'ai avec les pneus ou parfois il détecte un H au lieu d'un '11' et ce genre de choses c'est à cause de ma methode "RemoveBG" Qui va retirer tous les pixels plus sombres que le background. Sauf que cela va aussi retirer des pixels dans le chiffre lui même et qui va donc defigurer les 11 :

J'ai réussi à les changer en :

Mais au final cela n'a pas augmenté la précision de la reconnaissance. Je pense que je vais donc devoir encore changer.

Je pense que une bonne facon de trouver serait dabord de trouver la couleur du pneu. Et si il n'y a pas assez de couleur alors c'est que le pneu contient une lettre. Le but est d'arrêter de chercher des lettres ou des chiffres. Comme ca les 11 arrêteront d'être pris pour des 'H'

En fait on peut faire encore plus simple que ca.

On peut simplement regarder la couleur dominante et determiner le pneu. En effet même si il y a une lettre sur fond noir pour décrire le pneu, mon methode de récupèration de la couleur dominante ommet les pixels trop noirs alors il est quand même possible de determiner le type de pneus.

Et tout simplement si il n'arrive pas à lire le chiffre c'est que c'est une lettre et que donc on est à 0 tours. Cela marche plutôt bien et cela simplifie pas mal le processing.

Voila, la je vais me remettre à la documentation sinon je vais encore prendre du retard.

3.11 Mardi 25 Avril 2023

Encore une fois j'ai pris du temps de doc pour changer des choses sur la partie OCR. Mais en même temps en documentant je vois des choses que j'ai soit mal fait soit que je pourrais faire mieux en changeant très peu de choses. J'éspère que les changement que j'ai fait vont aider au moins à la cohérence du code et un peu pour les performances.

Il semble que dans les conditions que j'ai testé le nombre de tour soit plutôt fiable mais je pense que je devrai faire un peu de travail en aval dans la récupération de ces données car je sens que cela va poser problème quelques fois. Je pense que en utilisant bien l'historique on peut potentiellement se passer de l'utilisation de ce chiffre pas toujours complêtement fiable.

Mais sinon aujourd'hui c'est encore une fois un gros jour de doc. J'essaie d'expliquer les différents procédés avant de les oublier. J'essaie aussi de donner un maximum d'exemples sous formes de photos intermédiaires mais ca me prend pas mal de temps car il faut que j'ajoute un peu partout dans le code des lignes pour sortir des images intermédiaires.

En plus de la documentation je me suis aussi beaucoup occupé de nettoyer mon code et je suis assez content par ce que même en ayant du rajouter des couches de complexité pour mieux reconnaitres les temps au tour j'arrive à un temps de processing parfois en dessous des 2 secondes ce que je trouve honorable.

Quand j'aurai finit de nettoyer tous mes fichiers je ferai une release sur gitea et ce sera la version que j'utiliserai quand je voudrai faire un merge avec les autres parties du projet.

J'ai beaucoup beaucoup bossé aujourd'hui et je sui bien mort. Faire autant de documentation et de nettoyage de code c'est pas forcément bon pour le cerveau je crois. J'ai besoin d'une sieste.

Demain je pense que je vais commencer à avancer sur la partie récupèration des images. Je sais que la je fais un peu passer les tests à la trappe mais déja j'en ai fait tout le long du développement de OCR_DECODE et il faut vraiment que j'avance, quitte à revenir dessus quand j'aurai merge les deux projets ensemble.

3.12 26 Avril 2023

Aujourd'hui je vais devoir m'occuper de la partie récupération des images. J'ai déja eu l'occasion d'avancer sur ce projet pendant mopn poc et mes vacances. Donc la le but ca va être de voir ce qui manque comme véritables features et ensuite je vais pouvoir m'occuper de la vue et de son intégration avec le décodage.

Ok donc maintenant que j'au un programme qui arrive à prendre des images depuis la F1TV correctement et en bonne résolution. Je pense qu'il est temps de passer à l'implémentation de la Forme que ca va prendre.

C'est important de se poser au moins cinq minutes la question de comment je prévois de faire car même si ca n'est pas la version finale, cette dernière prendra très fort inspiration du desing que je vais faire.

Dans cette form j'aurais besoin de :

- Pouvoir selectionner un Grand Prix en insèrant l'URL du feed.
- Pouvoir lancer la calibration si besoin
- Indiquer le titre et la date du Grand Prix
- Indiquer si le Grand Prix vient de commencer ou si il y a déja un certain nombre de tours lancés.

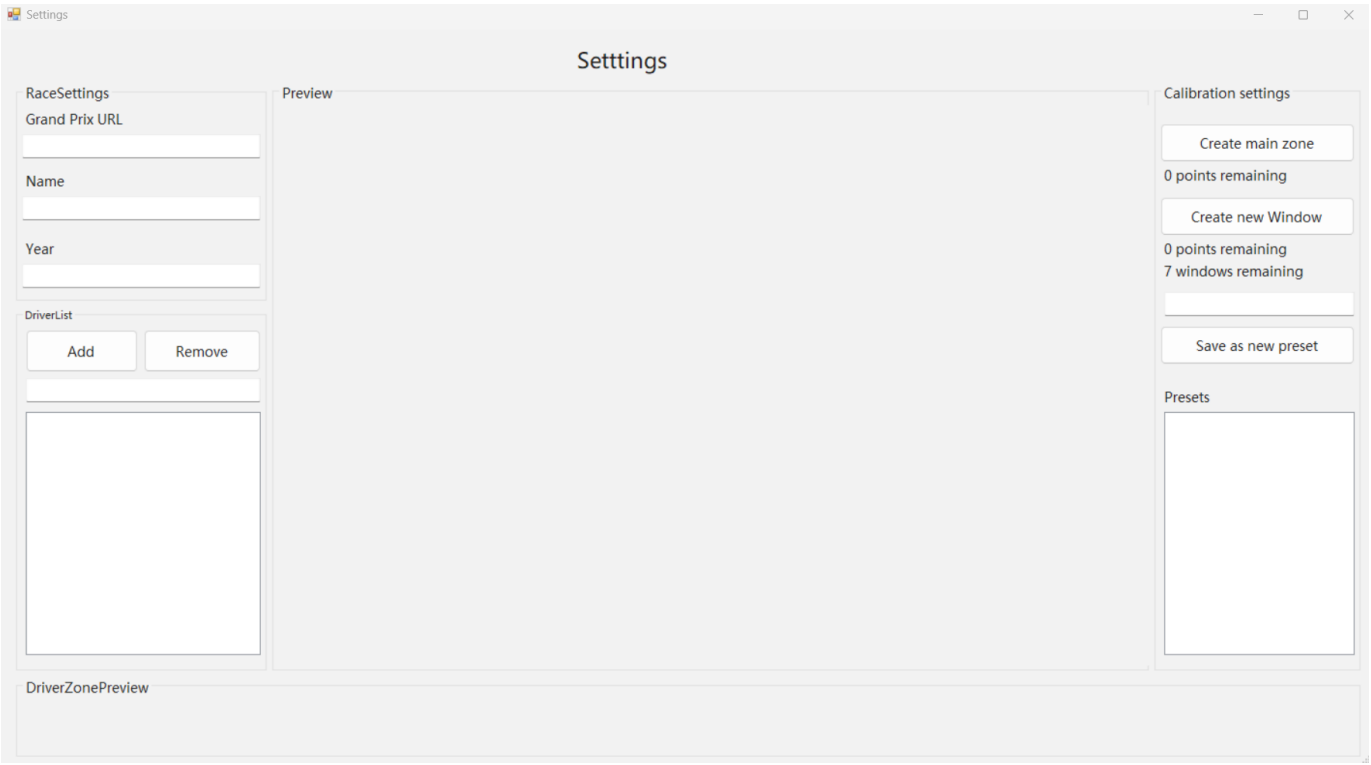
Et c'est à peu près tout en fait...

J'ai tellement poussé pour avoir un programme qui fait tout tout seul que il ne me faut pas grand chose de plus. Je pense que ce qui serait pas mal ca serait du coup d'utiliser ce temps pour bien implémenter la calibration qui elle aura besoin d'une UI un peu plus balèze.

On pourrait même imaginer que la calibration fasse partie intégrante des settings... Ca serait peut-être bien que quand l'application se lance on se retrouve sur la page principale d'affichage de données et qu'on puisse simplement cliquer sur la page options qui contient la page calibration et qui permet de rentrer les infos du Grand Prix.

Je pense que je vais faire ca.

Voici l'interface que j'ai développée pour regrouper tout ca :



La police le style le placement et les couleurs ne sont pas définitfs, cependant je pense que c'est un bon début. Le but maintenant va être de permettre de faire fonctionner la calibration et la récupèration d'images.

Si j'arrive à faire fonctionner ces deux choses sur un même projet avant la fin de la semaine cela serait super !

Bon J'ai pu avancer sur l'intègration de Selenium mais cela prend un peu de temps car je veux implémetner un moyen de pouvoir prendre une Screenshot à nimporte quel moment et pas juste en boucle.

Demain je finis de faire fonctionner ca et ensuite je commence le cablage du reste.

3.13 Jeudi 27 Avril 2023

C'est assez dur de faire l'importation car il y a des petites différences qui obligent à presque tout réécrire.

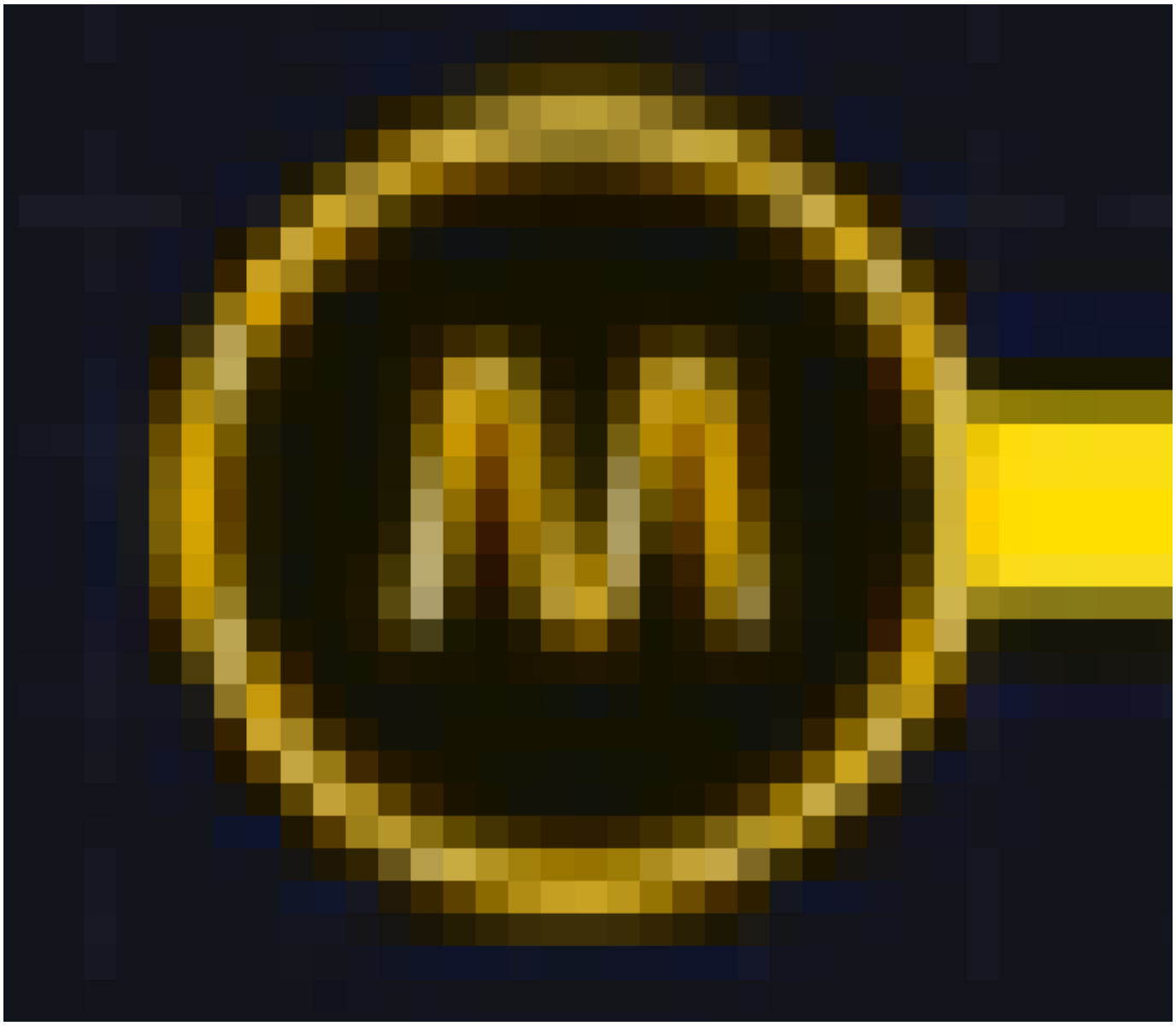
En fait le programme de calibration avait déja implémenté la fonction de Windows et de Zones mais il fonctionnait juste assez différemment pour qu'il faille tout refaire.

La je suis en train de perdre énormément de temps à cause d'un soucis de coordonnées. J'ai repris le code de la calibration pour detecter ou l'utilisateur a cliqué pour créer les zones. Cependant, je n'arrive pas à le faire fonctionner correctement. La zone est tout le temps décalée en haut et en bas mais pas de la même facon. En haut, la valeur Y est trop grande alors que en bas la valeur Y est trop petite... Je ne comprends pas bien pourquoi.

Si c'était un simple décalage cela ne serait pas compliqué à gèrer mais la...

J'ai un soucis également avec la résolution des screenshots que je récupère en full Headless.

Voici un exemple de résolution que j'arrive à récupèrer sans le headless :



Il y a clairement un soucis et le problème c'est que avec une résolution pareille, impossible de faire une reconnaissance correcte.

BON J?EN PEUX PLUS LA. Ca fait des heures que je bosse sur ce problème débile et impossible de trouver une solution. J'ai essayé cinq facons de forcer le browser headless a prendre une plus haute résolution aucune ne fonctionne je ne comprends pas.

A chaque fois que je me retrouve avec une résolution de 1366 x 768 Ou une variante de basse résolution du style. J'en peux plus je ne trouve aucune réponse sur internet ni même avec chatGPT.

Super... La seule chose que j'ai pu faire qui change quelque chose fait que les images font maintenant du 926x517... j'ai un peu envie de commentre un crime de guerre au plus vite.

3.14 Vendredi 28 Avril 2023

Une des solutions que je n'ai pas encore essayé est de changer ma version de GeckoDriver. Sauf que ca m'oblige à changer les versions de mes libraries ce qui est très pénible, je vais continuer le debugging dans le projet Selenium_clean.

Il faut savoir que la librairie de Selenium que j'utilise est bloquée en 0.27 ce qui fait que je ne peux utiliser qu'une version obsolète du Gecko Driver.

J'ai tenté de changer vers une version en 64 bits du GeckoDriver 0.27 mais pareil, je me retrouve toujours avec des images de M.

J'essaie toutes les solutions que je trouve sur internet aucune ne convient c'est infernal. J'essaie de changer la résolution DPI, j'essaie de changer les paramêtres par défaut des player de Firefox, j'essaie de changer la résolution pendant et au début de l'execution IMPOSSIBLE DE FAIRE MARCHER CETTE MERDE C'EST PAS POSSIBLE !!!

J'ai essayé avec chrome mais je ne peux pas l'utiliser car les DRM m'empêcheront de prendre des screenshot du flux vidéo.

J'ai essayé de faire tourner avec edge mais edge ne peut pas tourner en headless. JE VAIS DEVENIR FOUF FPWQOVMQEKOVNVIBDBJDAIVOBI.

ET MAINTENANT JE N'ARRIVE PLUS A FAIRE DE PROJET AVEC SELENIUM VOIWQNV(UEWQBVU)WEQN=OEJNIVIUWVBWUEV

ON CHERCHE A ME FAIRE PETER UN PLOMB C'EST PAS POSSIBLE GIWEGUWEQN

VOICI UN EXEMPLE DU CODE QUE JE DEMANDE A UN NOUVEAU PROJET AVEC EXACTEMENT LES MEMES LIBRARIES INSTALLEES :

```
// Create a new FirefoxDriver instance
            IWebDriver driver = new FirefoxDriver();

            // Navigate to the specified URL
            driver.Navigate().GoToUrl("https://www.example.com");

            // Do something with the driver (e.g., find elements or take screenshots)

            // Quit the driver
            driver.Quit();
```

Je ne demande que ca ET MEME CA CA NE VEUT PAS FONCTIONNER VOIWENB)IWUQENV

Oui je suis un peu énervé ca se voit? A bon?

Et maintenant NUGGET ne fonctionne plus... j'en peux plus la.

Je ne peux plus télécharger de librairie sur aucun de mes projets...

J'ai tenté de supprimer le fichier de config et redémarrer Visual Studio mais cela ne fait rien. J'ai aussi tenté de faire un 'nugget restore' toujours rien.

Bon apparemment je ne suis pas le seul qui ne peut pas accèder à Nuget donc bon c'est pas juste chez moi qu'il y a un soucis.

Mais même en mettant ma 4G pour me connecter, je n'arrive pas à accèder à certains sites y compris Nuget et je ne peux pas download de librairies...

Je ne comprends pas ce qui se passe et du coup je ne peux juste pas bosser... J'ai redémarré trois fois mon pc et visual studio, j'ai essayé de changer mes settings DNS etc... impossible de bosser.

Je crois que je n'aurais pas du me reveiller aujourd'hui.

Bon je vais tenter d'avancer sur mon poster en attendant que le réseau soit en meilleur état.

3.15 Lundi 1 Mai 2023

Bon je bosse depuis chez moi donc j'espère que Nuget va mieux fonctionner.

Après un weekend à réfléchir au sujet de cette resolution je me suis dit deux choses.

1. La seule personne sur internet que j'ai vu avoir le meme soucis avait une résolution de 1920x1200 comme moi. Cela veut donc sûrement dire que le soucis vient de cette résolution de laptop comme moi.
2. Si vraiment je n'arrive pas dans un premier temps à faire fonctionner le Headless correctement, je peux toujours laisser la page de côté et m'occuper du reste du programme. Certes ca serait vraiment infernal d'avoir à garder une page chrome ouvert en tous temps et en plus elle doit être en plein écran mais bon... Si il n'y a vraiment pas d'autres solutions malheureusement je serai bien obligé.

BON ! JE N'ARRIVE MEME PLUS A FAIRE UN PROJET QUI UTILISE SELENIUM ET QUI MARCHE JE VAIS FAIRE BRÛLER GENEVE. C'est pas possible serieux, je ne comprends pas j'essaie tout ce que je trouve et impossible de juste lancer firefox c'est du grand nimporte quoi. Je prend les même putain de librairies que sur les autres projets les mêmes versions, je prend le même exact code. Sur le nouveau projet impossible de le faire fonctionner. Je commence à croire que on essaie de me faire pêter un cable.

Du coup dans un élan de désespoir je vais tenter de passer sur une autre librairie qui avec un peu de chance marche et en plus me permettrais de prendre des foutues screenshot dans le bon format.

