Rapport de projet MPA

Estimation de la hauteur de vol d’un trimaran ultime par analyse d’image

Mars 2023

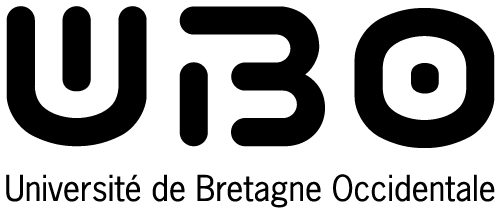
Par Tristan LE PAPE

Dans le cadre de L’UE Mise en Pratique Accompagné

Pour MerConcept

Tuteur UBO : Jean-Philippe Babau

Tuteur MerConcept : Yann Le Norment



# Introduction

La course au large est un sport qui met à l'épreuve les compétences des skippers et des bateaux dans des conditions météorologiques extrêmes. Les bateaux de course au large doivent être rapides, stables, résistants et équipés des dernières technologies pour naviguer en toute sécurité dans les mers et les océans du monde entier. Les voiliers de courses modernes possèdent des foils qui sont des appendices situés sous la coque qui permettent au bateau de se soulever hors de l'eau, ce système augmente considérablement la vitesse de navigation en minimisant la résistance de l'eau.

Les voiliers de compétition sont équipés de nombreux capteurs permettant le suivi et l’analyse des performances du bateau. Le trimaran SVR-Lazartigue de la classe Ultime possède deux caméras, placées sur les deux flotteurs bâbord et tribord, qui filment en permanence la coque vue de profil. Ce projet a pour but d’évaluer la pertinence d’utiliser ces images pour obtenir des informations sur la hauteur de vol du bateau et l’état de la mer.

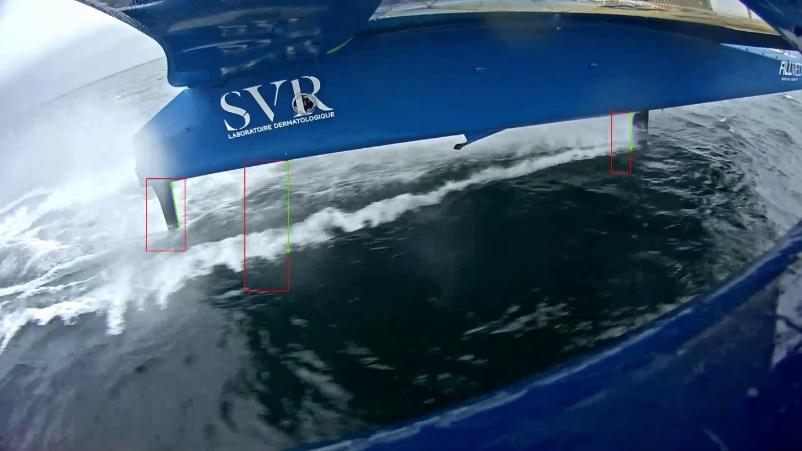
Le développement s'est fait sous Windows 11 avec Python 3.10 [1] sur un ordinateur portable avec la configuration matérielle suivante : 16 Go de Ram, CPU Intel Core i7-8750h, GPU Nvidia GeForce GTX 1060. Le programme utilise plusieurs bibliothèques dont la gestion est effectuée avec le gestionnaire de paquet pip, voici la liste des bibliothèques :

* OpenCV [2], Bibliothèque de traitement d’image.
* SciPy [3], Bibliothèque de traitement de données.
* Plotly [4], Bibliothèque de visualisation de données.
* NumPy [5], Bibliothèque de manipulation de tableaux.
* Pandas [6], Bibliothèque de manipulation de données (dataframe).

# Développement

La première étape du projet est de trouver la bonne bibliothèque de traitement d’image adaptée à la problématique. La bibliothèque retenue est OpenCV [2] pour plusieurs raisons. C’est une bibliothèque open source, rapide et pensée pour le traitement en temps réel, développée en C++. C’est la librairie de traitement d’image la plus utilisée, il existe énormément de documentations et tutoriels, notamment pour trouver des contours dans une image [7].

