Guide d’utilisation

Février 2023

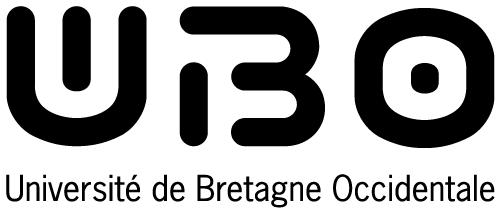
Par Tristan LE PAPE

Dans le cadre de L’UE Mise en Pratique Accompagné

Pour MerConcept

Tuteur UBO : Jean-Philippe Babau

Tuteur MerConcept : Yann Le Norment



Toute la configuration se passe dans le constructeur de la classe Main

Exemple de configuration en vidéo : <https://youtu.be/3LBw02-P4Ow>

# Chargement d’une vidéo

Par défaut, le programme charge la vidéo portant le nom ccc.mkv dans le dossier video

self.cap = cv2.VideoCapture(os.path.dirname(\_\_file\_\_) + "/video/ccc.mkv")

On peut charger plusieurs vidéos. Par exemple, on peut charger les 2 flux rtsp de cette manière :

self.cap = cv2.VideoCapture("rtsp://root:M101\_svr@192.168.1.56:554/axis-media/media.amp")

self.cap2 = cv2.VideoCapture("rtsp://root:M101\_svr@192.168.1.55:554/axis-media/media.amp")

# Création des zones d’analyses

Il existe 3 types de zones (safran, mousse, dérive) pour lesquelles il faut obtenir les coordonnées.

## 2.1 Obtention d’une image de la vidéo

Pour obtenir une image de la vidéo à traiter on peut simplement lancer le programme main.py avec une vidéo et appuyer sur **ctrl** + **s** au moment où on veut enregistrer.

## 2.2 Obtention des coordonnées des zones d’analyse

L’analyse s’appuie sur des zones bien spécifiques de l’image, zones rectangles qu’il faut identifier à l’aide de deux points. Le premier point est situé en **haut à gauche** et en deuxième point en **bas à droite** du rectangle.

Lancer le programme setup.py et choisir une image.

Clic gauche -> Placer un point

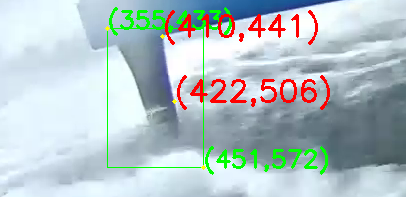
Maintenir le clic gauche -> Tracer un rectangle

Touche r -> Effacer tous les tracés

Touche Echap -> Quitter le programme

A l’aide des commandes ci-dessus vous pouvez obtenir toutes les coordonnées nécessaires.

**Pour le safran :**

****

En plus de la zone à analyser (rectangle vert caractérisé par le point PtHg et PtBd), le safran nécessite de renseigner un premier point (Point1) tout en haut du safran et un deuxième point (Point2 ) n’importe où sur la tranche du safran (coordonnées rouges)

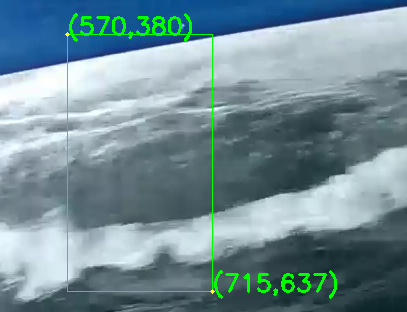
A partir des coordonnées, on peut paramétrer l'objet dans le programme :

AnalyseSafran(xPtHg, yPtHg, xPtBd, yPtBd, xPoint1, yPoint1, xPoint2, yPoint2, limiteDeQualité)

La limiteDeQualité est un entier à partir duquel toute valeur qui a un indice qualité inférieur à cette limite est considérée comme fausse.