Les deux seules librairies qui pourraient potentiellement faire l'affaire sont les librairies :

- PhantomJS
- CefSharp

Je vais les tester et simplement prier pour qu'elles fonctionnent et que je puisse faire ce que je veux avec.

Alors pour le moment avec CEFSharp j'arrive à lancer une instance de chrome et prendre une screenshot avec ce code :

```
CefSettings settings = new CefSettings();
            settings.CachePath = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData), "CefSharp\\Cache"); // Set cache path
            settings.LogSeverity = LogSeverity.Disable; // Disable logging
            Cef.Initialize(settings); // Initialize CEF

            using (var browser = new ChromiumWebBrowser("www.google.com", new BrowserSettings())) // Launch Chromium in off-screen mode
            {
                browser.Load("https://www.example.com"); // Navigate to the test URL
                browser.Size = new Size(1920, 1080); // Set the browser size to 1920x1080
                browser.ScreenshotAsync().ContinueWith(task =>
                {
                    var bitmap = task.Result;
                    bitmap.Save("screenshot.png", System.Drawing.Imaging.ImageFormat.Png); // Take a screenshot and save it as a PNG file
                }).Wait();
            }

            Cef.Shutdown(); // Shutdown CEF
```

Avec ca il faut ces using :

```
using System;
using System.Drawing;
using System.IO;
using CefSharp;
using CefSharp.OffScreen;
```

C'est assez prometteur même si il faut encore beaucoup pour remplacer selenium.

Ah bah lol en fait non on peut pas utiliser cette librarie pour faire tourner firefox... J'EN AI MARRE J'AVAIS CHERCHE PRECISEMENT UNE LIB QUI MARCHE AVEC FIREFOX

Et phantomJS non plus ne fonctionne pas avec firefox... J'en ai marre.

Donc je vais plutôt partir sur la librairie GeckoFX qui semble pouvoir contrôler une instance de firefox. Mais j'avais justement pris un putain de projet C# et pas JS pour ne pas me taper ces problèmes de librairies...

Et si cette option ne fonctionne pas mon dernier espoir sera de directement intéragir avec le geckodriver.exe et la ca risque de pas être drôle.

JE NE COMPRENDS RIEN !!!!! Ca n'a aucun sens la doc est inexistante le seul lien qui pourrait amener sur une doc envoie sur la page principale de bitbucket. Tous les exemples de code que je trouve ne fonctionnent pas.

Je n'arrive à rien je commence à devenir fou. Tout ce travail pour rien c'est pas possible.

Même en essayant directement d'intéragir avec le process geckodriver.exe je ne peux pas arriver à mes fins. J'arrive à lancer le service et tout, mais je n'arrive pas à vraiment contrôler ce qu'il se passe donc impossible de venir prendre des screenshot.

Je ne sais tout simplement pas quoi faire ... Je suis bloqué. Je me suis cassé la tête à faire un truc qui marchait bien avec selenium et tout. Mais maintenant plus rien ne fonctionne du jour au lendemain et il n'y a simplement aucune alternative.

Je vais essayer de changer directement le projet Selenium_Clean mais bon ca va pas être drôle.

Ok alors j'ai tout repris depuis le début et je crois que j'ai enfin une solution.

Pour la trouver j'ai re-essayé toutes les techniques que j'avais tenté avant mais dans l'ordre et en les isolant à chaque fois.

Cela inclus :

Tenter de changer la densité de pixels. En effet je me suis dit que comme la résolution était plus basse le soucis était que le virtual screen avait simplement une DPI réduite.

```
profile.SetPreference("layout.css.devPixelsPerPx", "2.0");
```

J'ai aussi tenté de réduire à un seule le nombre de process de Firefox. J'ai pu lire sur internet que parfois cela pouvait influer sur les performances du renderer.

```
profile.SetPreference("dom.ipc.processCount", 1);
```

Ensuite j'ai tenté tout bêtement de rajouter dans la liste des arguments la taille voulue de l'écran.

```
options.AddArgument("--window-size=1920,1080");
```

Mais comme cela ne foncionnait pas, je me suis rabattu sur un script JS pour tenter de forcer la fenêtre à être plus grande.

```
js.ExecuteScript("window.resizeTo(1920, 1080);");
```

Comme cela n'a pas marché j'ai pu lire que cela pouvait être la taille intérieure qui devait être changée

```
js.ExecuteScript("window.innerWidth = 1920; window.innerHeight = 1080;");
```

Encore une fois sans succès. J'ai ensuite tenté d'utiliser trois autres versions du GeckoDriver, 0.27,0.26,0.25 et aucune ne m'aidait.

Mais en fait la seule chose qui a changé quoi que ce soit était la technique suivante :

Changer la window size en utilisant :

```
options.AddArgument("--width=1920");
options.AddArgument("--height=1200");
```

Ca ne marchait pas car j'utilisais une autre methode pour resize en même temps, qui elle ne marchait pas mais qui empêchait celle la de marcher. Ensuite le soucis que j'avais c'est que en mettant 1920-1080 je me retrouvais avec 1920-998 ou un truc du genre ce qui n'était pas normal alors je me disais que cette technique ne marchait pas non plus et je l'ai passée.

Alors tout n'est pas encore gagné, il faut que j'arrive à implémenter ca dans un plus gros projet et que la vidéo puisse être prise seule. Demain je m'occupe de ca.

3.16 Mardi 2 Mai 2023

Bon aujourd'hui je change le programme principal. Le soucis que j'ai c'est que en ajoutant ce système de resize, maintenant la page fait 100x100 et est grise. Il doit y avoir une technique que j'ai oublié de retirer ou un comportement un peu bizarrre.

Bon clairement je ne sais pas QUI DECIDE DE ME POURRIR LA VIE mais il est fort. J'ai télécharger EXACTEMENT les mêmes librairies que sur mon autre projet et j'utilise l'EXACT même geckodriver.exe mais dans le projet principal impossible de lui faire chier une image même avec l'EXACT même code. POURQUOI VOUS ME FAITES CA????=

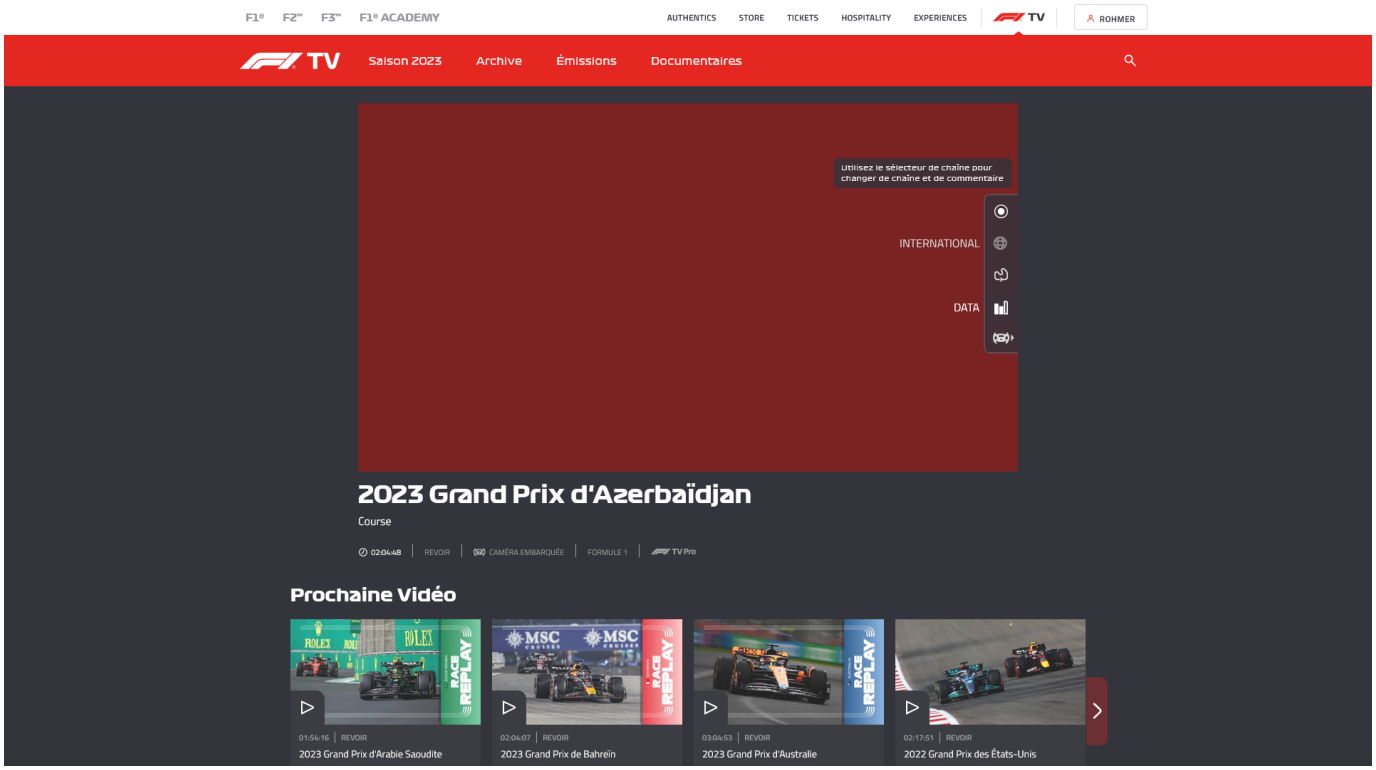
La je ne comprend vraiment pas ce qui peut se passer pour que rien ne fonctionne alors que tout est pareil.

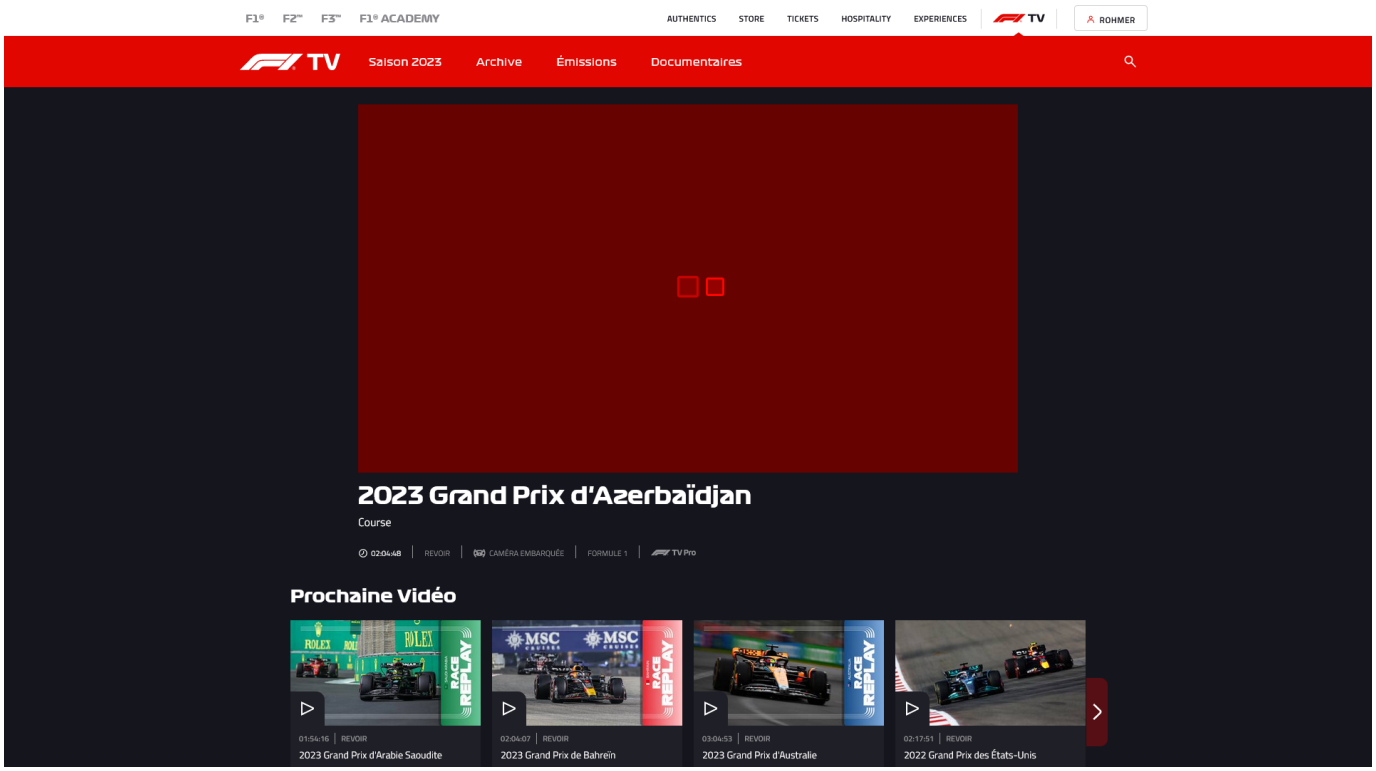
JE VIENS DE TOUT VERIFIER TOUT EST PAREIL JE NE COMPRENDS PAS.

Bon après avoir supprimé l'intégralité de ma classe Emulator cela semble marcher un peu mieux. Je ne vais pas m'étendre sur la castrophe niveau temps que cela représente. Si au moins j'arrive à faire fonctionner quelque chose je suis content.

Maintenant j'ai un soucis un peu spécial. Depuis que j'ai changé la résolution, il semble que le programme aie du mal à cliquer sur l'icone de settings.

En prenant des screenshots du moment ou l'erreur apparait, j'ai pu me rendre compte que en fait le stream est toujours en train de charger et c'est pour ca que on arrive pas à trouver le bouton :





Je pense que je n'ai le soucis que maintenant car le flux en 1080p se lance moins vite. Je vais essayer de voir si je peux detecter un élément d'HTML qui correspond au loading comme ca je peux attendre qu'il disparaisse. Sinon je peux aussi juste essayer de trouver le bouton en boucle pendant une dixaine de secondes.

Bon la j'essaie pendant genre plus de 50 secondes et ca ne marche toujours pas.

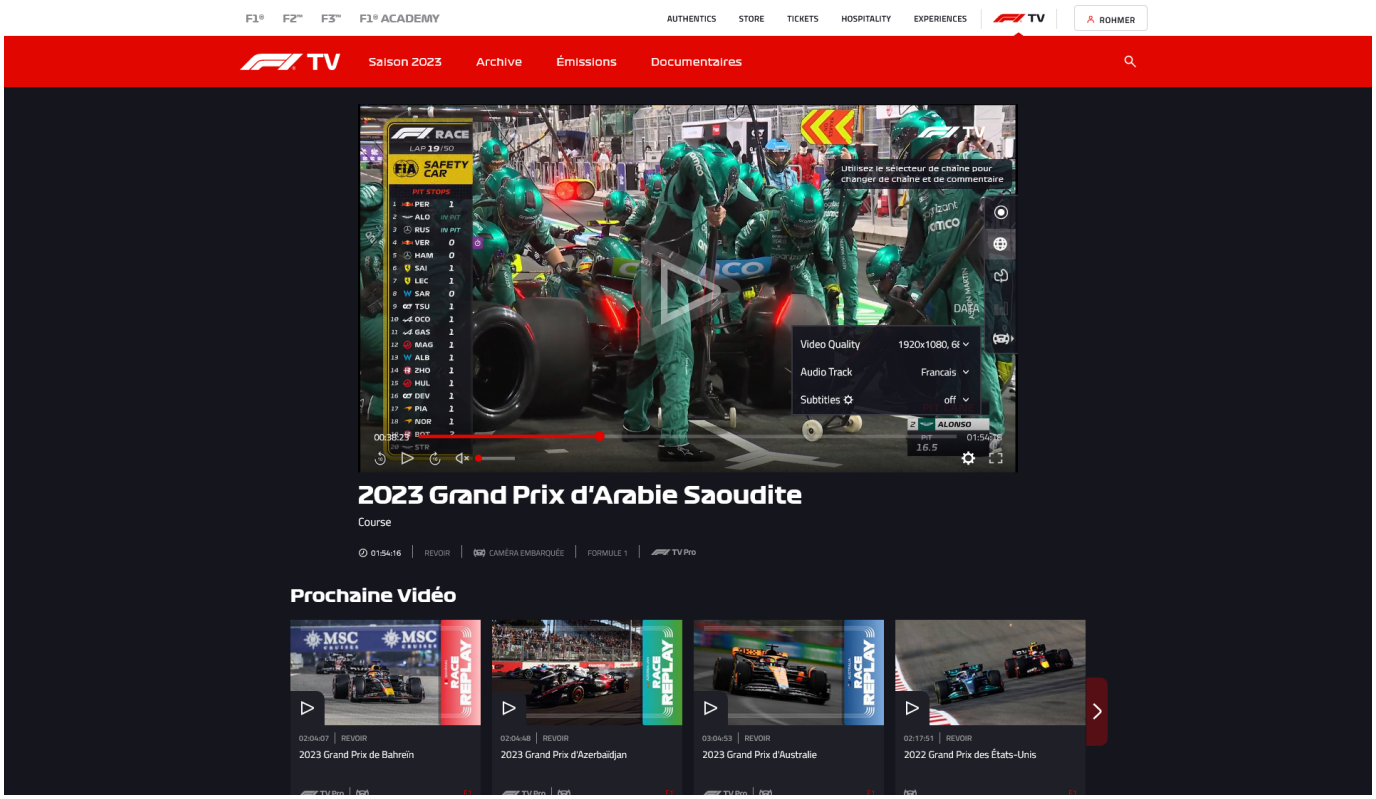
Il semblerait que au final le problème vienne du GP d'azerbidjan.

En effet, quand je teste un autre Grand Prix tout va bien.