Pour pouvoir mesurer la hauteur de vol du trimaran il faut repérer qu’est ce qui pourrait être mesuré pour retrouver la hauteur. Les appendices de la coque sont pour cela idéal. Il existe 3 appendices sous la coque, le safran, le bras moteur, et la dérive. Le bras moteur étant trop petit, il est inutile. Un autre élément qui change quand le bateau vole, est l’augmentation de la mousse visible en dessous de la coque. On a donc 3 éléments mesurables qui permettent de retrouver l’évolution de la hauteur du bateau, le safran, la dérive et la mousse.



On peut observer sur cette image la vision de la caméra située sur le flotteur tribord, avec les zones analysées en rouge et la distance mesurée en vert.

Pour augmenter la rapidité de l’analyse, l’image globale n’est pas analysée, seulement les zones qui correspondent à nos 3 éléments mesurables. Ces zones sont définies manuellement par l’utilisateur avant de lancer le programme. Il existe 2 types de programme en fonction de la zone analysée, la classe « AnalyseMousse » qui utilise la mousse pour mesurer la hauteur et la classe « AnalysesSafran » qui utilise le safran pour mesurer la hauteur. Dans les deux cas, pour pouvoir utiliser l’algorithme FindContour d’OpenCV et mesurer les éléments intéressants dans les zones, le programme exécute une suite de traitements pour transformer l’image.

Les traitements, dans l’ordre, sont les suivants :

### Le filtre Noir et Blanc est le tout premier filtre utilisé, la majorité les algorithmes de traitement d’image travails sur des images en noir et blanc, ce format permet un traitement plus rapide.

### Le flou gaussien est utilisé pour réduire le bruit, les petits détails de l'image sont lissés et il ne reste plus que les aspects structurels de l'image, le floutage est très important pour améliorer les algorithmes de seuillage et de détection de bords.

### Transformer l’image noir et blanc en image binaire composé uniquement de pixel d’une valeur 0 (noir) ou 255 (blanc). Pour cela on peut utiliser deux algorithmes différents. Dans les deux cas, les paramètres sont évalués automatiquement en fonction de la moyenne de la valeur des pixels de la zone.

### 3.1) Le seuillage, utilisé pour la mesure de la mousse, il prend en paramètre un entier et transforme tous les pixels plus élevés en blanc et tous les pixels moins élevés en noir.

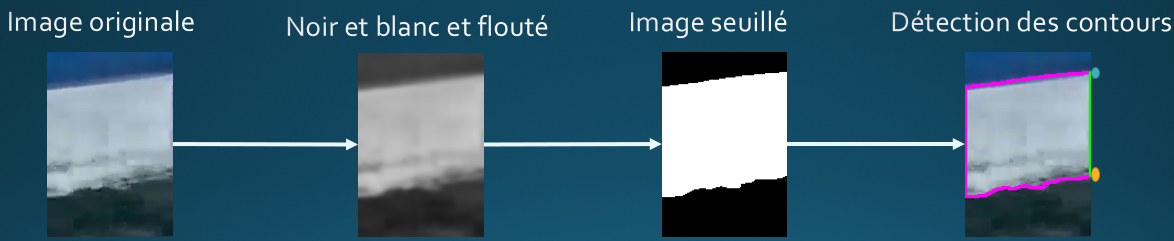
3.2) La détection de bords Canny, utilisé pour la mesure du safran, il prend en paramètres deux entiers, un seuil supérieur et un seuil inférieur, tous les pixels au-dessus du seuil supérieur sont considérés comme un bord. Tous les pixels en dessous du seuil inférieur ne sont pas pris en compte, et enfin tous les pixels entre les deux valeurs sont considérés comme un bord s’ils touchent un pixel au-dessus de la limite supérieure.

### Au final on utilise l’algorithme d’OpenCV FindContour(), qui va retourner la liste des coordonnées des contours trouvés dans l’image binaire.

