

Use Case : Serli

Approche : Se servir des contours du circuit pour déterminer la position du kart. 1

Etape 1 : Récupérer les contour du circuit à partir des images : 1

Etape 2 : Comparer la version du circuit trouvée au circuit vue de dessus : 6

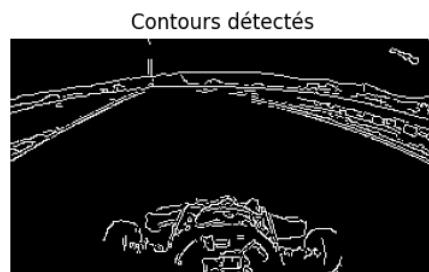
Etape 3 : Déterminer la position de chaque image en utilisant les étapes précédentes : 7

Approche : Se servir des contours du circuit pour déterminer la position du kart.

Etape 1 : Récupérer les contour du circuit à partir des images :



Image originale



Lundi : 20/01

Pour réduire la taille des données en entrées nous avons découpé la vidéo en image de façon suivante : on alterne 12 ignorées, et 4 récupérées.

Nous utilisons OpenCV avec la fonction Canny, pour chaque image, on récupère une image avec un fond noir et en blanc les différents contour détecter.

Mardi : 21/01

Il nous faut encore traiter ces données pour ne récupérer que les bords de piste et la position de la caméra par rapport au bord, pour cela on a fait du flou et redessiné par dessus afin de ne garder que les lignes et les courbes (utilisation d'une Hystérésis), puis ajouter le calcul de l'empreinte carbone. De plus, nous allons aussi récupérer la ligne d'arrivée afin de calibrer chaque tour.

Segmentation d'images avec scikit-image (non poursuivi) :

Dans un premier temps nous avons utilisé Canny pour tracer les contours de piste. Mais cela n'était peut être pas la meilleure solution pour le projet. Alors pour remédier à cela, nous avons testé en parallèle l'obtention de contour grâce à la segmentation d'image donnée par la library scikit-images.

On obtient la segmentation ci dessous :



Image de base

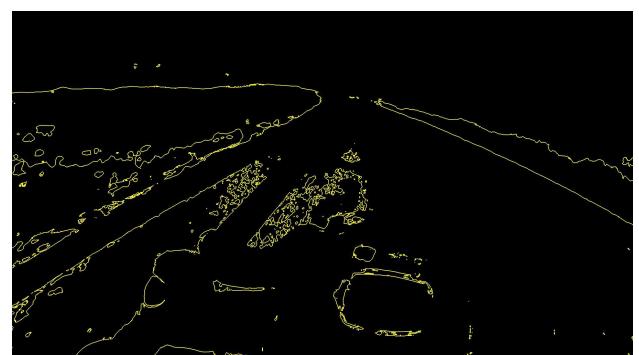


Image segmentée avec Scikit

Malheureusement, en raison d'un manque de temps pour approfondir la recherche de solutions, nous n'avons pas pu explorer toutes les options disponibles dans scikit-image. Bien que la méthode de Canny ait permis de détecter certains contours, nous avons constaté que la segmentation obtenue avec scikit-image présentait des

résultats intéressants, mais nécessitent encore des ajustements pour améliorer la précision et la qualité des segments.

mercredi 22/01

Nous avons fait un réseau de neurones afin de savoir où est le kart sur la piste (hors de la piste, à gauche, à droite, au milieu). Pour cela nous avons pris un ensemble d'images (70% environ) tirées de la vidéo target. Nous avons créé un fichier d'entraînement csv dans lequel nous avons manuellement écrit si le kart était à gauche, droite, centre ou hors de la piste. Le fichier rempli est de la forme suivante:

```
name, class  
frame_0012.jpg, 1  
frame_0013.jpg, 1  
frame_0076.jpg, 2  
frame_0077.jpg, 2  
frame_0079.jpg, 2  
frame_0143.jpg, 2  
frame_0174.jpg, 3  
frame_0175.jpg, 3  
frame_0188.jpg, 3  
frame_0189.jpg, 3  
frame_0190.jpg, 2  
frame_0191.jpg, 2  
frame_0204.jpg, 1  
frame_0205.jpg, 1  
frame_0206.jpg, 1  
frame_0207.jpg, 1  
frame_0220.jpg, 1  
frame_0221.jpg, 1  
frame_0222.jpg, 2  
frame_0223.jpg, 2  
frame_0236.jpg, 3  
frame_0237.jpg, 3
```

Nous avons décidé que pour les labels seraient 0, 1, 2 ou 3 pour, respectivement, un kart hors piste, un kart situé sur le bord gauche de la piste, au centre ou sur le bord droit. Nous avons ensuite créé un modèle de classification en utilisant l'API Keras de Tensorflow que nous avons entraîné sur nos images. En le testant ensuite sur les images que nous obtenons avec Canny on observe une précision d'environ 95%.

Nous voulons reconstituer le tracé du circuit vu dans la vision satellite à partir des frames de la vidéo. Nous ajoutons des repères visibles (arbres, bâtiments, bords de route) afin de pouvoir reconstituer le circuit.

Puis nous utiliserons la fonction permettant de convertir des coordonnées GPS en position X,Y pour dessiner le tracé du kart sur le circuit, vue de dessus.

Après quelques tests, nous avons adopté une nouvelle stratégie pour modéliser le circuit en vision satellite. Nous mettons des points d'intérêts loin sur chaque image et nous regardons le rapprochement du point entre deux images, pour cela nous utilisons **ORB**.