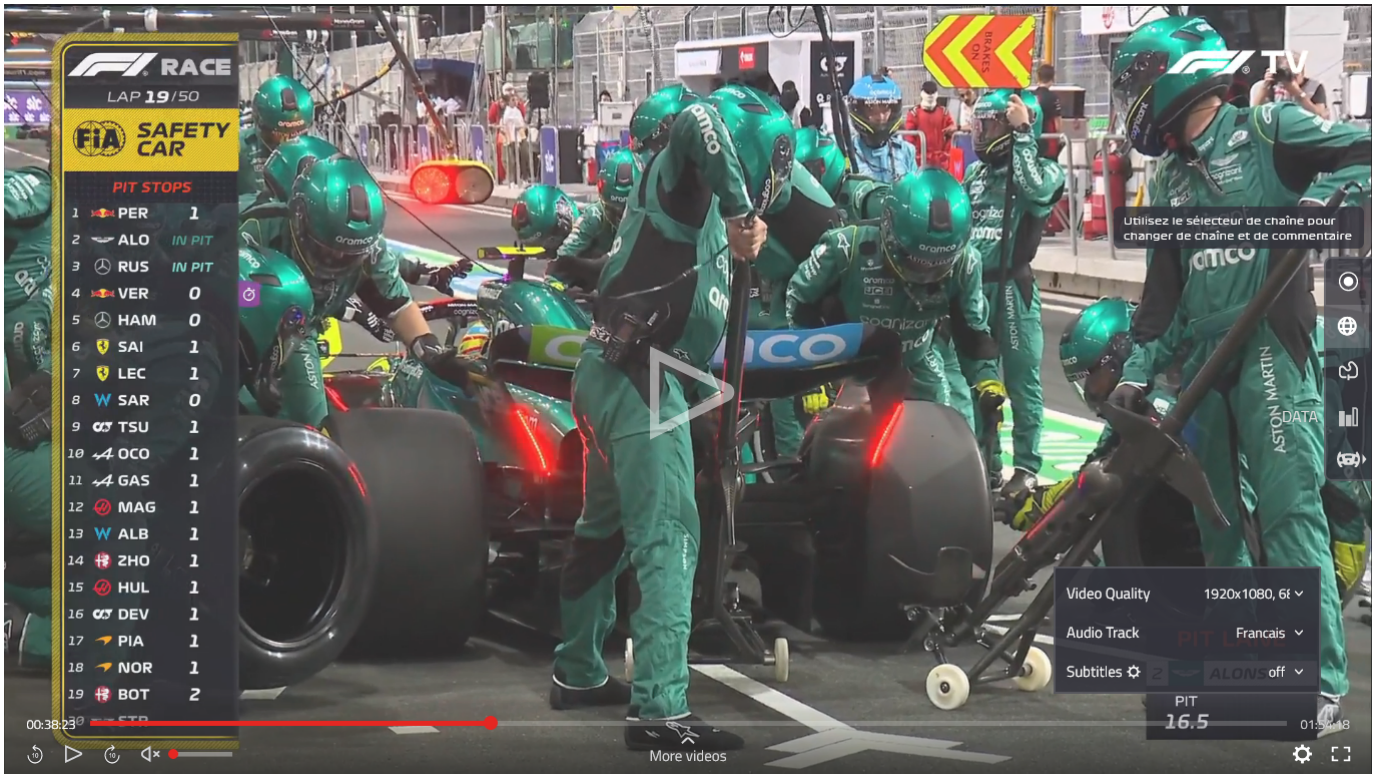
ET MERDE !

J'ai réussi à avoir des images en 1080P mais dés que je passe l'image en plein écran c'est de nouveau du 1366X768

Avant de mettre en plein écran:



Après:



On peut voir sur l'image que l'option 1080P est effectivement bien selectionnée mais il doit y avoir un paramètre de Firefox qui s'occupe de la résolution d'un player vidéo. Il va juste falloir trouver ce paramêtre...

J'ai essayé d'utiliser :

```
Driver.Manage().Window.Size = new System.Drawing.Size(windowWidth, windowHeight);
```

Sans succès.

```
options.AddArgument("--start-maximized");
```

Pareil

```
Driver.Manage().Window.Maximize();
```

Toujours rien

```
profile.SetPreference("full-screen-api.ignore-widgets", true);
```

Nada

```
profile.SetPreference("media.hardware-video-decoding.enabled", true);
```

Toujours pas

J'ai vraiment cru que j'avais trouvé la solution en trouvant cette commande `profile.SetPreference("full-screen-api.enabled", true);` Mais non toujours pas...

Je commence à perdre patience.

C'EST BON.

Après littérallement 3h de debugging avec M.Bonvin (Que je remercie IMMENSEMENT) on a réussi à trouver au fin fond d'un thread github que la valeur était hard codée dans les variables d'environnement et que donc quoi que je fasse je n'aurais pas pu le changer.

En fait la seul moyen de tout règler a été de changer les variables d'environnement de ma machine:

`MOZ_HEADLESS_WIDTH` et `MOZ_HEADLESS_HEIGHT`.

Et ce qu'il y a de bien c'est que maintenant je peux mettre de la 4K et cela permet de faire un meilleur upscaling.

3.17 Recrutement Payerne Mai 2023

J'ai du faire mon recrutement à Payerne Mercredi et Jeudi. Si vous êtes curieux je peux vous dire que comme il n'y avait presque plus de places cet été je ferai Canonnier Lance mines. C'était assez frustrant d'avoir perdu deux jours de travail mais on va faire avec.

3.18 Vendredi 5 Mai 2023

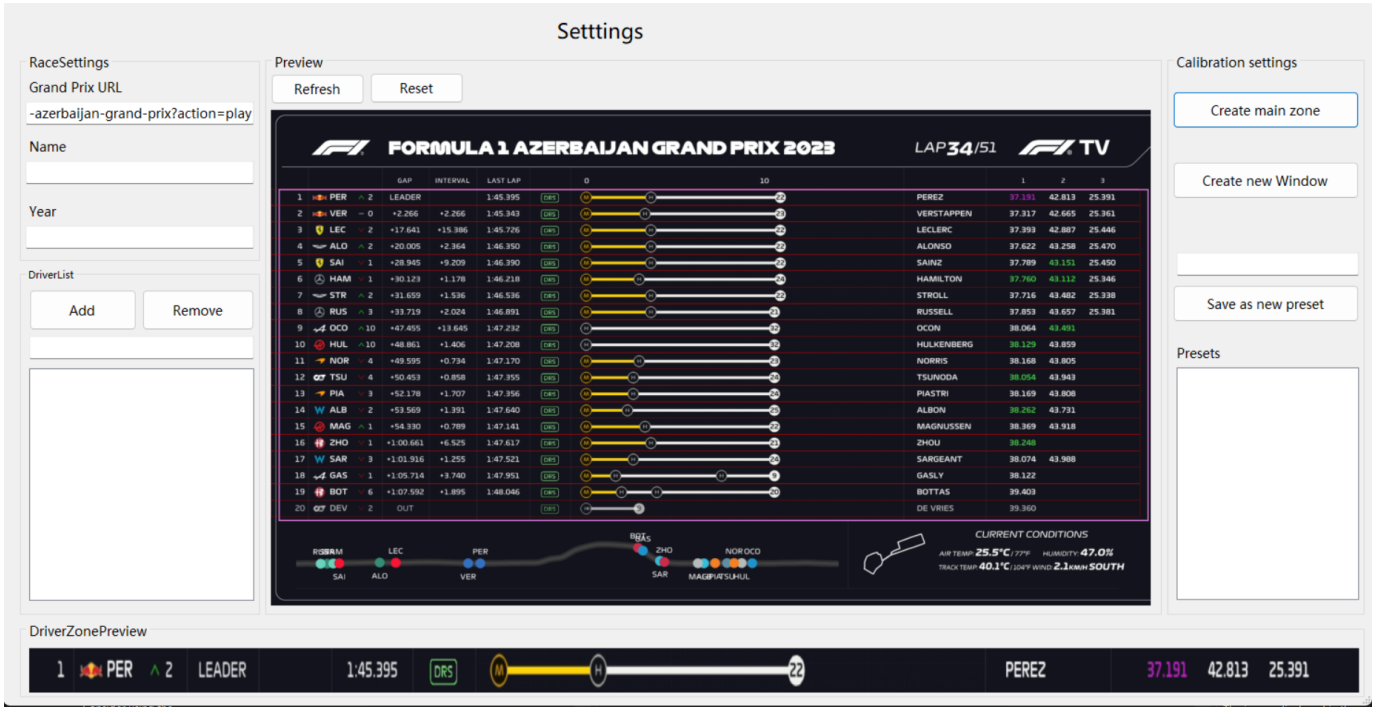
Bon malgrés les courbatures il faut que je me mette au boulot un peu serieusement par ce que sinon ca va être compliqué de rattraper mon retard.

La dernière fois si je me souviens bien j'avais réussi à trouver un moyen de prendres des images en bonne résolution. Il faut maintenant que je commence à faire fonctionner la calibration et ce qui serait bien ca serait que je commence à ajouter la partie OCR au projet. Il faut que je me dépêche car Lundi je dois m'occuper du Poster.

OK j'ai compris le soucis que j'avais quand j'essayais de faire la calibration. J'avais mis l'image en ZOOM ce qui fait que si la hauteur n'était pas la bonne, l'image était recentrée ce qui fait que cela faussait totalement les résultats.

Quand on fait en sorte que l'image prenne toute la place, les coordonnées sont prises correctement.

Voici un exemple d'ou en est la partie calibration.



Normalement il me suffit d'implémenter les windows, et on devrait relativement facilement ajouter les pilotes.

Et voila. J'ai pu implémenter les windows et les pilotes. Et je peux aussi exporter des presets et les loader. Bon le loading est un peu beuggé au niveau de l'affichage mais il semble qu'il fonctionne bien quand je save les images.

Lundi je m'occupe du poster etc.. mais je pense que la suite va être l'implémentation de l'OCR.

3.19 Lundi 8 Mai 2023

Aujourd'hui c'est journée Poster.

Je pense que je ne vais pas finir la journée content car les limitations sont un peu trop présentes.

J'ai fait une version que Garcia pourrait accepter, c'est à dire en noir et blanc et avec un tout petit peu plus de détail.

Track Trends

Maxime Rohmer

Diploma work Tech II 2023

Tesseract OCR | CSharp | Selenium | Python | Headless Firefox

Le truc c'est que en blanc je trouve que ca ne marche pas super. Et le concept d'avoir trois parties au projet qui se posent autour d'un circuit c'est peut-être pas la meilleure idée.

Je me suis dit que la bonne idée serait peut-être de prendre un autre circuit pour qu'il y aie bien trois parties :

Track Trends

+37.029

+37.029

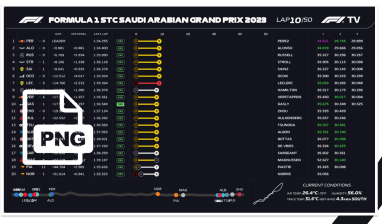
+37.029

+37.029

1 Emulate Recover

2 Detect Decode Store

+37.029	+9.787
+48.419	+15.501
+53.609	+5.190



3 Interpret Display

1 + 0.000

2 + 0.895 DRS

3 + 3.568 DRS

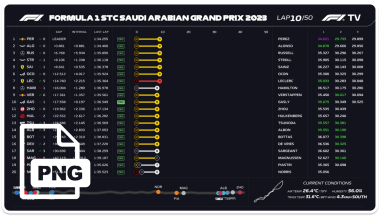
Maxime Rohmer

Diploma work Tech II 2023

Tesseract OCR · CSharp · Selenium · Python · Headless Firefox

Clairement ce poster doit faire partie des pires. C'est pas clair et ca part dans tous les sens. Je vais essayer avec un autre layout de circuit.

Track Trends



1 Emulate Recover

2 Detect Decode Store

+37.029 +37.029 +37.029 +37.029

+37.029	+9.787
+48.419	+15.501
+53.609	+5.190

```
{
    "Name": "Verstappen",
    "LapTime": "1:32.923",
    "Sector 1": "37.029",
    "Tyre": "Soft",
    "DRS": True,
}
```

3 Interpret Display

1 + 0.000
2 + 0.895 DRS
3 + 3.568 DRS

Maxime Rohmer

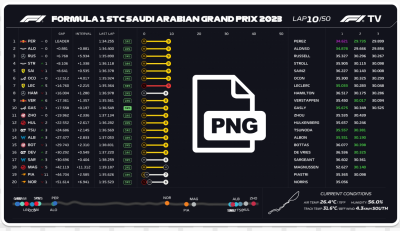
Diploma work Tech II 2023

Tesseract OCR · CSharp · Selenium · Python · Headless Firefox

Je me suis ensuite dit que le circuit n'était peut être tout simplement pas une bonne idée. J'ai donc essayé de faire quelque chose de plus classique avec juste un peu de background pour qu'on puisse éviter le soucis de la page blanche derrière :

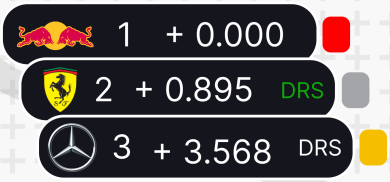
Track Trends

1 Emulate Recover



2 Detect Decode Store

3 Interpret Display



Maxime Rohmer

Diploma work Tech II 2023

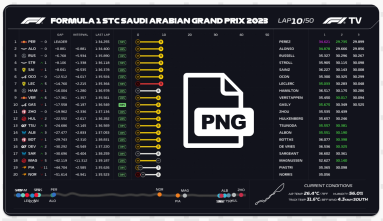
Tesseract OCR · CSharp · Selenium · Python · Headless Firefox

Puis je me suis dit que finalement le circuit me manquait. Alors j'ai décidé de combiner le background et le circuit ainsi que simplifier légèrement les diagrammes en retouchant un peu tout le reste on pouvait arriver à quelque chose de sympatique :

Track Trends

"NOOOOOOO!!"
Charles Leclerc (2022)

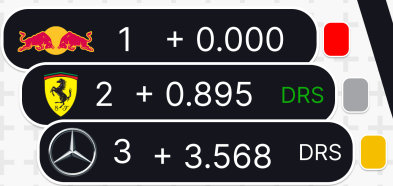
"Ok ..."
Kimi Raikonnen (2013)



1
Emulate
Recover

2
Detect
Decode
Store

3
Interpret
Display



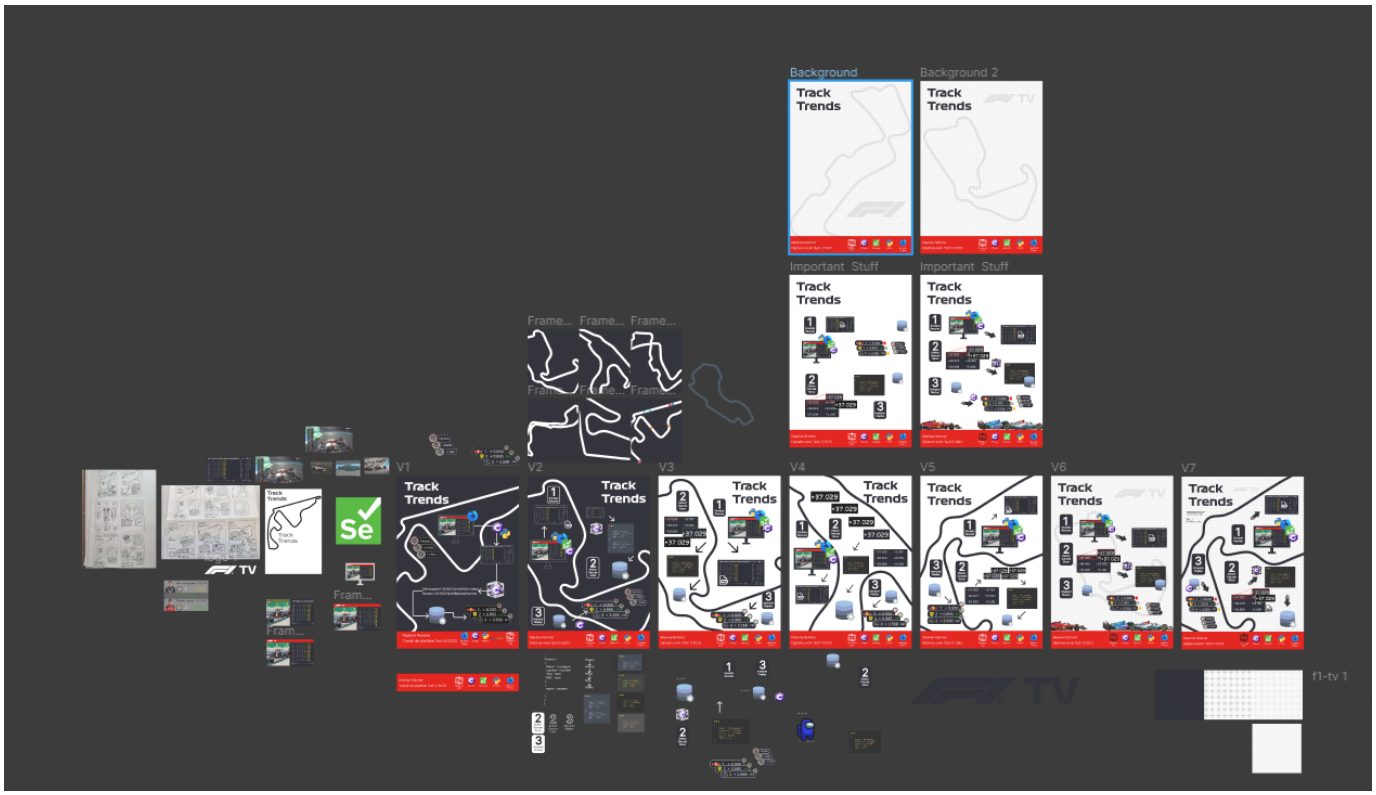
Maxime Rohmer

Diploma work Tech II 2023

Tesseract OCR · CSharp · Selenium · Python · Headless Firefox

Je ne suis pas content à 100% mais bon je pense que je vais m'en satisfaire.

Pour donner une idée de la galère que c'est de créer un poster, voici ce à quoi ressemble mon espace de travail Figma :



Je ne suis pas un graphiste et ca se voit '^^.

Je pense que comme il me reste un peu de temps aujourd'hui, je vais faire un peu de documentation de la partie récupèration d'images. En effet, je pense que je n'aurai plus besoin de changer grand chose à ce niveau. Mais je ne ferai pas la partie analyse fonctionnelle car l'interface n'est clairement pas terminée.

En fait j'avais oublié mais j'ai eu un rendez vous médical du coup je n'ai pas eu trop le temps de faire la doc que je voulais. Mais au moins je pense avoir finit mon travail sur le poster et le abstract en Anglais qui sont les deux gros livrables à venir.

3.20 Mardi 9 Mai 2023

Bon je viens de me rendre compte que apparemment on doit rendre l'abstract anglais, le Poster, ET LE PROJET. Je pense que mes deux jours à l'armée m'ont fait perdre un peu la notion du temps car j'avais l'impression que l'evaluation intermédiaire 1 était il y a genre moins d'une semaine.

Donc aujourd'hui je ne vais pas trop avancer sur le code et vraiment me focus sur la documentation de la récupèration d'images. Je pense que je vais aussi ajouter la partie calibration à la documentation. Je pense que c'est important que je prenne le temps maintenant car sinon le prof aura l'impression que ca n'a pas trop avancé depuis la dernière fois.

Et puis je pense que la partie calibration et récupèration d'images ne va pas trop changer et la partie calibration encore moins.

La partie anglaise je fais la revoir un peu mais je l'avais déja faite pendant les premiers jours alors ca devrait aller.