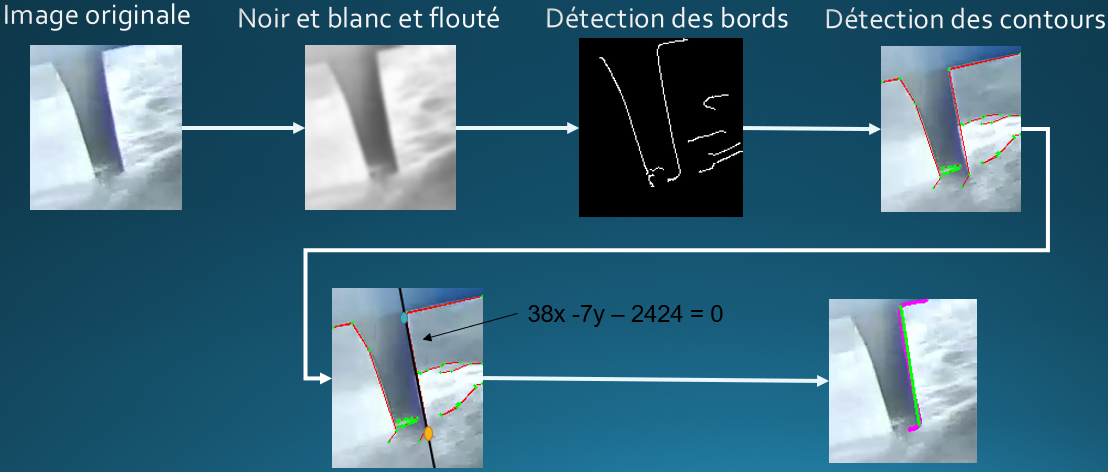
Pour le calcul de la distance, les deux classes « AnalyseMousse » et « AnalyseSafran » ont des méthodes différentes.

Pour trouver la hauteur de la mousse, le programme va chercher le point du contour le plus bas sur le bord droit de la zone. Il suffit ensuite de calculer la distance en pixel entre ce point et le point situé juste en dessous de la coque, donné manuellement par l’utilisateur.

La mesure de la hauteur du safran est différente car l’utilisateur va donner manuellement 2 points, le premier correspondant au début du safran et le second n’importe ou sur la tranche du safran, ces deux points vont permettre de calculer l’équation de la droite du safran et garder uniquement les points du contour aligné avec le safran.