ORB (Oriented Fast and Rotated BRIEF) : ORB est un algorithme utilisé pour la détection et la description de points clés dans une image. Il détecte les points clés avec FAST, les points clés repérés sont des coins (les pixels sont analysés pour repérer les coins). Ensuite il va faire une description des points avec BRIEF amélioré pour prendre en compte les rotations. Il crée un descripteur binaire qui compare l'intensité des pixels autour du points d'intérêt

Nous avons ajouté des filtres sur le canny de la vidéo afin d'augmenter la couleur blanche et ajuster le contraste pour faire ressortir les bords de la piste

jeudi 23/01

Il faut améliorer l'image faite avec canny afin d'avoir seulement les lignes blanches du bord du circuit. Avec ces images nous allons pouvoir insérer les points d'intérêts sur les lignes blanches afin de reproduire le circuit.

Liste de tests effectués :

Partie filtre et ajustement sur le canny

- **Blur** pour ajouter un flou Gaussien sur la vidéo pour réduire le bruit autour du kart -> résultat très concluant ;
- **Ajustement** du contraste et de la luminosité pour influencer la capture d'information au niveau des lignes blanches -> résultat amélioré légèrement ;

- **HSV** -> Détection par segmentation des couleurs en isolant les couleurs blanches avec un masque -> résultat positif ;
- **Hough** pour influencer les traits : Rendu illisible dû à l'amas de trait résultats -> résultat médiocre ;
- **Filtre la circularités** pour tenter de supprimer les formes circulaire comme kart ou arrière-plan sur l'image : fonction `cv2.arcLength(..)` -> résultat mauvais, ce n'est pas exploitable ;
- **Skimage filters** (Frangi, sato, hessian,..) : Rendu en 3 images par seconde car les algorithmes sont beaucoup trop lourd, le rendu est de plus illisible -> résultat mauvais ;
- **Active Contours/snakes** : aucun résultat, aucun suivi des lignes ni rendu correcte ;
- **Morphologie mathématique** : Utilisé pour réduire les points lumineux dû au soleil et 'fermer' les trous dans des lignes discontinues -> Résultat plutôt satisfaisant ;

Test du **crop** de la vidéo pour limiter la prise d'information inutile : suppression du bas de la vidéo pour enlever au mieux le kart et le bruit -> résultat plutôt positif, tentative de combinaison avec les KeyPoints ;



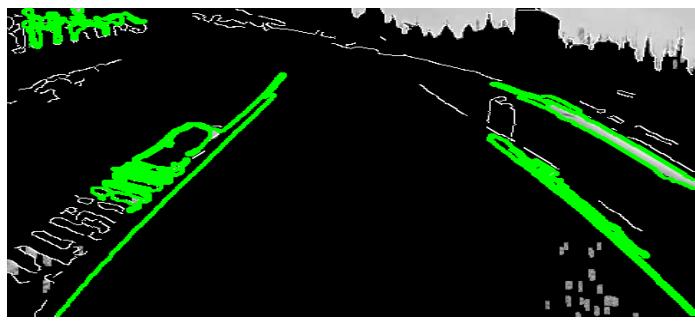
Vidéo originale mise en noir et blanc et coupée de façon à garder que le circuit



Vidéo avec filtre Blur



Canny conçu par rapport à la vidéo avec les techniques HSV et Morphologie mathématique

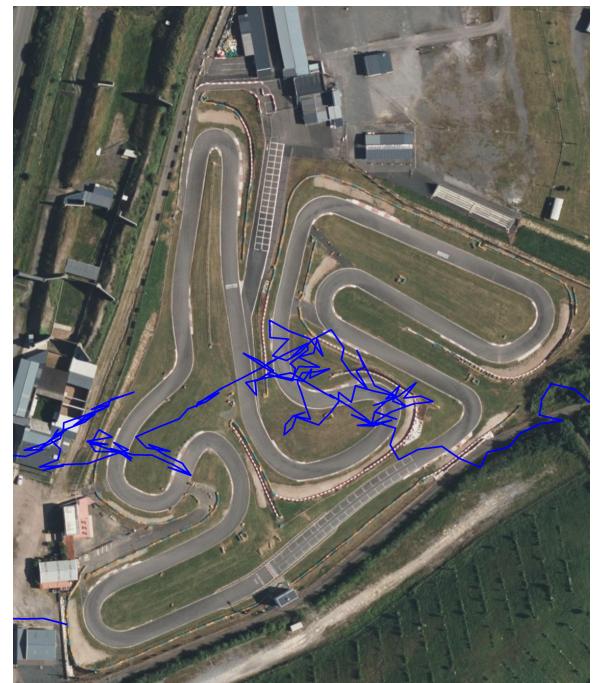


Ajout du Find_Contours sur le canny

Etape 2 : Comparer la version du circuit trouvée au circuit vue de dessus :

Génération du circuit à l'aide de ORB pour placer des keypoints sur une image et les rechercher sur l'image suivante. De cette manière on calcule la rotation et la translation effectuée entre deux images.

Pour cette étape il faut comparer le circuit donné avec celui créé grâce aux images analysées en les superposant.



On peut voir sur les figures précédentes les circuits obtenus avec notre algorithme superposés sur l'image du circuit réel. Nous avons des résultats assez décevants pour le moment. Cependant, quand les résultats correspondront mieux au visuel de référence nous pourrons utiliser les coordonnées des débuts de secteurs disponibles

dans le fichier world.json afin de voir si on peut superposer et faire correspondre notre résultat au circuit.

Etape 3 : Déterminer la position de chaque image en utilisant les étapes précédentes :

Par manque de temps nous ne pourrons pas implémenter cette étape. Cependant nous avons réfléchi à ce que nous aurions dû faire.

Après avoir associé l'image du circuit en version satellite et notre circuit reconstruit, nous allons devoir reconnaître les positions de chaque image de la vidéo par rapport au circuit. Nous pourrons ensuite placer un point sur le circuit représentant le kart et tracer la courbe de trajectoire du pilote sur la piste.