AnalyseSafran(355, 433, 451, 572, 410, 441, 422, 506, 21)

**Pour la mousse :**

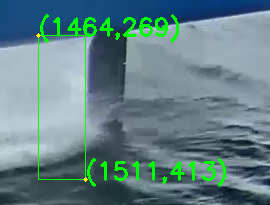
****

Pour la “mousse”, il faut fournir la zone à analyser (rectangle vert caractérisé par le point PtHg et PtBd). Le rectangle doit être choisi afin que son coin en haut à droite soit à la limite de la coque.

AnalyseMousse(xPtHg, yPtHg, xPtBd, yPtBd,limiteDeQualité)

AnalyseMousse(570, 380, 715, 637, 31)

**Pour la dérive :**

****

Pour la mesure de la hauteur de la dérive il faut sélectionner la zone juste derrière la dérive, il ne faut pas que la dérive soit incluse dedans. Il faut aussi que le point en haut à droite soit tout en haut de la dérive.

AnalyseMousse(xPtHg, yPtHg, xPtBd, yPtBd, limiteDeQualité)

AnalyseMousse(1464, 269, 1511, 413, 27)

Au final, on a une liste de zones dans le constructeur de la classe main comme pour cet exemple :

self.analysesBabord = {}

self.analysesBabord["derive"] = AnalyseDerive(1464, 269, 1511, 413)

self.analysesBabord["safran"] = AnalyseSafran(1234, 537, 1337, 669, 1315, 550, 1262, 645)

self.analysesBabord["mousse"] = AnalyseMousse(1170, 549, 1218, 665)

Il est possible d’avoir plusieurs listes de zones pour plusieurs vidéos. Par contre, seuls les objets d'analyse Derive, Safran et Mousse sont fournis par l’outil.

# Création des threads

Pour finir, il faut créer le ou les objets AnalyseVideo de type threads. On doit créer autant d’objet Analysevideo qu’il y a de vidéo à analyser.

AnalyseVideo prend en paramètre l’objet vidéo, la liste des zones à analyser sur cette vidéo et un nom qui est utilisé comme identifiant.

Par exemple pour deux vidéos self.cap et self.cap2, on a :

self.analyseVideoList = []

self.analyseVideoList.append(AnalyseVideo(self.cap, self.analysesTribord, "Tribord"))

self.analyseVideoList.append(AnalyseVideo(self.cap2, self.analysesBabord, "Babord"))

# 4. Les données

## 4.1 Traitement des données

Les données sont traitées dans la partie ###### Traitement des données ##### du fichier Main.py

On y retrouve 3 traitements qui peuvent bien sûr être modifiés et/ou commentés.

Traitement qui supprime les lignes qui contiennent des données manquantes.

dataRecovery.data[aMeasureKey] = dataRecovery.data[aMeasureKey].dropna()

Traitement qui supprime les lignes pour lesquelles l’indice qualité est inférieur à la limite fixée dans les paramètres des classes d’analyses.

dataRecovery.data[aMeasureKey] = dataRecovery.data[aMeasureKey].loc[dataRecovery.data[aMeasureKey]["quality"] > self.analyses[aMeasureKey].qualityLimit]

Une moyenne glissante, ici sur 25 valeurs.

dataRecovery.data[aMeasureKey]["height"] = uniform\_filter1d(dataRecovery.data[aMeasureKey]["height"].values.tolist(), size=25)

## 4.2 Affichage et enregistrement des données

L’affichage et l’enregistrement des données se fait dans la partie ###### Affichage des résultats ###### du fichier Main.py

A la fin de l’analyse des vidéos, le programme affiche, dans le navigateur par défaut, les courbes des hauteurs en premier et les courbes des indices de qualité en deuxième.

Toutes les données brutes sont enregistrées dans des fichiers excel, dans le dossier files. Attention, les anciennes données sont écrasées par les nouvelles données.

# Enregistrements

Si vous voulez enregistrer la vidéo, il suffit de se rendre dans le constructeur de la classe AnalyseVideo et le mettre le booléen self.record à true. L’enregistrement est désactivé par défaut.

Activer l’enregistrement : self.record = True

Désactiver l’enregistrement : self.record = False