4. Code

4.1 ConfigurationTool.cs

```
/// Author : Maxime Rohmer
/// Date : 08/05/2023
/// File : ConfigurationTool.cs
/// Brief : Class that contains all the methods needed to create a config file for the OCR
/// Version : 0.1

using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Tesseract;
using System.IO;

namespace Test_Merge
{
    public class ConfigurationTool
    {
        public Zone MainZone;
        public const int NUMBER_OF_DRIVERS = 20;
        public const int NUMBER_OF_ZONES = 9;
        public const string CONFIGS_FOLDER_NAME = "./Presets/";

        public ConfigurationTool(Bitmap fullImage, Rectangle mainZoneDimensions)
        {
            MainZone = new Zone(fullImage, mainZoneDimensions,"Main");
            AutoCalibrate();
        }
        public void ResetMainZone()
        {
            MainZone.ResetZones();
        }
        public void ResetWindows()
        {
            MainZone.ResetWindows();
        }
        public void SaveToJson(List<string> drivers, string configName)
        {
            string JSON = "";

            JSON += "{" + Environment.NewLine;
            JSON += MainZone.ToJSON() + "," + Environment.NewLine;
            JSON += "\"Drivers\":[" + Environment.NewLine;

            for (int i = 0; i < drivers.Count; i++)
            {
                JSON += "\"" + drivers[i] + "\"";
                if (i < drivers.Count - 1)
                    JSON += ",";
                JSON += Environment.NewLine;
            }

            JSON += "]" + Environment.NewLine;

            JSON += "}";

            if (!Directory.Exists(CONFIGS_FOLDER_NAME))
                Directory.CreateDirectory(CONFIGS_FOLDER_NAME);

            string path = CONFIGS_FOLDER_NAME + configName;

            if (File.Exists(path + ".json"))
            {
                //We need to create a new name
                int count = 2;
                while (File.Exists(path + "_" + count + ".json"))
                {
                    count++;
                }
                path += "_" + count + ".json";
            }
            else
            {
                path += ".json";
            }

            File.WriteAllText(path, JSON);
        }
        public void AddWindows(List<Rectangle> rectangles)
        {
            foreach (Zone driverZone in MainZone.Zones)
            {
```

```
            Bitmap zoneImage = driverZone.ZoneImage;

            for (int i = 1; i <= rectangles.Count; i++)
            {
                switch (i)
                {
                    case 1:
                        //First zone should be the driver's Position
                        driverZone.AddWindow(new DriverPositionWindow(driverZone.ZoneImage, rectangles[i - 1], false));
                        break;
                    case 2:
                        //First zone should be the Gap to leader
                        driverZone.AddWindow(new DriverGapToLeaderWindow(driverZone.ZoneImage, rectangles[i - 1], false));
                        break;
                    case 3:
                        //First zone should be the driver's Lap Time
                        driverZone.AddWindow(new DriverLapTimeWindow(driverZone.ZoneImage, rectangles[i - 1], false));
                        break;
                    case 4:
                        //First zone should be the driver's DRS status
                        driverZone.AddWindow(new DriverDrsWindow(driverZone.ZoneImage, rectangles[i - 1], false));
                        break;
                    case 5:
                        //First zone should be the driver's Tyre's informations
                        driverZone.AddWindow(new DriverTyresWindow(driverZone.ZoneImage, rectangles[i - 1], false));
                        break;
                    case 6:
                        //First zone should be the driver's Name
                        driverZone.AddWindow(new DriverNameWindow(driverZone.ZoneImage, rectangles[i - 1], false));
                        break;
                    case 7:
                        //First zone should be the driver's First Sector
                        driverZone.AddWindow(new DriverSectorWindow(driverZone.ZoneImage, rectangles[i - 1], 1, false));
                        break;
                    case 8:
                        //First zone should be the driver's Second Sector
                        driverZone.AddWindow(new DriverSectorWindow(driverZone.ZoneImage, rectangles[i - 1], 2, false));
                        break;
                    case 9:
                        //First zone should be the driver's Position Sector
                        driverZone.AddWindow(new DriverSectorWindow(driverZone.ZoneImage, rectangles[i - 1], 3, false));
                        break;
                }
            }
        }
    }
    public void AutoCalibrate()
    {
        List<Rectangle> detectedText = new List<Rectangle>();
        List<Zone> zones = new List<Zone>();

        TesseractEngine engine = new TesseractEngine(Window.TESS_DATA_FOLDER.FullName, "eng", EngineMode.Default);
        Image image = MainZone.ZoneImage;
        var tessImage = Pix.LoadFromMemory(Window.ImageToByte(image));

        Page page = engine.Process(tessImage);
        using (var iter = page.GetIterator())
        {
            iter.Begin();
            do
            {
                Rect boundingBox;
                if (iter.TryGetBoundingBox(PageIteratorLevel.Word, out boundingBox))
                {
                    //var text = iter.GetText(PageIteratorLevel.Word).ToUpper();
                    //We remove all the rectangles that are definitely too big
                    if (boundingBox.Height < image.Height / NUMBER_OF_DRIVERS)
                    {
                        //Now we add a filter to only get the boxes in the right because they are much more reliable in size
                        if (boundingBox.X1 > image.Width / 2)
                        {
                            //Now we check if an other square box has been found roughly in the same y axis
                            bool match = false;
                            //The tolerance is roughly half the size that a window will be
                            int tolerance = (image.Height / NUMBER_OF_DRIVERS) / 2;

                            foreach (Rectangle rect in detectedText)
                            {
                                if (rect.Y > boundingBox.Y1 - tolerance && rect.Y < boundingBox.Y1 + tolerance)
                                {
                                    //There already is a rectangle in this line
                                    match = true;
                                }
                            }
                            //if nothing matched we can add it
                            if (!match)
                                detectedText.Add(new Rectangle(boundingBox.X1, boundingBox.Y1, boundingBox.Width, boundingBox.Height));
                        }
                    }
                }
            } while (iter.Next(PageIteratorLevel.Word));
        }
        //DEBUG
        int i = 1;
```

```
            foreach (Rectangle Rectangle in detectedText)
            {
                Rectangle windowRectangle;
                Size windowSize = new Size(image.Width, image.Height / NUMBER_OF_DRIVERS);
                Point windowLocation = new Point(0, (Rectangle.Y + Rectangle.Height / 2) - windowSize.Height / 2);
                windowRectangle = new Rectangle(windowLocation, windowSize);
                //We add the driver zones
                Zone driverZone = new Zone(MainZone.ZoneImage, windowRectangle, "DriverZone");
                MainZone.AddZone(driverZone);

                driverZone.ZoneImage.Save("Driver" + i+".png");
                i++;
            }
        }
    }
}
```

4.2 DriverGapToLeaderWindow.cs

```
/// Author : Maxime Rohmer
/// Date : 08/05/2023
/// File : DriverGapToLeaderWindow.cs
/// Brief : Window containing infos about the gap to the leader of a driver
/// Version : 0.1

using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Test_Merge
{
    internal class DriverGapToLeaderWindow:Window
    {
        public DriverGapToLeaderWindow(Bitmap image, Rectangle bounds, bool generateEngine = true) : base(image, bounds,generateEngine)
        {
            Name = "GapToLeader";
        }
        /// <summary>
        /// Decodes the gap to leader using Tesseract OCR
        /// </summary>
        /// <returns></returns>
        public override async Task<object> DecodePng()
        {
            int result = await GetTimeFromPng(WindowImage, OcrImage.WindowType.Gap, Engine);
            return result;
        }
    }
}
```

4.3 DriverPositionWindow.cs

```
/// Author : Maxime Rohmer
/// Date : 08/05/2023
/// File : DriverPosition.cs
/// Brief : Window containing infos about the position of a driver.
/// Version : 0.1

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Drawing;

namespace Test_Merge
{
    public class DriverPositionWindow:Window
    {
        public DriverPositionWindow(Bitmap image, Rectangle bounds, bool generateEngine = true) : base(image, bounds,generateEngine)
        {
            Name = "Position";
        }
        /// <summary>
        /// Decodes the position number using Tesseract OCR
        /// </summary>
        /// <returns>The position of the pilot in int</returns>
        public override async Task<object> DecodePng()
        {
            string ocrResult = await GetStringFromPng(WindowImage, Engine, "0123456789");

            int position;
            try
            {
                position = Convert.ToInt32(ocrResult);
            }
            catch
            {
                position = -1;
            }
            return position;
        }
    }
}
```

4.4 F1TVEmulator.cs

```
/// Author : Maxime Rohmer
/// Date : 08/05/2023
/// File : F1TVEmulator.cs
/// Brief : Class that contains methods to emulate a browser and navigate the F1TV website
/// Version : 0.1

using OpenQA.Selenium;
using OpenQA.Selenium.Firefox;
using OpenQA.Selenium.Interactions;
using OpenQA.Selenium.Support.UI;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

namespace Test_Merge
{
    internal class F1TVEmulator
    {
        public const string COOKIE_HOST = ".formula1.com";
        public const string PYTHON_COOKIE_RETRIEVAL_FILENAME = "recoverCookiesCSV.py";
        public const string GECKODRIVER_FILENAME = @"geckodriver-v0.27.0-win64\geckodriver.exe";
        //BE CAREFULL IF YOU CHANGE IT HERE YOU NEED TO CHANGE IT IN THE PYTHON SCRIPT TOO
        public const string COOKIES_CSV_FILENAME = "cookies.csv";

        private FirefoxDriver Driver;

        private bool _ready;
        private string _grandPrixUrl;
        public string GrandPrixUrl { get => _grandPrixUrl; private set => _grandPrixUrl = value; }
        public bool Ready { get => _ready; set => _ready = value; }
        public F1TVEmulator(string grandPrixUrl)
        {
            GrandPrixUrl = grandPrixUrl;
            Ready = false;
        }
        private void StartCookieRecovering()
        {
            string scriptPath = PYTHON_COOKIE_RETRIEVAL_FILENAME;
            Process process = new Process();
            process.StartInfo.FileName = "python.exe";
            process.StartInfo.Arguments = scriptPath;
            process.StartInfo.UseShellExecute = false;
            process.StartInfo.RedirectStandardOutput = true;
            process.Start();
            string output = process.StandardOutput.ReadToEnd();
            process.WaitForExit();
        }
        public string GetCookie(string host, string name)
        {
            StartCookieRecovering();
            string value = "";
            List<Cookie> cookies = new List<Cookie>();
            using (var reader = new StreamReader(COOKIES_CSV_FILENAME))
            {
                // Read the header row and validate column order
                string header = reader.ReadLine();
                string[] expectedColumns = { "host_key", "name", "value", "path", "expires_utc", "is_secure", "is_httponly" };
                string[] actualColumns = header.Split(',');
                for (int i = 0; i < expectedColumns.Length; i++)
                {
                    if (expectedColumns[i] != actualColumns[i])
                    {
                        throw new InvalidOperationException($"Expected column '{expectedColumns[i]}' at index {i} but found '{actualColumns[i]}'");
                    }
                }

                // Read each data row and parse values into a Cookie object
                while (!reader.EndOfStream)
                {
                    string line = reader.ReadLine();
                    string[] fields = line.Split(',');

                    string hostname = fields[0];
                    string cookieName = fields[1];

                    if (hostname == host && cookieName == name)
                    {
                        value = fields[2];
                    }
                }
            }
```

```
            return value;
        }
        public async Task<int> Start()
        {
            Ready = false;

            string loginCookieName = "login";
            string loginSessionCookieName = "login-session";
            string loginCookieValue = GetCookie(COOKIE_HOST, loginCookieName);
            string loginSessionValue = GetCookie(COOKIE_HOST, loginSessionCookieName);

            int windowWidth = 1920;
            int windowHeight = 768;

            var service = FirefoxDriverService.CreateDefaultService(GECKODRIVER_FILENAME);
            service.Host = "127.0.0.1";
            service.Port = 5555;

            FirefoxProfile profile = new FirefoxProfile();
            FirefoxOptions options = new FirefoxOptions();
            //profile.SetPreference("full-screen-api.ignore-widgets", true);
            //profile.SetPreference("media.hardware-video-decoding.enabled", true);
            //profile.SetPreference("full-screen-api.enabled", true);
            options.Profile = profile;
            profile.SetPreference("layout.css.devPixelsPerPx", "1.0");

            options.AcceptInsecureCertificates = true;
            options.AddArgument("--headless");
            //options.AddArgument("--start-maximized");
            //options.AddArgument("--window-size=1920x1080");
            //options.AddArgument("--width=" + windowWidth);
            //options.AddArgument("--height=" + windowHeight);
            //options.AddArgument("-window-size=1920x1080");
            //options.AddArgument("--width=1920");
            //options.AddArgument("--height=1080");
            //profile

            try
            {
                Driver = new FirefoxDriver(service, options);
            }
            catch
            {
                Ready = false;
                return 101;
            }

            Actions actions = new Actions(Driver);
            var loginCookie = new Cookie(loginCookieName, loginCookieValue, COOKIE_HOST, "/", DateTime.Now.AddDays(5));
            var loginSessionCookie = new Cookie(loginSessionCookieName, loginSessionValue, COOKIE_HOST, "/", DateTime.Now.AddDays(5));

            Driver.Navigate().GoToUrl("https://f1tv.formula1.com/");

            Driver.Manage().Cookies.AddCookie(loginCookie);
            Driver.Manage().Cookies.AddCookie(loginSessionCookie);

            try
            {
                Driver.Navigate().GoToUrl(GrandPrixUrl);
            }
            catch
            {
                //The url is not a valid url
                Driver.Dispose();
                return 103;
            }

            //Waits for the page to fully load
            Driver.Manage().Timeouts().PageLoad = TimeSpan.FromSeconds(30);

            //Removes the cookie prompt
            try
            {
                IWebElement conscentButton = Driver.FindElement(By.Id("truste-consent-button"));
                conscentButton.Click();
            }
            catch
            {
                //Could not locate the cookie button
                Screenshot("ERROR104");
                Driver.Dispose();
                return 104;
            }

            //Again waits for the page to fully load (when you accept cookies it takes a little time for the page to load)
            //Cannot use The timeout because the feed loading is not really loading so there is not event or anything
            Thread.Sleep(5000);

            //Switches to the Data channel
            try
            {
                IWebElement dataChannelButton = Driver.FindElement(By.ClassName("data-button"));
                dataChannelButton.Click();
            }
```

```
            catch
            {
                //If the data button does not exists its because the user is not connected
                Screenshot("ERROR102");
                Driver.Dispose();
                return 102;
            }

            //Open settings
            // Press the space key, this should make the setting button visible
            // It does not matter if the feed is paused because when changing channel it autoplays
            actions.SendKeys(OpenQA.Selenium.Keys.Space).Perform();
            //Clicks on the settings Icon

            int tries = 0;
            bool success = false;
            while (tries < 100 && !success)
            {
                Thread.Sleep(100);
                try
                {
                    IWebElement settingsButton = Driver.FindElement(By.ClassName("bmpui-ui-settingstogglebutton"));
                    settingsButton.Click();
                    IWebElement selectElement = Driver.FindElement(By.ClassName("bmpui-ui-videoqualityselectbox"));
                    SelectElement select = new SelectElement(selectElement);
                    IWebElement selectOption = selectElement.FindElement(By.CssSelector("option[value^='1080_']"));
                    selectOption.Click();
                    success = true;
                }
                catch
                {
                    //Sometimes it can crash because it could not get the options to show up in time. When it happens just retry
                    success = false;
                    tries++;
                }
            }

            if (!success)
            {
                Screenshot("ERROR105");
                Driver.Dispose();
                return 105;
            }

            Screenshot("BEFOREFULLSCREEN");

            //Makes the feed fullscreen
            //Driver.Manage().Window.Size = new System.Drawing.Size(windowWidth, windowHeight);
            Driver.Manage().Window.Maximize();
            WebDriverWait wait = new WebDriverWait(Driver, TimeSpan.FromSeconds(10));
            try
            {
                IWebElement fullScreenButton = Driver.FindElement(By.ClassName("bmpui-ui-fullscreentogglebutton"));
                fullScreenButton.Click();
            }
            catch
            {
                Screenshot("ERROR106");
                Driver.Dispose();
                return 106;
            }

            Screenshot("AFTERFULLSCREEN");

            //STARTUP FINISHED READY TO SCREENSHOT
            Ready = true;
            return 0;
        }
        public Bitmap Screenshot(string name = "TEST")
        {
            Bitmap result = new Bitmap(4242, 6969);
            try
            {
                //Screenshot scrsht = ((ITakesScreenshot)Driver).GetScreenshot();
                //profileriver.SetPreference("layout.css.devPixelsPerPx", "1.0");

                //Screenshot scrsht = Driver.GetFullPageScreenshot();
                Screenshot scrsht = Driver.GetScreenshot();


                byte[] screenshotBytes = Convert.FromBase64String(scrsht.AsBase64EncodedString);
                MemoryStream stream = new MemoryStream(screenshotBytes);

                result = new Bitmap(stream);
                //result.Save(name + ".png");
                scrsht.SaveAsFile(name + ".png");
            }
            catch
            {
                //Nothing for now
            }
            return result;
        }
        public void Stop()
```

```
        {
            Ready = false;
            Driver.Dispose();
        }
        public void ResetDriver()
        {
            Ready = false;
            Driver.Dispose();
            Driver = null;
        }
    }
}
```

4.5 Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Test_Merge
{
    internal static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

4.6 Window.cs

```
/// Author : Maxime Rohmer
/// Date : 08/05/2023
/// File : Window.cs
/// Brief : Default Window object that is mainly expected to be inherited.
/// Version : 0.1

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Drawing;
using System.IO;
using Tesseract;
using System.Text.RegularExpressions;
using System.Drawing.Drawing2D;

namespace Test_Merge
{
    public class Window
    {
        private Rectangle _bounds;
        private Bitmap _image;
        private string _name;
        protected TesseractEngine Engine;
        public Rectangle Bounds { get => _bounds; private set => _bounds = value; }
        public Bitmap Image { get => _image; set => _image = value; }
        public string Name { get => _name; protected set => _name = value; }
        //This will have to be changed if you want to make it run on your machine
        public static DirectoryInfo TESS_DATA_FOLDER = new DirectoryInfo(@"C:\Users\Moi\Pictures\SeleniumScreens\TessData");

        public Bitmap WindowImage
        {
            get
            {
                //This little trickery lets you have the image that the window sees
                Bitmap sample = new Bitmap(Bounds.Width, Bounds.Height);
                Graphics g = Graphics.FromImage(sample);
                g.DrawImage(Image, new Rectangle(0, 0, sample.Width, sample.Height), Bounds, GraphicsUnit.Pixel);
                return sample;
            }
        }
        public Window(Bitmap image, Rectangle bounds, bool generateEngine = true)
        {
            Image = image;
            Bounds = bounds;
            if (generateEngine)
            {
                Engine = new TesseractEngine(TESS_DATA_FOLDER.FullName, "eng", EngineMode.Default);
                Engine.DefaultPageSegMode = PageSegMode.SingleLine;
            }
        }
        /// <summary>
        /// Method that will have to be used by the childrens to let the model make them decode the images they have
        /// </summary>
        /// <returns>Returns an object because we dont know what kind of return it will be</returns>
        public virtual async Task<Object> DecodePng()
        {
            return "NaN";
        }
        /// <summary>
        /// Method that will have to be used by the childrens to let the model make them decode the images they have
        /// </summary>
        /// <param name="driverList">This is a list of the different possible drivers in the race. It should not be too big but NEVER be too short</param>
        /// <returns>Returns an object because we dont know what kind of return it will be</returns>
        public virtual async Task<Object> DecodePng(List<string> driverList)
        {
            return "NaN";
        }
        /// <summary>
        /// This converts an image into a byte[]. It can be usefull when doing unsafe stuff. Use at your own risks
        /// </summary>
        /// <param name="inputImage">The image you want to convert</param>
        /// <returns>A byte array containing the image informations</returns>
        public static byte[] ImageToByte(Image inputImage)
        {
            using (var stream = new MemoryStream())
            {
                inputImage.Save(stream, System.Drawing.Imaging.ImageFormat.Png);
                return stream.ToArray();
            }
        }
        /// <summary>
        /// This method is used to recover a time from a PNG using Tesseract OCR
        /// </summary>
        /// <param name="windowImage">The image where the text is</param>
        /// <param name="windowType">The type of window it is</param>
        /// <param name="Engine">The Tesseract Engine</param>
```

```
        /// <returns>The time in milliseconds</returns>
        public static async Task<int> GetTimeFromPng(Bitmap windowImage, OcrImage.WindowType windowType, TesseractEngine Engine)
        {
            //Kind of a big method but it has a lot of error handling and has to work with three special cases
            string rawResult = "";
            int result = 0;

            switch (windowType)
            {
                case OcrImage.WindowType.Sector:
                    //The usual sector is in this form : 33.456
                    Engine.SetVariable("tessedit_char_whitelist", "0123456789.");
                    break;
                case OcrImage.WindowType.LapTime:
                    //The usual Lap time is in this form : 1:45:345
                    Engine.SetVariable("tessedit_char_whitelist", "0123456789.:");
                    break;
                case OcrImage.WindowType.Gap:
                    //The usual Gap is in this form : + 34.567
                    Engine.SetVariable("tessedit_char_whitelist", "0123456789.+");
                    break;
                default:
                    Engine.SetVariable("tessedit_char_whitelist", "");
                    break;
            }


            Bitmap enhancedImage = new OcrImage(windowImage).Enhance(windowType);

            var tessImage = Pix.LoadFromMemory(ImageToByte(enhancedImage));

            Page page = Engine.Process(tessImage);
            Graphics g = Graphics.FromImage(enhancedImage);
            // Get the iterator for the page layout
            using (var iter = page.GetIterator())
            {
                // Loop over the elements of the page layout
                iter.Begin();
                do
                {
                    // Get the text for the current element
                    try
                    {
                        rawResult += iter.GetText(PageIteratorLevel.Word);
                    }
                    catch
                    {
                        //nothing we just dont add it if its not a number
                    }
                } while (iter.Next(PageIteratorLevel.Word));
            }

            List<string> rawNumbers;

            //In the gaps we can find '+' but we dont care about it its redondant a driver will never be - something
            if (windowType == OcrImage.WindowType.Gap)
                rawResult = Regex.Replace(rawResult, "[^0-9.:]", "");

            //Splits into minuts seconds miliseconds
            rawNumbers = rawResult.Split('.', ':').ToList<string>();
            //removes any empty cells (tho this usually sign of a really bad OCR implementation tbh will have to be fixed higher in the chian)
            rawNumbers.RemoveAll(x => ((string)x) == "");

            if (rawNumbers.Count == 3)
            {
                //mm:ss:ms
                result = (Convert.ToInt32(rawNumbers[0]) * 1000 * 60) + (Convert.ToInt32(rawNumbers[1]) * 1000) + Convert.ToInt32(rawNumbers[2]);
            }
            else
            {
                if (rawNumbers.Count == 2)
                {
                    //ss:ms
                    result = (Convert.ToInt32(rawNumbers[0]) * 1000) + Convert.ToInt32(rawNumbers[1]);

                    if (result > 999999)
                    {
                        //We know that we have way too much seconds to make a minut
                        //Its usually because the ":" have been interpreted as a number
                        int minuts = (int)(rawNumbers[0][0] - '0');
                        // rawNumbers[0][1] should contain the : that has been mistaken
                        int seconds = Convert.ToInt32(rawNumbers[0][2].ToString() + rawNumbers[0][3].ToString());
                        int ms = Convert.ToInt32(rawNumbers[1]);
                        result = (Convert.ToInt32(minuts) * 1000 * 60) + (Convert.ToInt32(seconds) * 1000) + Convert.ToInt32(ms);
                    }
                }
                else
                {
                    if (rawNumbers.Count == 1)
                    {
                        try
                        {
                            result = Convert.ToInt32(rawNumbers[0]);
                        }
```

```
                        catch
                        {
                            //It can be because the input is empty or because its the LEADER bracket
                            result = 0;
                        }
                    }
                    else
                    {
                        //Auuuugh
                        result = 0;
                    }
                }
            }
            page.Dispose();
            return result;
        }
        /// <summary>
        /// Method that recovers strings from an image using Tesseract OCR
        /// </summary>
        /// <param name="WindowImage">The image of the window that contains text</param>
        /// <param name="Engine">The Tesseract engine</param>
        /// <param name="allowedChars">The list of allowed chars</param>
        /// <param name="windowType">The type of window the text is on. Depending on the context the OCR will behave differently</param>
        /// <returns>the string it found</returns>
        public static async Task<string> GetStringFromPng(Bitmap WindowImage, TesseractEngine Engine, string allowedChars = "", OcrImage.WindowType windowType
= OcrImage.WindowType.Text)
        {
            string result = "";

            Engine.SetVariable("tessedit_char_whitelist", allowedChars);

            Bitmap rawData = WindowImage;
            Bitmap enhancedImage = new OcrImage(rawData).Enhance(windowType);

            Page page = Engine.Process(enhancedImage);
            using (var iter = page.GetIterator())
            {
                iter.Begin();
                do
                {
                    result += iter.GetText(PageIteratorLevel.Word);
                } while (iter.Next(PageIteratorLevel.Word));
            }
            page.Dispose();
            return result;
        }
        /// <summary>
        /// Get a smaller image from a bigger one
        /// </summary>
        /// <param name="inputBitmap">The big bitmap you want to get a part of</param>
        /// <param name="newBitmapDimensions">The dimensions of the new bitmap</param>
        /// <returns>The little bitmap</returns>
        protected Bitmap GetSmallBitmapFromBigOne(Bitmap inputBitmap, Rectangle newBitmapDimensions)
        {
            Bitmap sample = new Bitmap(newBitmapDimensions.Width, newBitmapDimensions.Height);
            Graphics g = Graphics.FromImage(sample);
            g.DrawImage(inputBitmap, new Rectangle(0, 0, sample.Width, sample.Height), newBitmapDimensions, GraphicsUnit.Pixel);
            return sample;
        }
        /// <summary>
        /// Returns the closest string from a list of options
        /// </summary>
        /// <param name="options">an array of all the possibilities</param>
        /// <param name="testString">the string you want to compare</param>
        /// <returns>The closest option</returns>
        protected static string FindClosestMatch(List<string> options, string testString)
        {
            var closestMatch = "";
            var closestDistance = int.MaxValue;

            foreach (var item in options)
            {
                var distance = LevenshteinDistance(item, testString);
                if (distance < closestDistance)
                {
                    closestMatch = item;
                    closestDistance = distance;
                }
            }
            return closestMatch;
        }
        //This method has been generated with the help of ChatGPT
        /// <summary>
        /// Method that computes a score of distance between two strings
        /// </summary>
        /// <param name="string1">The first string (order irrelevant)</param>
        /// <param name="string2">The second string (order irrelevant)</param>
        /// <returns>The levenshtein distance</returns>
        protected static int LevenshteinDistance(string string1, string string2)
        {
            if (string.IsNullOrEmpty(string1))
            {
                return string.IsNullOrEmpty(string2) ? 0 : string2.Length;
            }
```

```
            if (string.IsNullOrEmpty(string2))
            {
                return string.IsNullOrEmpty(string1) ? 0 : string1.Length;
            }

            var d = new int[string1.Length + 1, string2.Length + 1];
            for (var i = 0; i <= string1.Length; i++)
            {
                d[i, 0] = i;
            }

            for (var j = 0; j <= string2.Length; j++)
            {
                d[0, j] = j;
            }

            for (var i = 1; i <= string1.Length; i++)
            {
                for (var j = 1; j <= string2.Length; j++)
                {
                    var cost = (string1[i - 1] == string2[j - 1]) ? 0 : 1;
                    d[i, j] = Math.Min(Math.Min(d[i - 1, j] + 1, d[i, j - 1] + 1), d[i - 1, j - 1] + cost);
                }
            }

            return d[string1.Length, string2.Length];
        }
        public virtual string ToJSON()
        {
            string result = "";

            result += "\"" + Name + "\"" + ":{" + Environment.NewLine;
            result += "\t" + "\"x\":" + Bounds.X + "," + Environment.NewLine;
            result += "\t" + "\"y\":" + Bounds.Y + "," + Environment.NewLine;
            result += "\t" + "\"width\":" + Bounds.Width + Environment.NewLine;
            result += "}";

            return result;
        }
    }
}
```

4.7 DriverData.cs

```
/// Author : Maxime Rohmer
/// Date : 08/05/2023
/// File : DriverData.cs
/// Brief : Class used to store Driver informations
/// Version : 0.1

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Test_Merge
{
    public class DriverData
    {
        public bool DRS;        //True = Drs is opened
        public int GapToLeader; //In ms
        public int LapTime;     //In ms
        public string Name;     //Ex: LECLERC
        public int Position;    //Ex: 1
        public int Sector1;     //in ms
        public int Sector2;     //in ms
        public int Sector3;     //in ms
        public Tyre CurrentTyre;//Ex Soft 11 laps

        public DriverData(bool dRS, int gapToLeader, int lapTime, string name, int position, int sector1, int sector2, int sector3, Tyre tyre)
        {
            DRS = dRS;
            GapToLeader = gapToLeader;
            LapTime = lapTime;
            Name = name;
            Position = position;
            Sector1 = sector1;
            Sector2 = sector2;
            Sector3 = sector3;
            CurrentTyre = tyre;
        }
        public DriverData()
        {
            DRS = false;
            GapToLeader = -1;
            LapTime = -1;
            Name = "Unknown";
            Position = -1;
            Sector1 = -1;
            Sector2 = -1;
            Sector3 = -1;
            CurrentTyre = new Tyre(Tyre.Type.Undefined, -1);
        }
        /// <summary>
        /// Method that displays all the data found in a string
        /// </summary>
        /// <returns>string containing all the driver datas</returns>
        public override string ToString()
        {
            string result = "";

            //Position
            result += "Position : " + Position + Environment.NewLine;
            //Gap
            result += "GapToLeader : " + Reader.ConvertMsToTime(GapToLeader) + Environment.NewLine;
            //LapTime
            result += "LapTime : " + Reader.ConvertMsToTime(LapTime) + Environment.NewLine;
            //DRS
            result += "DRS : " + DRS + Environment.NewLine;
            //Tyres
            result += "Uses " + CurrentTyre.Coumpound + " tyre " + CurrentTyre.NumberOfLaps + " laps old" + Environment.NewLine;
            //Name
            result += "DriverName : " + Name + Environment.NewLine;
            //Sector 1
            result += "Sector1 : " + Reader.ConvertMsToTime(Sector1) + Environment.NewLine;
            //Sector 1
            result += "Sector2 : " + Reader.ConvertMsToTime(Sector2) + Environment.NewLine;
            //Sector 1
            result += "Sector3 : " + Reader.ConvertMsToTime(Sector3) + Environment.NewLine;

            return result;
        }
    }
    //Structure to store tyres infos
    public struct Tyre
    {
        //If new tyres were to be added you will have to need to change this enum
        public enum Type
        {
            Soft,
            Medium,
```

```
            Hard,
            Inter,
            Wet,
            Undefined
        }
        public Type Coumpound;
        public int NumberOfLaps;
        public Tyre(Type type, int laps)
        {
            Coumpound = type;
            NumberOfLaps = laps;
        }
    }
}
```

4.8 DriverLapTimeWindow.cs

```
/// Author : Maxime Rohmer
/// Date : 08/05/2023
/// File : DriverLapTimeWindow
/// Brief : Window containing infos about the lap time of a driver
/// Version : 0.1

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Drawing;

namespace Test_Merge
{
    internal class DriverLapTimeWindow:Window
    {
        public DriverLapTimeWindow(Bitmap image, Rectangle bounds, bool generateEngine = true) : base(image, bounds,generateEngine)
        {
            Name = "LapTime";
        }
        /// <summary>
        /// Decodes the lap time contained in the image using OCR Tesseract
        /// </summary>
        /// <returns>The laptime in int (ms)</returns>
        public override async Task<object> DecodePng()
        {
            int result = await GetTimeFromPng(WindowImage, OcrImage.WindowType.LapTime, Engine);
            return result;
        }
    }
}
```

4.9 DriverSectorWindow.cs

```
/// Author : Maxime Rohmer
/// Date : 08/05/2023
/// File : DriverSectorWindow.cs
/// Brief : Window containing infos about a driver sector time. Can be the first second or third, does not matter.
/// Version : 0.1

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Drawing;

namespace Test_Merge
{
    internal class DriverSectorWindow:Window
    {
        public DriverSectorWindow(Bitmap image, Rectangle bounds, int sectorId, bool generateEngine = true) : base(image, bounds,generateEngine)
        {
            Name = "Sector"+sectorId;
        }
        /// <summary>
        /// Decodes the sector
        /// </summary>
        /// <returns>the sector time in int (ms)</returns>
        public override async Task<object> DecodePng()
        {
            int ocrResult = await GetTimeFromPng(WindowImage, OcrImage.WindowType.Sector, Engine);
            return ocrResult;
        }
    }
}
```

4.10 Form1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Test_Merge
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void btnSettings_Click(object sender, EventArgs e)
        {
            Settings settingsForm = new Settings();
            settingsForm.ShowDialog();
            MessageBox.Show(settingsForm.GrandPrixUrl + Environment.NewLine + settingsForm.GrandPrixName + Environment.NewLine + settingsForm.GrandPrixYear);
        }
    }
}
```

4.11 Reader.cs

```
/// Author : Maxime Rohmer
/// Date : 08/05/2023
/// File : Reader.cs
/// Brief : Class used to Read the config file for the OCR
/// Version : 0.1

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Drawing;
using System.Windows.Forms;
using System.IO;
using System.Text.Json;

namespace Test_Merge
{
    public class Reader
    {
        const int NUMBER_OF_DRIVERS = 20;
        public List<string> Drivers;
        public List<Zone> MainZones;

        public Reader(string configFile, Bitmap image,bool loadOCR = true)
        {
            MainZones = Load(image,configFile,ref Drivers,loadOCR);
        }
        /// <summary>
        /// Method that reads the JSON config file and create all the Zones and Windows
        /// </summary>
        /// <param name="imageNumber">The image #id on wich you want to create the zones on</param>
        public static List<Zone> Load(Bitmap image,string configFilePath,ref List<string> driverListToFill,bool LoadOCR)
        {
            List<Zone> mainZones = new List<Zone>();
            Bitmap fullImage = image;
            List<string> drivers;
            Zone mainZone;

            try
            {
                using (var streamReader = new StreamReader(configFilePath))
                {
                    var jsonText = streamReader.ReadToEnd();
                    var jsonDocument = JsonDocument.Parse(jsonText);

                    var driversNames = jsonDocument.RootElement.GetProperty("Drivers");
                    driverListToFill = new List<string>();

                    foreach (var nameElement in driversNames.EnumerateArray())
                    {
                        driverListToFill.Add(nameElement.GetString());
                    }

                    var mainProperty = jsonDocument.RootElement.GetProperty("Main");
                    Point MainPosition = new Point(mainProperty.GetProperty("x").GetInt32(), mainProperty.GetProperty("y").GetInt32());
                    Size MainSize = new Size(mainProperty.GetProperty("width").GetInt32(), mainProperty.GetProperty("height").GetInt32());
                    Rectangle MainRectangle = new Rectangle(MainPosition, MainSize);
                    mainZone = new Zone(image, MainRectangle,"Main");

                    var zones = mainProperty.GetProperty("Zones");
                    var driverZone = zones[0].GetProperty("DriverZone");

                    Point FirstZonePosition = new Point(driverZone.GetProperty("x").GetInt32(), driverZone.GetProperty("y").GetInt32());
                    Size FirstZoneSize = new Size(driverZone.GetProperty("width").GetInt32(), driverZone.GetProperty("height").GetInt32());

                    var windows = driverZone.GetProperty("Windows");

                    var driverPosition = windows[0].GetProperty("Position");
                    Size driverPositionArea = new Size(driverPosition.GetProperty("width").GetInt32(), FirstZoneSize.Height);
                    Point driverPositionPosition = new Point(driverPosition.GetProperty("x").GetInt32(), driverPosition.GetProperty("y").GetInt32());

                    var driverGapToLeader = windows[0].GetProperty("GapToLeader");
                    Size driverGapToLeaderArea = new Size(driverGapToLeader.GetProperty("width").GetInt32(), FirstZoneSize.Height);
                    Point driverGapToLeaderPosition = new Point(driverGapToLeader.GetProperty("x").GetInt32(), driverGapToLeader.GetProperty("y").GetInt32());

                    var driverLapTime = windows[0].GetProperty("LapTime");
                    Size driverLapTimeArea = new Size(driverLapTime.GetProperty("width").GetInt32(), FirstZoneSize.Height);
                    Point driverLapTimePosition = new Point(driverLapTime.GetProperty("x").GetInt32(), driverLapTime.GetProperty("y").GetInt32());


                    var driverDrs = windows[0].GetProperty("DRS");
                    Size driverDrsArea = new Size(driverDrs.GetProperty("width").GetInt32(), FirstZoneSize.Height);
                    Point driverDrsPosition = new Point(driverDrs.GetProperty("x").GetInt32(), driverDrs.GetProperty("y").GetInt32());

                    var driverTyres = windows[0].GetProperty("Tyres");
                    Size driverTyresArea = new Size(driverTyres.GetProperty("width").GetInt32(), FirstZoneSize.Height);
                    Point driverTyresPosition = new Point(driverTyres.GetProperty("x").GetInt32(), driverTyres.GetProperty("y").GetInt32());
```

```
                    var driverName = windows[0].GetProperty("Name");
                    Size driverNameArea = new Size(driverName.GetProperty("width").GetInt32(), FirstZoneSize.Height);
                    Point driverNamePosition = new Point(driverName.GetProperty("x").GetInt32(), driverName.GetProperty("y").GetInt32());

                    var driverSector1 = windows[0].GetProperty("Sector1");
                    Size driverSector1Area = new Size(driverSector1.GetProperty("width").GetInt32(), FirstZoneSize.Height);
                    Point driverSector1Position = new Point(driverSector1.GetProperty("x").GetInt32(), driverSector1.GetProperty("y").GetInt32());

                    var driverSector2 = windows[0].GetProperty("Sector2");
                    Size driverSector2Area = new Size(driverSector2.GetProperty("width").GetInt32(), FirstZoneSize.Height);
                    Point driverSector2Position = new Point(driverSector2.GetProperty("x").GetInt32(), driverSector2.GetProperty("y").GetInt32());

                    var driverSector3 = windows[0].GetProperty("Sector3");
                    Size driverSector3Area = new Size(driverSector3.GetProperty("width").GetInt32(), FirstZoneSize.Height);
                    Point driverSector3Position = new Point(driverSector3.GetProperty("x").GetInt32(), driverSector3.GetProperty("y").GetInt32());

                    float offset = (((float)mainZone.ZoneImage.Height - (float)(driverListToFill.Count * FirstZoneSize.Height)) /
(float)driverListToFill.Count);
                    Bitmap MainZoneImage = mainZone.ZoneImage;
                    List<Zone> zonesToAdd = new List<Zone>();
                    List<Bitmap> zonesImages = new List<Bitmap>();

                    for (int i = 0; i < NUMBER_OF_DRIVERS; i++)
                    {
                        Point tmpPos = new Point(0, FirstZonePosition.Y + i * FirstZoneSize.Height - Convert.ToInt32(i * offset));
                        Zone newDriverZone = new Zone(MainZoneImage, new Rectangle(tmpPos, FirstZoneSize), "DriverZone");
                        zonesToAdd.Add(newDriverZone);
                        zonesImages.Add(newDriverZone.ZoneImage);

                        newDriverZone.ZoneImage.Save("Driver"+i+".png");
                    }

                    //Parallel.For(0, NUMBER_OF_DRIVERS, i =>
                    for (int i = 0; i < NUMBER_OF_DRIVERS; i++)
                    {
                        Zone newDriverZone = zonesToAdd[(int)i];
                        Bitmap zoneImg = zonesImages[(int)i];

                        newDriverZone.AddWindow(new DriverPositionWindow(zoneImg, new Rectangle(driverPositionPosition, driverPositionArea),LoadOCR));
                        newDriverZone.AddWindow(new DriverGapToLeaderWindow(zoneImg, new Rectangle(driverGapToLeaderPosition, driverGapToLeaderArea), LoadOCR));
                        newDriverZone.AddWindow(new DriverLapTimeWindow(zoneImg, new Rectangle(driverLapTimePosition, driverLapTimeArea), LoadOCR));
                        newDriverZone.AddWindow(new DriverDrsWindow(zoneImg, new Rectangle(driverDrsPosition, driverDrsArea), LoadOCR));
                        newDriverZone.AddWindow(new DriverTyresWindow(zoneImg, new Rectangle(driverTyresPosition, driverTyresArea), LoadOCR));
                        newDriverZone.AddWindow(new DriverNameWindow(zoneImg, new Rectangle(driverNamePosition, driverNameArea), LoadOCR));
                        newDriverZone.AddWindow(new DriverSectorWindow(zoneImg, new Rectangle(driverSector1Position, driverSector1Area),1, LoadOCR));
                        newDriverZone.AddWindow(new DriverSectorWindow(zoneImg, new Rectangle(driverSector2Position, driverSector2Area),2, LoadOCR));
                        newDriverZone.AddWindow(new DriverSectorWindow(zoneImg, new Rectangle(driverSector3Position, driverSector3Area),3, LoadOCR));

                        mainZone.AddZone(newDriverZone);
                    }//);
                    //MessageBox.Show("We have a main zone with " + MainZone.Zones.Count() + " Driver zones with " + MainZone.Zones[4].Windows.Count() + "
windows each and we have " + Drivers.Count + " drivers");
                    mainZones.Add(mainZone);
                }
            }
            catch (IOException ex)
            {
                MessageBox.Show("Error reading JSON file: " + ex.Message);
            }
            catch (JsonException ex)
            {
                MessageBox.Show("Invalid JSON format: " + ex.Message);
            }
            return mainZones;
        }
        /// <summary>
        /// Method that calls all the zones and windows to get the content they can find on the image to display them
        /// </summary>
        /// <param name="idImage">The id of the image we are working with</param>
        /// <returns>a string representation of all the returns</returns>
        public async Task<string> Decode(List<Zone> mainZones,List<string> drivers)
        {
            string result = "";
            List<DriverData> mainResults = new List<DriverData>();

            //Decode
            for (int mainZoneId = 0; mainZoneId < mainZones.Count; mainZoneId++)
            {
                switch (mainZoneId)
                {
                    case 0:
                        //Main Zone
                        foreach (Zone z in mainZones[mainZoneId].Zones)
                        {
                            mainResults.Add(await z.Decode(Drivers));
                        }
                        break;
                        //Next there could be a Title Zone and TrackInfoZone
                }
            }

            //Display
            foreach (DriverData driver in mainResults)
```

```
            {
                result += driver.ToString();
                result += Environment.NewLine;
            }

            return result;
        }
        /// <summary>
        /// Method that can be used to convert an amount of miliseconds into a more readable human form
        /// </summary>
        /// <param name="amountOfMs">The given amount of miliseconds ton convert</param>
        /// <returns>A human readable string that represents the ms</returns>
        public static string ConvertMsToTime(int amountOfMs)
        {
            //Convert.ToInt32 would round upand I dont want that
            int minuts = (int)((float)amountOfMs / (1000f * 60f));
            int seconds = (int)((amountOfMs - (minuts * 60f * 1000f)) / 1000);
            int ms = amountOfMs - ((minuts * 60 * 1000) + (seconds * 1000));

            return minuts + ":" + seconds.ToString("00") + ":" + ms.ToString("000");
        }
        /// <summary>
        /// Old method that can draw on an image where the windows and zones are created. mostly used for debugging
        /// </summary>
        /// <param name="idImage">the #id of the image we are working with</param>
        /// <returns>the drawed bitmap</returns>
        public Bitmap Draw(Bitmap image,List<Zone> mainZones)
        {

            Graphics g = Graphics.FromImage(image);

            foreach (Zone z in mainZones)
            {
                int count = 0;
                foreach (Zone zz in z.Zones)
                {
                    g.DrawRectangle(Pens.Red, z.Bounds);
                    foreach (Window w in zz.Windows)
                    {
                        g.DrawRectangle(Pens.Blue, new Rectangle(z.Bounds.X + zz.Bounds.X, z.Bounds.Y + zz.Bounds.Y, zz.Bounds.Width, zz.Bounds.Height));
                    }

                    count++;
                }
            }

            return image;
        }
    }
}
```

4.12 Zone.cs

```
/// Author : Maxime Rohmer
/// Date : 08/05/2023
/// File : Zone.cs
/// Brief : Class that contains all the methods and infos for a zone. This is designed to be potentially be inherited.
/// Version : 0.1

using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Test_Merge
{
    public class Zone
    {
        private Rectangle _bounds;
        private List<Zone> _zones;
        private List<Window> _windows;
        private Bitmap _image;
        private string _name;

        public Bitmap ZoneImage
        {
            get
            {
                //This little trickery lets you have the image that the zone sees
                Bitmap sample = new Bitmap(Bounds.Width, Bounds.Height);
                Graphics g = Graphics.FromImage(sample);
                g.DrawImage(Image, new Rectangle(0, 0, sample.Width, sample.Height), Bounds, GraphicsUnit.Pixel);
                return sample;
            }
        }
        public Bitmap Image
        {
            get { return _image; }
            set
            {
                //It automatically sets the image for the contained windows and zones
                _image = Image;
                foreach (Window w in Windows)
                {
                    w.Image = ZoneImage;
                }
                foreach (Zone z in Zones)
                {
                    z.Image = Image;
                }
            }
        }

        public Rectangle Bounds { get => _bounds; protected set => _bounds = value; }
        public List<Zone> Zones { get => _zones; protected set => _zones = value; }
        public List<Window> Windows { get => _windows; protected set => _windows = value; }
        public string Name { get => _name; protected set => _name = value; }

        public Zone(Bitmap image, Rectangle bounds, string name)
        {
            Windows = new List<Window>();
            Zones = new List<Zone>();
            Name = name;

            //You cant set the image in the CTOR because the processing is impossible at first initiation
            _image = image;
            Bounds = bounds;
        }
        /// <summary>
        /// Adds a zone to the list of zones
        /// </summary>
        /// <param name="zone">The zone you want to add</param>
        public virtual void AddZone(Zone zone)
        {
            Zones.Add(zone);
        }
        /// <summary>
        /// Add a window to the list of windows
        /// </summary>
        /// <param name="window">the window you want to add</param>
        public virtual void AddWindow(Window window)
        {
            Windows.Add(window);
        }
        /// <summary>
        /// Calls all the windows to do OCR and to give back the results so we can send them to the model
        /// </summary>
        /// <param name="driverList">A list of all the driver in the race to help with text recognition</param>
        /// <returns>A driver data object that contains all the infos about a driver</returns>
```

```
        public virtual async Task<DriverData> Decode(List<string> driverList)
        {
            int sectorCount = 0;
            DriverData result = new DriverData();
            Parallel.ForEach(Windows, async w =>
            {
                // A switch would be prettier but I dont think its supported in this C# version
                if (w is DriverNameWindow)
                    result.Name = (string)await (w as DriverNameWindow).DecodePng(driverList);
                if (w is DriverDrsWindow)
                    result.DRS = (bool)await (w as DriverDrsWindow).DecodePng();
                if (w is DriverGapToLeaderWindow)
                    result.GapToLeader = (int)await (w as DriverGapToLeaderWindow).DecodePng();
                if (w is DriverLapTimeWindow)
                    result.LapTime = (int)await (w as DriverLapTimeWindow).DecodePng();
                if (w is DriverPositionWindow)
                    result.Position = (int)await (w as DriverPositionWindow).DecodePng();
                if (w is DriverSectorWindow)
                {
                    sectorCount++;
                    if (sectorCount == 1)
                        result.Sector1 = (int)await (w as DriverSectorWindow).DecodePng();
                    if (sectorCount == 2)
                        result.Sector2 = (int)await (w as DriverSectorWindow).DecodePng();
                    if (sectorCount == 3)
                        result.Sector3 = (int)await (w as DriverSectorWindow).DecodePng();
                }
                if (w is DriverTyresWindow)
                    result.CurrentTyre = (Tyre)await (w as DriverTyresWindow).DecodePng();
            });
            return result;
        }
        public virtual Bitmap Draw()
        {
            Bitmap img;

            //If its the main zone we want to see everything
            if (Zones.Count > 0)
            {
                img = Image;
            }
            else
            {
                img = ZoneImage;
            }

            Graphics g = Graphics.FromImage(img);

            //If its the main zone we need to visualize the Zone bounds displayed
            if (Zones.Count > 0)
                g.DrawRectangle(new Pen(Brushes.Violet, 5), Bounds);

            foreach (Zone z in Zones)
            {
                Rectangle newBounds = new Rectangle(z.Bounds.X, z.Bounds.Y + Bounds.Y, z.Bounds.Width, z.Bounds.Height);
                g.DrawRectangle(Pens.Red, newBounds);
            }
            foreach (Window w in Windows)
            {
                g.DrawRectangle(Pens.Blue, w.Bounds);
            }
            return img;
        }
        public void ResetZones()
        {
            Zones.Clear();
        }
        public void ResetWindows()
        {
            foreach (Zone z in Zones)
            {
                z.ResetWindows();
            }
            Windows.Clear();
        }
        public virtual string ToJSON()
        {
            string result = "";
            result += "\"" + Name + "\":{" + Environment.NewLine;
            result += "\t" + "\"x\":" + Bounds.X + "," + Environment.NewLine;
            result += "\t" + "\"y\":" + Bounds.Y + "," + Environment.NewLine;
            result += "\t" + "\"width\":" + Bounds.Width + "," + Environment.NewLine;
            result += "\t" + "\"height\":" + Bounds.Height;

            if (Windows.Count != 0)
            {
                result += "," + Environment.NewLine;

                result += "\t" + "\"Windows\":[" + Environment.NewLine;
                result += "\t\t{" + Environment.NewLine;
                int Wcount = 0;
                foreach (Window w in Windows)
                {
                    result += "\t\t" + w.ToJSON();
```

```
                    Wcount++;
                    if (Wcount != Windows.Count)
                        result += ",";
                }
                result += "\t\t}" + Environment.NewLine;
                result += "\t" + "]" + Environment.NewLine;
            }
            else
            {
                result += Environment.NewLine;
            }
            if (Zones.Count != 0)
            {
                result += "," + Environment.NewLine;

                result += "\t" + "\"Zones\":[" + Environment.NewLine;
                result += "\t\t{" + Environment.NewLine;
                int Zcount = 0;
                //foreach (Zone z in Zones)
                //{
                result += "\t\t" + Zones[0].ToJSON();
                Zcount++;
                if (Zcount != Zones.Count)
                    //result += ",";
                    //}
                    result += "\t\t}" + Environment.NewLine;
                result += "\t" + "]" + Environment.NewLine;
            }
            else
            {
                result += Environment.NewLine;
            }

            result += "}";

            return result;
        }
        /// <summary>
        /// Checks if the given Rectangle fits in the current zone
        /// </summary>
        /// <param name="InputRectangle">The Rectangle you want to check the fittment</param>
        /// <returns></returns>
        protected bool Fits(Rectangle inputRectangle)
        {
            if (inputRectangle.X + inputRectangle.Width > Bounds.Width || inputRectangle.Y + inputRectangle.Height > Bounds.Height || inputRectangle.X < 0 ||
inputRectangle.Y < 0)
            {
                return false;
            }
            else
            {
                return true;
            }
        }
    }
}
```

4.13 DriverDrsWindow.cs

```
/// Author : Maxime Rohmer
/// Date : 08/05/2023
/// File : DriverDrsWindow.cs
/// Brief : Window containing DRS related method and infos
/// Version : 0.1

using System;
using System.Collections.Generic;
using System.Drawing;
using System.Drawing.Imaging;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Tesseract;

namespace Test_Merge
{
    internal class DriverDrsWindow:Window
    {
        private static int EmptyDrsGreenValue = -1;
        private static Random rnd = new Random();
        public DriverDrsWindow(Bitmap image, Rectangle bounds,bool generateEngine = true) : base(image, bounds,generateEngine)
        {
            Name = "DRS";
        }
        public override async Task<object> DecodePng()
        {
            bool result = false;
            int greenValue = GetGreenPixels();
            if (EmptyDrsGreenValue == -1)
                EmptyDrsGreenValue = greenValue;

            if (greenValue > EmptyDrsGreenValue + EmptyDrsGreenValue / 100 * 30)
                result = true;

            return result;
        }
        private unsafe int GetGreenPixels()
        {
            int tot = 0;

            Bitmap bmp = WindowImage;
            Rectangle rect = new Rectangle(0, 0, bmp.Width, bmp.Height);
            BitmapData bmpData = bmp.LockBits(rect, ImageLockMode.ReadOnly, bmp.PixelFormat);
            int bytesPerPixel = Bitmap.GetPixelFormatSize(bmp.PixelFormat) / 8;

            unsafe
            {
                byte* ptr = (byte*)bmpData.Scan0.ToPointer();
                for (int y = 0; y < bmp.Height; y++)
                {
                    byte* currentLine = ptr + (y * bmpData.Stride);
                    for (int x = 0; x < bmp.Width; x++)
                    {
                        byte* pixel = currentLine + (x * bytesPerPixel);

                        byte blue = pixel[0];
                        byte green = pixel[1];
                        byte red = pixel[2];

                        if (green > blue * 1.5 && green > red * 1.5)
                        {
                            tot++;
                        }
                    }
                }
            }
            bmp.UnlockBits(bmpData);

            return tot;
        }
        public Rectangle GetBox()
        {
            var tessImage = Pix.LoadFromMemory(ImageToByte(WindowImage));
            Engine.SetVariable("tessedit_char_whitelist", "");
            Page page = Engine.Process(tessImage);

            using (var iter = page.GetIterator())
            {
                iter.Begin();
                do
                {
                    Rect boundingBox;

                    // Get the bounding box for the current element
                    if (iter.TryGetBoundingBox(PageIteratorLevel.Word, out boundingBox))
                    {
                        page.Dispose();
```

```
                        return new Rectangle(boundingBox.X1, boundingBox.X2, boundingBox.Width, boundingBox.Height);
                    }
                } while (iter.Next(PageIteratorLevel.Word));

                page.Dispose();
                return new Rectangle(0, 0, 0, 0);
            }
        }
    }
}
```

4.14 DriverNameWindow.cs

```
/// Author : Maxime Rohmer
/// Date : 08/05/2023
/// File : DriverNameWindow
/// Brief : Window containing infos about the name of the driver
/// Version : 0.1

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Drawing;

namespace Test_Merge
{
    public class DriverNameWindow : Window
    {
        public static Random rnd = new Random();
        public DriverNameWindow(Bitmap image, Rectangle bounds, bool generateEngine = true) : base(image, bounds,generateEngine)
        {
            Name = "Name";
        }
        /// <summary>
        /// Decodes using OCR wich driver name is in the image
        /// </summary>
        /// <param name="DriverList"></param>
        /// <returns>The driver name in string</returns>
        public override async Task<object> DecodePng(List<string> DriverList)
        {
            string result = "";
            result = await GetStringFromPng(WindowImage, Engine);

            if (!IsADriver(DriverList, result))
            {
                //I put everything in uppercase to try to lower the chances of bad answers
                result = FindClosestMatch(DriverList.ConvertAll(d => d.ToUpper()), result.ToUpper());
            }
            return result;
        }
        /// <summary>
        /// Verifies that the name found in the OCR is a valid name
        /// </summary>
        /// <param name="driverList"></param>
        /// <param name="potentialDriver"></param>
        /// <returns>If ye or no the driver exists</returns>
        private static bool IsADriver(List<string> driverList, string potentialDriver)
        {
            bool result = false;
            //I cant use drivers.Contains because it has missmatched cases and all
            foreach (string name in driverList)
            {
                if (name.ToUpper() == potentialDriver.ToUpper())
                    result = true;
            }
            return result;
        }
    }
}
```

4.15 DriverTyresWindow.cs

```
/// Author : Maxime Rohmer
/// Date : 08/05/2023
/// File : DriverTyresWindow.cs
/// Brief : Window containing infos about a driver's tyre
/// Version : 0.1

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Drawing;

namespace Test_Merge
{
    public class DriverTyresWindow:Window
    {
        private static Random rnd = new Random();
        int seed = rnd.Next(0, 10000);

        //Those are the colors I found but you can change them if they change in the future like in 2019
        public static Color SOFT_TYRE_COLOR = Color.FromArgb(0xff, 0x00, 0x00);
        public static Color MEDIUM_TYRE_COLOR = Color.FromArgb(0xf5, 0xbf, 0x00);
        public static Color HARD_TYRE_COLOR = Color.FromArgb(0xa4, 0xa5, 0xa8);
        public static Color INTER_TYRE_COLOR = Color.FromArgb(0x00, 0xa4, 0x2e);
        public static Color WET_TYRE_COLOR = Color.FromArgb(0x27, 0x60, 0xa6);
        public static Color EMPTY_COLOR = Color.FromArgb(0x20, 0x20, 0x20);

        public DriverTyresWindow(Bitmap image, Rectangle bounds, bool generateEngine = true) : base(image, bounds,generateEngine)
        {
            Name = "Tyres";
        }
        /// <summary>
        /// This will decode the content of the image
        /// </summary>
        /// <returns>And object containing what was on the image</returns>
        public override async Task<object> DecodePng()
        {
            return await GetTyreInfos();
        }
        /// <summary>
        /// Method that will decode whats on the image and return the tyre infos it could manage to recover
        /// </summary>
        /// <returns>A tyre object containing tyre infos</returns>
        private async Task<Tyre> GetTyreInfos()
        {
            Bitmap tyreZone = GetSmallBitmapFromBigOne(WindowImage, FindTyreZone());
            Tyre.Type type = Tyre.Type.Undefined;
            type = GetTyreTypeFromColor(OcrImage.GetAvgColorFromBitmap(tyreZone));
            int laps = -1;

            string number = await GetStringFromPng(tyreZone, Engine, "0123456789", OcrImage.WindowType.Tyre);
            try
            {
                laps = Convert.ToInt32(number);
            }
            catch
            {
                //We could not convert the number so its a letter so its 0 laps old
                laps = 0;
            }
            //tyreZone.Save(Reader.DEBUG_DUMP_FOLDER + "Tyre" + type + "Laps" + laps + '#' + rnd.Next(0, 1000) + ".png");
            return new Tyre(type, laps);
        }
        /// <summary>
        /// Finds where the important part of the image is
        /// </summary>
        /// <returns>A rectangle containing position and dimensions of the important part of the image</returns>
        private Rectangle FindTyreZone()
        {
            Bitmap bmp = WindowImage;
            int currentPosition = bmp.Width;
            int height = bmp.Height / 2;
            Color limitColor = Color.FromArgb(0x50, 0x50, 0x50);
            Color currentColor = Color.FromArgb(0, 0, 0);

            Size newWindowSize = new Size(bmp.Height - Convert.ToInt32((float)bmp.Height / 100f * 25f), bmp.Height - Convert.ToInt32((float)bmp.Height / 100f *
35f));

            while (currentColor.R <= limitColor.R && currentColor.G <= limitColor.G && currentColor.B <= limitColor.B && currentPosition > 0)
            {
                currentPosition--;
                currentColor = bmp.GetPixel(currentPosition, height);
            }

            //Its here to let the new window include a little bit of the right
            int CorrectedX = currentPosition - (newWindowSize.Width) + Convert.ToInt32((float)newWindowSize.Width / 100f * 10f);
            int CorrectedY = Convert.ToInt32((float)newWindowSize.Height / 100f * 35f);
```

```
            if (CorrectedX <= 0)
                return new Rectangle(0, 0, newWindowSize.Width, newWindowSize.Height);

            return new Rectangle(CorrectedX, CorrectedY, newWindowSize.Width, newWindowSize.Height);
        }
        //This method has been created with the help of chatGPT
        /// <summary>
        /// Methods that compares a list of colors to see wich is the closest from the input color and decide wich tyre type it is
        /// </summary>
        /// <param name="inputColor">The color that you found</param>
        /// <returns>The tyre type</returns>
        public Tyre.Type GetTyreTypeFromColor(Color inputColor)
        {
            Tyre.Type type = Tyre.Type.Undefined;
            List<Color> colors = new List<Color>();
            //dont forget that if for some reason someday F1 adds a new Tyre type you will need to add it in the constants but also here in the list
            //You will also need to add it below in the Tyre object's enum and add an if in the end of this method
            colors.Add(SOFT_TYRE_COLOR);
            colors.Add(MEDIUM_TYRE_COLOR);
            colors.Add(HARD_TYRE_COLOR);
            colors.Add(INTER_TYRE_COLOR);
            colors.Add(WET_TYRE_COLOR);
            colors.Add(EMPTY_COLOR);

            Color closestColor = colors[0];
            int closestDistance = int.MaxValue;
            foreach (Color color in colors)
            {
                int distance = Math.Abs(color.R - inputColor.R) + Math.Abs(color.G - inputColor.G) + Math.Abs(color.B - inputColor.B);
                if (distance < closestDistance)
                {
                    closestColor = color;
                    closestDistance = distance;
                }
            }

            //We cant use a switch as the colors cant be constants ...
            if (closestColor == SOFT_TYRE_COLOR)
                type = Tyre.Type.Soft;
            if (closestColor == MEDIUM_TYRE_COLOR)
                type = Tyre.Type.Medium;
            if (closestColor == HARD_TYRE_COLOR)
                type = Tyre.Type.Hard;
            if (closestColor == INTER_TYRE_COLOR)
                type = Tyre.Type.Inter;
            if (closestColor == WET_TYRE_COLOR)
                type = Tyre.Type.Wet;
            if (closestColor == EMPTY_COLOR)
                return Tyre.Type.Undefined;

            return type;
        }
    }
}
```

4.16 OcrImage.cs

```
/// Author : Maxime Rohmer
/// Date : 08/05/2023
/// File : OcrImage.cs
/// Brief : Class containing all the methods used to enhance images for OCR
/// Version : 0.1

using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Drawing.Imaging;

namespace Test_Merge
{
    public class OcrImage
    {
        //this is a hardcoded value based on the colors of the F1TV data channel background you can change it if sometime in the future the color changes
        //Any color that has any of its R,G or B channel higher than the treshold will be considered as being usefull information
        public static Color F1TV_BACKGROUND_TRESHOLD = Color.FromArgb(0x50, 0x50, 0x50);
        Bitmap InputBitmap;
        public enum WindowType
        {
            LapTime,
            Text,
            Sector,
            Gap,
            Tyre,
        }

        /// <summary>
        /// Create a new Ocr image to help enhance the given bitmap for OCR
        /// </summary>
        /// <param name="inputBitmap">The image you want to enhance</param>
        public OcrImage(Bitmap inputBitmap)
        {
            InputBitmap = inputBitmap;
        }
        /// <summary>
        /// Enhances the image depending on wich type of window the image comes from
        /// </summary>
        /// <param name="type">The type of the window. Depending on it different enhancing features will be applied</param>
        /// <returns>The enhanced Bitmap</returns>
        public Bitmap Enhance(WindowType type = WindowType.Text)
        {
            Bitmap outputBitmap = (Bitmap)InputBitmap.Clone();
            switch (type)
            {
                case WindowType.LapTime:
                    outputBitmap = Tresholding(outputBitmap, 185);
                    outputBitmap = Resize(outputBitmap, 2);
                    outputBitmap = Dilatation(outputBitmap, 1);
                    outputBitmap = Erode(outputBitmap, 1);
                    break;
                case WindowType.Text:
                    outputBitmap = InvertColors(outputBitmap);
                    outputBitmap = Tresholding(outputBitmap, 165);
                    outputBitmap = Resize(outputBitmap, 2);
                    outputBitmap = Dilatation(outputBitmap, 1);
                    break;
                case WindowType.Tyre:
                    outputBitmap = RemoveUseless(outputBitmap);
                    outputBitmap = Resize(outputBitmap, 4);
                    outputBitmap = Dilatation(outputBitmap, 1);
                    break;
                default:
                    outputBitmap = Tresholding(outputBitmap, 165);
                    outputBitmap = Resize(outputBitmap, 4);
                    outputBitmap = Erode(outputBitmap, 1);
                    break;
            }
            return outputBitmap;
        }
        /// <summary>
        /// Method that convert a colored RGB bitmap into a GrayScale image
        /// </summary>
        /// <param name="inputBitmap">The Bitmap you want to convert</param>
        /// <returns>The bitmap in grayscale</returns>
        public static Bitmap Grayscale(Bitmap inputBitmap)
        {
            Rectangle rect = new Rectangle(0, 0, inputBitmap.Width, inputBitmap.Height);
            BitmapData bmpData = inputBitmap.LockBits(rect, ImageLockMode.ReadWrite, inputBitmap.PixelFormat);
            int bytesPerPixel = Bitmap.GetPixelFormatSize(inputBitmap.PixelFormat) / 8;

            unsafe
            {
                byte* ptr = (byte*)bmpData.Scan0.ToPointer();
                for (int y = 0; y < inputBitmap.Height; y++)
```

```
            {
                byte* currentLine = ptr + (y * bmpData.Stride);
                for (int x = 0; x < inputBitmap.Width; x++)
                {
                    byte* pixel = currentLine + (x * bytesPerPixel);

                    byte blue = pixel[0];
                    byte green = pixel[1];
                    byte red = pixel[2];

                    //Those a specific values to correct the weights so its more pleasing to the human eye
                    int gray = (int)(red * 0.3 + green * 0.59 + blue * 0.11);

                    pixel[0] = pixel[1] = pixel[2] = (byte)gray;
                }
            }
        }
        inputBitmap.UnlockBits(bmpData);

        return inputBitmap;
    }
    /// <summary>
    /// Method that binaries the input image up to a certain treshold given
    /// </summary>
    /// <param name="inputBitmap">the bitmap you want to convert to binary colors</param>
    /// <param name="threshold">The floor at wich the color is considered as white or black</param>
    /// <returns>The binarised bitmap</returns>
    public static Bitmap Tresholding(Bitmap inputBitmap, int threshold)
    {
        Rectangle rect = new Rectangle(0, 0, inputBitmap.Width, inputBitmap.Height);
        BitmapData bmpData = inputBitmap.LockBits(rect, ImageLockMode.ReadWrite, inputBitmap.PixelFormat);
        int bytesPerPixel = Bitmap.GetPixelFormatSize(inputBitmap.PixelFormat) / 8;

        unsafe
        {
            byte* ptr = (byte*)bmpData.Scan0.ToPointer();
            int bmpHeight = inputBitmap.Height;
            int bmpWidth = inputBitmap.Width;
            Parallel.For(0, bmpHeight, y =>
            {
                byte* currentLine = ptr + (y * bmpData.Stride);
                for (int x = 0; x < bmpWidth; x++)
                {
                    byte* pixel = currentLine + (x * bytesPerPixel);

                    byte blue = pixel[0];
                    byte green = pixel[1];
                    byte red = pixel[2];
                    //Those a specific values to correct the weights so its more pleasing to the human eye
                    int gray = (int)(red * 0.3 + green * 0.59 + blue * 0.11);
                    int value = gray < threshold ? 0 : 255;

                    pixel[0] = pixel[1] = pixel[2] = (byte)value;
                }
            });
        }
        inputBitmap.UnlockBits(bmpData);

        return inputBitmap;
    }
    /// <summary>
    /// Method that removes the pixels that are flagged as background
    /// </summary>
    /// <param name="inputBitmap">The bitmap you want to remove the background from</param>
    /// <returns>The Bitmap without the background</returns>
    public static Bitmap RemoveBG(Bitmap inputBitmap)
    {
        Rectangle rect = new Rectangle(0, 0, inputBitmap.Width, inputBitmap.Height);
        BitmapData bmpData = inputBitmap.LockBits(rect, ImageLockMode.ReadWrite, inputBitmap.PixelFormat);
        int bytesPerPixel = Bitmap.GetPixelFormatSize(inputBitmap.PixelFormat) / 8;

        unsafe
        {
            byte* ptr = (byte*)bmpData.Scan0.ToPointer();
            for (int y = 0; y < inputBitmap.Height; y++)
            {
                byte* currentLine = ptr + (y * bmpData.Stride);
                for (int x = 0; x < inputBitmap.Width; x++)
                {
                    byte* pixel = currentLine + (x * bytesPerPixel);

                    int B = pixel[0];
                    int G = pixel[1];
                    int R = pixel[2];

                    if (R <= F1TV_BACKGROUND_TRESHOLD.R && G <= F1TV_BACKGROUND_TRESHOLD.G && B <= F1TV_BACKGROUND_TRESHOLD.B)
                        pixel[0] = pixel[1] = pixel[2] = 0;
                }
            }
        }
        inputBitmap.UnlockBits(bmpData);

        return inputBitmap;
    }
```

```
        /// <summary>
        /// Method that removes all the useless things from the image and returns hopefully only the numbers
        /// </summary>
        /// <param name="inputBitmap">The bitmap you want to remove useless things from (Expects a cropped part of the TyreWindow)</param>
        /// <returns>The bitmap with (hopefully) only the digits</returns>
        public unsafe static Bitmap RemoveUseless(Bitmap inputBitmap)
        {
            //Note you can use something else than a cropped tyre window but I would recommend checking the code first to see if it fits your intended use
            Rectangle rect = new Rectangle(0, 0, inputBitmap.Width, inputBitmap.Height);
            BitmapData bmpData = inputBitmap.LockBits(rect, ImageLockMode.ReadWrite, inputBitmap.PixelFormat);
            int bytesPerPixel = Bitmap.GetPixelFormatSize(inputBitmap.PixelFormat) / 8;

            byte* ptr = (byte*)bmpData.Scan0.ToPointer();
            for (int y = 0; y < inputBitmap.Height; y++)
            {
                byte* currentLine = ptr + (y * bmpData.Stride);

                List<int> pixelsToRemove = new List<int>();

                bool fromBorder = true;

                for (int x = 0; x < inputBitmap.Width; x++)
                {
                    byte* pixel = currentLine + (x * bytesPerPixel);

                    int B = pixel[0];
                    int G = pixel[1];
                    int R = pixel[2];

                    if (fromBorder && B < F1TV_BACKGROUND_TRESHOLD.B && G < F1TV_BACKGROUND_TRESHOLD.G && R < F1TV_BACKGROUND_TRESHOLD.R)
                    {
                        pixelsToRemove.Add(x);
                    }
                    else
                    {
                        if (fromBorder)
                        {
                            fromBorder = false;
                            pixelsToRemove.Add(x);
                        }
                    }
                }
                fromBorder = true;
                for (int x = inputBitmap.Width - 1; x > 0; x--)
                {
                    byte* pixel = currentLine + (x * bytesPerPixel);

                    int B = pixel[0];
                    int G = pixel[1];
                    int R = pixel[2];

                    if (fromBorder && B < F1TV_BACKGROUND_TRESHOLD.B && G < F1TV_BACKGROUND_TRESHOLD.G && R < F1TV_BACKGROUND_TRESHOLD.R)
                    {
                        pixelsToRemove.Add(x);
                    }
                    else
                    {
                        if (fromBorder)
                        {
                            fromBorder = false;
                            pixelsToRemove.Add(x);
                        }
                    }
                }

                foreach (int pxPos in pixelsToRemove)
                {
                    byte* pixel = currentLine + (pxPos * bytesPerPixel);

                    pixel[0] = 0xFF;
                    pixel[1] = 0xFF;
                    pixel[2] = 0xFF;
                }
            }

            //Removing the color parts
            for (int y = 0; y < inputBitmap.Height; y++)
            {
                byte* currentLine = ptr + (y * bmpData.Stride);
                for (int x = 0; x < inputBitmap.Width; x++)
                {
                    byte* pixel = currentLine + (x * bytesPerPixel);

                    int B = pixel[0];
                    int G = pixel[1];
                    int R = pixel[2];

                    if (R >= F1TV_BACKGROUND_TRESHOLD.R + 15 || G >= F1TV_BACKGROUND_TRESHOLD.G + 15 || B >= F1TV_BACKGROUND_TRESHOLD.B + 15)
                    {
                        pixel[0] = 0xFF;
                        pixel[1] = 0xFF;
                        pixel[2] = 0xFF;
                    }
                }
```

```
            }

            inputBitmap.UnlockBits(bmpData);
            return inputBitmap;
        }
        /// <summary>
        /// Recovers the average colors from the Image. NOTE : It wont take in account colors that are lower than the background
        /// </summary>
        /// <param name="inputBitmap">The bitmap you want to get the average color from</param>
        /// <returns>The average color of the bitmap</returns>
        public static Color GetAvgColorFromBitmap(Bitmap inputBitmap)
        {
            Rectangle rect = new Rectangle(0, 0, inputBitmap.Width, inputBitmap.Height);
            BitmapData bmpData = inputBitmap.LockBits(rect, ImageLockMode.ReadWrite, inputBitmap.PixelFormat);
            int bytesPerPixel = Bitmap.GetPixelFormatSize(inputBitmap.PixelFormat) / 8;

            int totR = 0;
            int totG = 0;
            int totB = 0;

            int totPixels = 1;

            unsafe
            {
                byte* ptr = (byte*)bmpData.Scan0.ToPointer();
                int bmpHeight = inputBitmap.Height;
                int bmpWidth = inputBitmap.Width;
                Parallel.For(0, bmpHeight, y =>
                {
                    byte* currentLine = ptr + (y * bmpData.Stride);
                    for (int x = 0; x < bmpWidth; x++)
                    {
                        byte* pixel = currentLine + (x * bytesPerPixel);

                        int B = pixel[0];
                        int G = pixel[1];
                        int R = pixel[2];

                        if (R >= F1TV_BACKGROUND_TRESHOLD.R || G >= F1TV_BACKGROUND_TRESHOLD.G || B >= F1TV_BACKGROUND_TRESHOLD.B)
                        {
                            totPixels++;
                            totB += pixel[0];
                            totG += pixel[1];
                            totR += pixel[2];
                        }
                    }
                });
            }
            inputBitmap.UnlockBits(bmpData);

            return Color.FromArgb(255, Convert.ToInt32((float)totR / (float)totPixels), Convert.ToInt32((float)totG / (float)totPixels),
Convert.ToInt32((float)totB / (float)totPixels));
        }
        /// <summary>
        /// This method simply inverts all the colors in a Bitmap
        /// </summary>
        /// <param name="inputBitmap">the bitmap you want to invert the colors from</param>
        /// <returns>The bitmap with inverted colors</returns>
        public static Bitmap InvertColors(Bitmap inputBitmap)
        {
            Rectangle rect = new Rectangle(0, 0, inputBitmap.Width, inputBitmap.Height);
            BitmapData bmpData = inputBitmap.LockBits(rect, ImageLockMode.ReadWrite, inputBitmap.PixelFormat);
            int bytesPerPixel = Bitmap.GetPixelFormatSize(inputBitmap.PixelFormat) / 8;

            unsafe
            {
                byte* ptr = (byte*)bmpData.Scan0.ToPointer();
                for (int y = 0; y < inputBitmap.Height; y++)
                {
                    byte* currentLine = ptr + (y * bmpData.Stride);
                    for (int x = 0; x < inputBitmap.Width; x++)
                    {
                        byte* pixel = currentLine + (x * bytesPerPixel);

                        pixel[0] = (byte)(255 - pixel[0]);
                        pixel[1] = (byte)(255 - pixel[1]);
                        pixel[2] = (byte)(255 - pixel[2]);
                    }
                }
            }
            inputBitmap.UnlockBits(bmpData);

            return inputBitmap;
        }
        /// <summary>
        /// Methods that applies Bicubic interpolation to increase the size and resolution of an image
        /// </summary>
        /// <param name="inputBitmap">The bitmap you want to resize</param>
        /// <param name="resizeFactor">The factor of resizing you want to use. I recommend using even numbers</param>
        /// <returns>The bitmap witht the new size</returns>
        public static Bitmap Resize(Bitmap inputBitmap, int resizeFactor)
        {
            var resultBitmap = new Bitmap(inputBitmap.Width * resizeFactor, inputBitmap.Height * resizeFactor);
```

```
            using (var graphics = Graphics.FromImage(resultBitmap))
            {
                graphics.InterpolationMode = InterpolationMode.HighQualityBicubic;
                graphics.DrawImage(inputBitmap, new Rectangle(0, 0, resultBitmap.Width, resultBitmap.Height));
            }

            return resultBitmap;
        }
        /// <summary>
        /// method that Highlights the countours of a Bitmap
        /// </summary>
        /// <param name="inputBitmap">The bitmap you want to highlight the countours of</param>
        /// <returns>The bitmap with countours highlighted</returns>
        public static Bitmap HighlightContours(Bitmap inputBitmap)
        {
            Bitmap outputBitmap = new Bitmap(inputBitmap.Width, inputBitmap.Height);

            Bitmap grayscale = Grayscale(inputBitmap);
            Bitmap thresholded = Tresholding(grayscale, 128);
            Bitmap dilated = Dilatation(thresholded, 3);
            Bitmap eroded = Erode(dilated, 3);

            for (int y = 0; y < inputBitmap.Height; y++)
            {
                for (int x = 0; x < inputBitmap.Width; x++)
                {
                    Color pixel = inputBitmap.GetPixel(x, y);
                    Color dilatedPixel = dilated.GetPixel(x, y);
                    Color erodedPixel = eroded.GetPixel(x, y);

                    int gray = (int)(pixel.R * 0.3 + pixel.G * 0.59 + pixel.B * 0.11);
                    int threshold = dilatedPixel.R;

                    if (gray > threshold)
                    {
                        outputBitmap.SetPixel(x, y, Color.FromArgb(255, 255, 255));
                    }
                    else if (gray <= threshold && erodedPixel.R == 0)
                    {
                        outputBitmap.SetPixel(x, y, Color.FromArgb(255, 0, 0));
                    }
                    else
                    {
                        outputBitmap.SetPixel(x, y, Color.FromArgb(0, 0, 0));
                    }
                }
            }

            return outputBitmap;
        }
        /// <summary>
        /// Method that that erodes the morphology of a bitmap
        /// </summary>
        /// <param name="inputBitmap">The bitmap you want to erode</param>
        /// <param name="kernelSize">The amount of Erosion you want (be carefull its expensive on ressources)</param>
        /// <returns>The Bitmap with the eroded contents</returns>
        public static Bitmap Erode(Bitmap inputBitmap, int kernelSize)
        {
            Bitmap outputBitmap = new Bitmap(inputBitmap.Width, inputBitmap.Height);

            int[,] kernel = new int[kernelSize, kernelSize];

            for (int i = 0; i < kernelSize; i++)
            {
                for (int j = 0; j < kernelSize; j++)
                {
                    kernel[i, j] = 1;
                }
            }

            for (int y = kernelSize / 2; y < inputBitmap.Height - kernelSize / 2; y++)
            {
                for (int x = kernelSize / 2; x < inputBitmap.Width - kernelSize / 2; x++)
                {
                    bool flag = true;

                    for (int i = -kernelSize / 2; i <= kernelSize / 2; i++)
                    {
                        for (int j = -kernelSize / 2; j <= kernelSize / 2; j++)
                        {
                            Color pixel = inputBitmap.GetPixel(x + i, y + j);
                            int gray = (int)(pixel.R * 0.3 + pixel.G * 0.59 + pixel.B * 0.11);

                            if (gray >= 128 && kernel[i + kernelSize / 2, j + kernelSize / 2] == 1)
                            {
                                flag = false;
                                break;
                            }
                        }

                        if (!flag)
                        {
                            break;
                        }
```

```
                    }

                    if (flag)
                    {
                        outputBitmap.SetPixel(x, y, Color.FromArgb(255, 255, 255));
                    }
                    else
                    {
                        outputBitmap.SetPixel(x, y, Color.FromArgb(0, 0, 0));
                    }
                }
            }

            return outputBitmap;
        }
        /// <summary>
        /// Method that that use dilatation of the morphology of a bitmap
        /// </summary>
        /// <param name="inputBitmap">The bitmap you want to use dilatation on</param>
        /// <param name="kernelSize">The amount of dilatation you want (be carefull its expensive on ressources)</param>
        /// <returns>The Bitmap after Dilatation</returns>
        public static Bitmap Dilatation(Bitmap inputBitmap, int kernelSize)
        {
            Bitmap outputBitmap = new Bitmap(inputBitmap.Width, inputBitmap.Height);

            int[,] kernel = new int[kernelSize, kernelSize];

            for (int i = 0; i < kernelSize; i++)
            {
                for (int j = 0; j < kernelSize; j++)
                {
                    kernel[i, j] = 1;
                }
            }

            for (int y = kernelSize / 2; y < inputBitmap.Height - kernelSize / 2; y++)
            {
                for (int x = kernelSize / 2; x < inputBitmap.Width - kernelSize / 2; x++)
                {
                    bool flag = false;

                    for (int i = -kernelSize / 2; i <= kernelSize / 2; i++)
                    {
                        for (int j = -kernelSize / 2; j <= kernelSize / 2; j++)
                        {
                            Color pixel = inputBitmap.GetPixel(x + i, y + j);
                            int gray = (int)(pixel.R * 0.3 + pixel.G * 0.59 + pixel.B * 0.11);

                            if (gray < 128 && kernel[i + kernelSize / 2, j + kernelSize / 2] == 1)
                            {
                                flag = true;
                                break;
                            }
                        }

                        if (flag)
                        {
                            break;
                        }
                    }

                    if (flag)
                    {
                        outputBitmap.SetPixel(x, y, Color.FromArgb(0, 0, 0));
                    }
                    else
                    {
                        outputBitmap.SetPixel(x, y, Color.FromArgb(255, 255, 255));
                    }
                }
            }

            return outputBitmap;
        }
    }
}
```

4.17 Settings.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;

namespace Test_Merge
{
    public partial class Settings : Form
    {
        private string _grandPrixUrl = "";
        private string _grandPrixName = "";
        private int _grandPrixYear = 2000;
        private List<string> _driverList = new List<string>();

        private F1TVEmulator Emulator = null;
        private ConfigurationTool Config = null;

        private bool CreatingZone = false;
        private Point ZoneP1;
        private Point ZoneP2;

        private bool CreatingWindow = false;
        private Point WindowP1;
        private Point WindowP2;

        List<Rectangle> WindowsToAdd = new List<Rectangle>();

        public string GrandPrixUrl { get => _grandPrixUrl; private set => _grandPrixUrl = value; }
        public string GrandPrixName { get => _grandPrixName; private set => _grandPrixName = value; }
        public int GrandPrixYear { get => _grandPrixYear; private set => _grandPrixYear = value; }
        public List<string> DriverList { get => _driverList; private set => _driverList = value; }

        public Settings()
        {
            InitializeComponent();
            Load();
        }
        private void Load()
        {
            RefreshUI();
        }
        private void RefreshUI()
        {

            lsbDrivers.DataSource = null;
            lsbDrivers.DataSource = DriverList;

            if (Directory.Exists(ConfigurationTool.CONFIGS_FOLDER_NAME))
            {
                lsbPresets.DataSource = null;
                lsbPresets.DataSource = Directory.GetFiles(ConfigurationTool.CONFIGS_FOLDER_NAME);
            }
            if (CreatingZone)
            {
                if (ZoneP1 == new Point(-1, -1))
                {
                    lblZonePointsRemaning.Text = "2 points Remaining";
                }
                else
                {
                    lblZonePointsRemaning.Text = "1 point Remaining";
                }
            }
            else
            {
                lblZonePointsRemaning.Text = "";
            }

            if (CreatingWindow)
            {
                if (WindowP1 == new Point(-1, -1))
                {
                    lblWindowPointsRemaining.Text = "2 points Remaining";
                }
                else
                {
                    lblWindowPointsRemaining.Text = "1 point Remaining";
                }
                lblWindowPointsRemaining.Text = ConfigurationTool.NUMBER_OF_ZONES - WindowsToAdd.Count() + " Windows remaining";
            }
            else
            {
```

```
            lblWindowPointsRemaining.Text = "";
            lblWindowsRemaining.Text = "";
        }
        if (Config != null)
        {
            pbxMain.Image = Config.MainZone.Draw();
            if(Config.MainZone.Zones.Count > 0)
                pbxDriverZone.Image = Config.MainZone.Zones[0].Draw();
        }
    }
    private void CreateNewZone(Point p1, Point p2)
    {
        Rectangle dimensions = CreateAbsoluteRectangle(p1, p2);
        Config = new ConfigurationTool((Bitmap)pbxMain.Image, dimensions);
        RefreshUI();
    }
    private void CreateWindows(List<Rectangle> dimensions)
    {
        if (Config != null)
        {
            Config.AddWindows(dimensions);
        }
    }
    private void tbxGpUrl_TextChanged(object sender, EventArgs e)
    {
        GrandPrixUrl = tbxGpUrl.Text;
    }

    private void tbxGpName_TextChanged(object sender, EventArgs e)
    {
        GrandPrixName = tbxGpName.Text;
    }

    private void tbxGpYear_TextChanged(object sender, EventArgs e)
    {
        int year;
        try
        {
            year = Convert.ToInt32(tbxGpYear.Text);
        }
        catch
        {
            year = 1545;
        }
        GrandPrixYear = year;
    }

    private void btnAddDriver_Click(object sender, EventArgs e)
    {
        string newDriver = tbxDriverName.Text;
        DriverList.Add(newDriver);
        tbxDriverName.Text = "";
        RefreshUI();
    }

    private void btnRemoveDriver_Click(object sender, EventArgs e)
    {
        if (lsbDrivers.SelectedIndex >= 0)
        {
            DriverList.RemoveAt(lsbDrivers.SelectedIndex);
        }
        RefreshUI();
    }
    private void SwitchZoneCreation()
    {
        if (CreatingZone)
        {
            CreatingZone = false;
            lblZonePointsRemaning.Text = "";
        }
        else
        {
            CreatingZone = true;

            if (Config != null)
                Config.ResetMainZone();

            if (CreatingWindow)
                SwitchWindowCreation();

            if (Emulator != null && Emulator.Ready)
            {
                Config = null;
                pbxMain.Image = Emulator.Screenshot();
            }

            ZoneP1 = new Point(-1, -1);
            ZoneP2 = new Point(-1, -1);

            lblZonePointsRemaning.Text = "2 Points left";
        }
        RefreshUI();
    }
    private void SwitchWindowCreation()
```

```
        {
            if (CreatingWindow)
            {
                CreatingWindow = false;
            }
            else
            {
                CreatingWindow = true;

                if (Config != null)
                    Config.ResetWindows();

                if (CreatingZone)
                    SwitchZoneCreation();

                WindowP1 = new Point(-1, -1);
                WindowP2 = new Point(-1, -1);

                WindowsToAdd = new List<Rectangle>();
            }
            RefreshUI();
        }
        private void btnCreatZone_Click(object sender, EventArgs e)
        {
            SwitchZoneCreation();
        }
        private void btnCreateWindow_Click(object sender, EventArgs e)
        {
            SwitchWindowCreation();
        }
        private void pbxMain_MouseClick(object sender, MouseEventArgs e)
        {
            if (CreatingZone && pbxMain.Image != null)
            {
                //Point coordinates = pbxMain.PointToClient(new Point(MousePosition.X, MousePosition.Y));
                Point coordinates = e.Location;
                float xOffset = (float)pbxMain.Image.Width / (float)pbxMain.Width;
                float yOffset = (float)pbxMain.Image.Height / (float)pbxMain.Height;
                Point newPoint = new Point(Convert.ToInt32((float)coordinates.X * xOffset), Convert.ToInt32((float)coordinates.Y * yOffset));

                //MessageBox.Show("Coordinates" + Environment.NewLine + "Old : " + coordinates.ToString() + Environment.NewLine + "New : " +
newPoint.ToString());

                if (ZoneP1 == new Point(-1, -1))
                {
                    ZoneP1 = newPoint;
                }
                else
                {
                    ZoneP2 = newPoint;
                    CreateNewZone(ZoneP1, ZoneP2);
                    SwitchZoneCreation();
                }
                RefreshUI();
            }
        }
        private void pbxMain_Click(object sender, EventArgs e)
        {
            //Not the right one to use visibly
        }
        private void pbxDriverZone_MouseClick(object sender, MouseEventArgs e)
        {
            if (CreatingWindow && pbxDriverZone.Image != null)
            {
                Point coordinates = e.Location;

                float xOffset = (float)pbxDriverZone.Image.Width / (float)pbxDriverZone.Width;
                float yOffset = (float)pbxDriverZone.Image.Height / (float)pbxDriverZone.Height;

                Point newPoint = new Point(Convert.ToInt32((float)coordinates.X * xOffset), Convert.ToInt32((float)coordinates.Y * yOffset));

                if (WindowP1 == new Point(-1, -1))
                {
                    WindowP1 = newPoint;
                }
                else
                {
                    WindowP2 = newPoint;
                    WindowsToAdd.Add(CreateAbsoluteRectangle(WindowP1, WindowP2));

                    if (WindowsToAdd.Count < ConfigurationTool.NUMBER_OF_ZONES)
                    {
                        WindowP1 = new Point(-1, -1);
                        WindowP2 = new Point(-1, -1);
                    }
                    else
                    {
                        WindowP1 = new Point(WindowP1.X, 0);
                        WindowP2 = new Point(WindowP2.X, pbxDriverZone.Image.Height);
                        CreateWindows(WindowsToAdd);
                        SwitchWindowCreation();
                    }
                }
                RefreshUI();
```

```
            }
        }
        private void pbxDriverZone_Click(object sender, EventArgs e)
        {
            //Not the right one to use visibly
        }
        private Rectangle CreateAbsoluteRectangle(Point p1, Point p2)
        {
            Point newP1 = new Point();
            Point newP2 = new Point();

            if (p1.X < p2.X)
            {
                newP1.X = p1.X;
                newP2.X = p2.X;
            }
            else
            {
                newP1.X = p2.X;
                newP2.X = p1.X;
            }

            if (p1.Y < p2.Y)
            {
                newP1.Y = p1.Y;
                newP2.Y = p2.Y;
            }
            else
            {
                newP1.Y = p2.Y;
                newP2.Y = p1.Y;
            }
            return new Rectangle(newP1.X, newP1.Y, newP2.X - newP1.X, newP2.Y - newP1.Y);
        }

        private async void btnRefresh_Click(object sender, EventArgs e)
        {
            btnRefresh.Enabled = false;
            if (Emulator == null || Emulator.GrandPrixUrl != tbxGpUrl.Text)
            {
                Emulator = new F1TVEmulator(tbxGpUrl.Text);
            }

            if (!Emulator.Ready)
            {
                Task<int> start = Task.Run(() => Emulator.Start());
                int errorCode = await start;
                if (errorCode != 0)
                {
                    string message;
                    switch (errorCode)
                    {
                        case 101:
                            message = "Error " + errorCode + " Could not start the driver. It could be because an other instance is runnin make sure you closed
them all before trying again";
                            break;
                        case 102:
                            message = "Error " + errorCode + " Could not navigate on the F1TV site. Make sure the correct URL has been given and that you
logged from chrome. It can take a few minutes to update";
                            break;
                        case 103:
                            message = "Error " + errorCode + " The url is not a valid url";
                            break;
                        case 104:
                            message = "Error " + errorCode + " The url is not a valid url";
                            break;
                        case 105:
                            message = "Error " + errorCode + " There has been an error trying to emulate button presses. Please try again";
                            break;
                        case 106:
                            message = "Error " + errorCode + " There has been an error trying to emulate button presses. Please try again";
                            break;
                        default:
                            message = "Could not start the emulator Error " + errorCode;
                            break;
                    }
                    MessageBox.Show(message);
                }
                else
                {
                    pbxMain.Image = Emulator.Screenshot();
                }
            }
            else
            {
                pbxMain.Image = Emulator.Screenshot();
            }
            btnRefresh.Enabled = true;
        }

        private void Settings_FormClosing(object sender, FormClosingEventArgs e)
        {
            if (Emulator != null)
            {
```

```
                Emulator.Stop();
            }
        }

        private void btnResetDriver_Click(object sender, EventArgs e)
        {
            if (Emulator != null)
            {
                Emulator.ResetDriver();
            }
        }

        private void btnSavePreset_Click(object sender, EventArgs e)
        {
            string presetName = tbxPresetName.Text;
            if (Config != null)
            {
                Config.SaveToJson(DriverList,presetName);
            }
            RefreshUI();
        }

        private void lsbPresets_SelectedIndexChanged(object sender, EventArgs e)
        {
            //Nothing
        }

        private void btnLoadPreset_Click(object sender, EventArgs e)
        {
            if (lsbPresets.SelectedIndex >= 0 && pbxMain.Image != null)
            {
                try
                {
                    Reader reader = new Reader(lsbPresets.Items[lsbPresets.SelectedIndex].ToString(), (Bitmap)pbxMain.Image,false);
                    //MainZones #0 is the big main zone containing driver zones
                    Config = new ConfigurationTool((Bitmap)pbxMain.Image, reader.MainZones[0].Bounds);
                    Config.MainZone = reader.MainZones[0];
                    DriverList = reader.Drivers;
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Could not load the settings error :" + ex);
                }
                RefreshUI();
            }
        }
    }
}
```

4.18 recoverCookiesCSV.py

```
# Rohmer Maxime
# RecoverCookies.py
# Little script that recovers the cookies stored in the chrome sqlite database and then decrypts them using the key stored in the chrome files
# This script has been created to be used by an other programm or for the data to not be used directly. This is why it stores all the decoded cookies in a csv.
(Btw could be smart for the end programm to delete the csv after using it)
# Parts of this cript have been created with the help of ChatGPT

import os
import json
import base64
import sqlite3
import win32crypt
from Cryptodome.Cipher import AES
from pathlib import Path
import csv

def get_master_key():
    with open(
        os.getenv("localappdata") + "\\Google\\Chrome\\User Data\\Local State", "r"
    ) as f:
        local_state = f.read()
        local_state = json.loads(local_state)
    master_key = base64.b64decode(local_state["os_crypt"]["encrypted_key"])
    master_key = master_key[5:]  # removing DPAPI
    master_key = win32crypt.CryptUnprotectData(master_key, None, None, None, 0)[1]
    print("MASTER KEY :")
    print(master_key)
    print(len(master_key))
    return master_key

def decrypt_payload(cipher, payload):
    return cipher.decrypt(payload)

def generate_cipher(aes_key, iv):
    return AES.new(aes_key, AES.MODE_GCM, iv)

def decrypt_password(buff, master_key):
    try:
        iv = buff[3:15]
        payload = buff[15:]
        cipher = generate_cipher(master_key, iv)
        decrypted_pass = decrypt_payload(cipher, payload)
        decrypted_pass = decrypted_pass[:-16].decode()  # remove suffix bytes
        return decrypted_pass
    except Exception:
        # print("Probably saved password from Chrome version older than v80\n")
        # print(str(e))
        return "Chrome < 80"


master_key = get_master_key()

cookies_path = Path(
    os.getenv("localappdata") + "\\Google\\Chrome\\User Data\\Default\\Network\\Cookies"
)

if not cookies_path.exists():
    raise ValueError("Cookies file not found")

with sqlite3.connect(cookies_path) as connection:
    connection.row_factory = sqlite3.Row
    cursor = connection.cursor()
    cursor.execute("SELECT * FROM cookies")

    with open('cookies.csv', 'a', newline='') as csvfile:
        fieldnames = ['host_key', 'name', 'value', 'path', 'expires_utc', 'is_secure', 'is_httponly']
        writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

        if csvfile.tell() == 0:
            writer.writeheader()

        for row in cursor.fetchall():
            decrypted_value = decrypt_password(row["encrypted_value"], master_key)
            writer.writerow({
                'host_key': row["host_key"],
                'name': row["name"],
                'value': decrypted_value,
                'path': row["path"],
                'expires_utc': row["expires_utc"],
                'is_secure': row["is_secure"],
                'is_httponly': row["is_httponly"]
            })

print("Finished CSV")
```