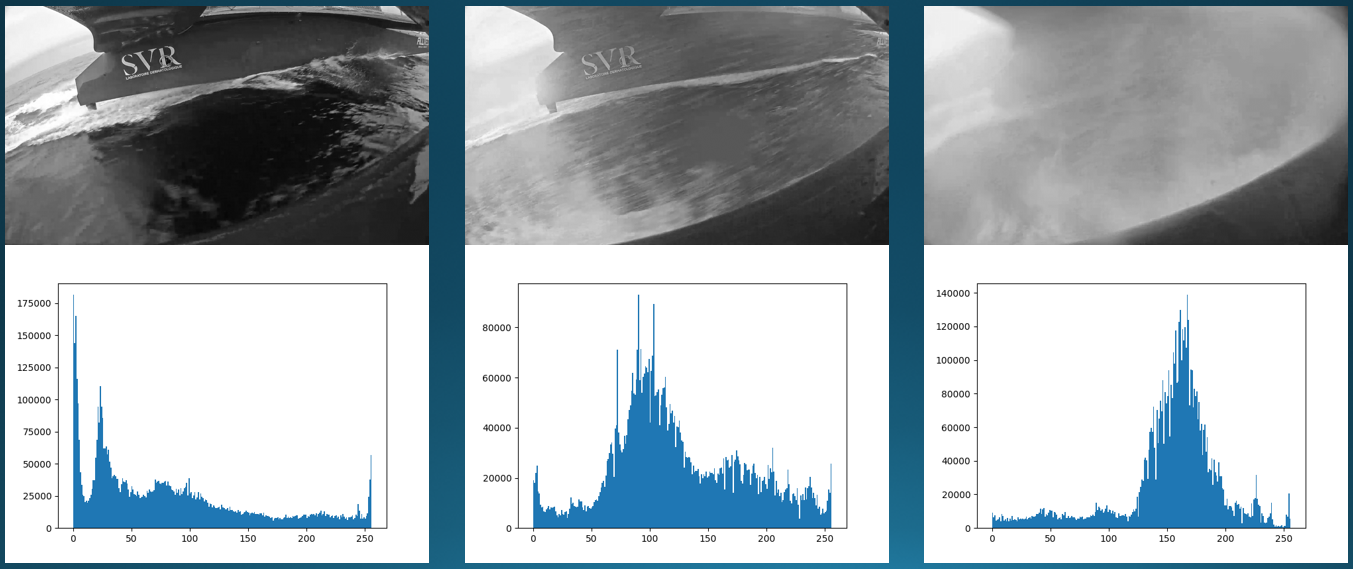


Sur le schéma ci-dessus on peut voir un exemple des différentes étapes appliquées à la zone située juste derrière la dérive, pour mesurer la dérive on mesure la mousse générée par la dérive.



Sur le schéma ci-dessus on peut voir un exemple des différentes étapes appliquées au safran.

Parfois la caméra est cachée par de l’embrun qui passe devant et il est impossible de faire une mesure, pour détecter cet événement, on calcul l’écart type de l’histogramme de la valeur des pixels. Plus l’écart type est faible, plus l’image est uniforme et donc la probabilité qu’elle soit masquée par de l’embrun est élevée. Cette valeur est utilisée comme indice qualité.



Comme on peut le voir sur ce schéma, on a 3 images avec leurs histogrammes en dessous qui indique le nombre de pixels pour chaque valeur, de 0 (noir) à 255 (blanc). De la gauche vers la droite, l’image devient de plus en plus masquée par l’embrun, et on observe l’histogramme se resserrer car l’image devient plus uniforme.

Pour pouvoir tester le programme dans plusieurs conditions et tester avec un flux en direct des caméras, une maquette à l’envers du bateau a été construite. Comme on peut le voir ci-dessous, la première image montre l’installation composée de la maquette, deux caméras, deux serveurs, un switch, et une alimentation. Sur la deuxième image on observe la simulation de la mer en faisant se déplacer une feuille bleue de haut en bas et de la droite vers la gauche.

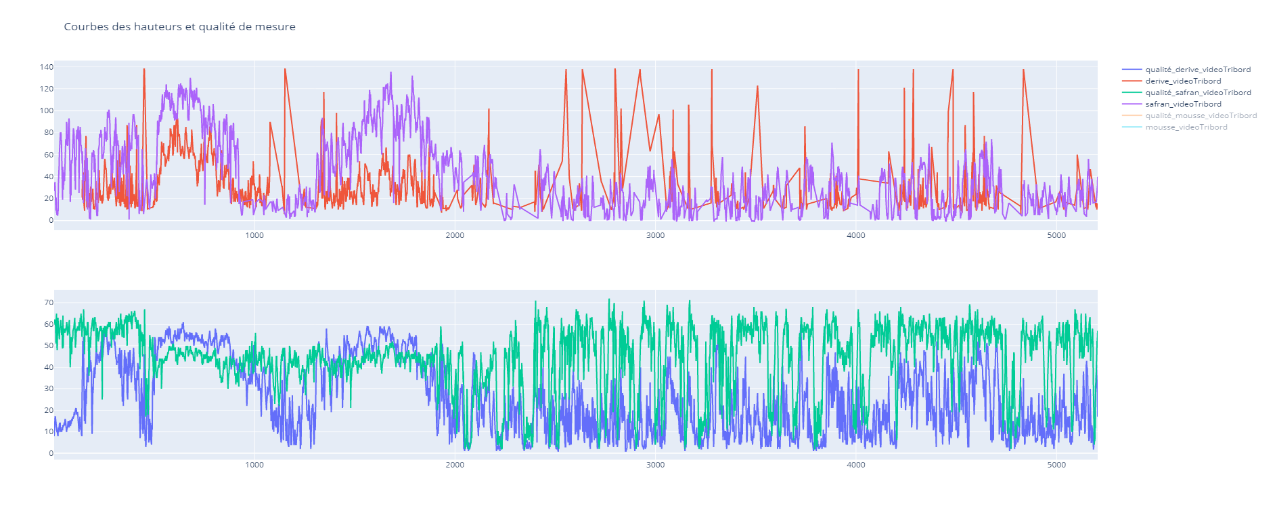
 

# Résultats

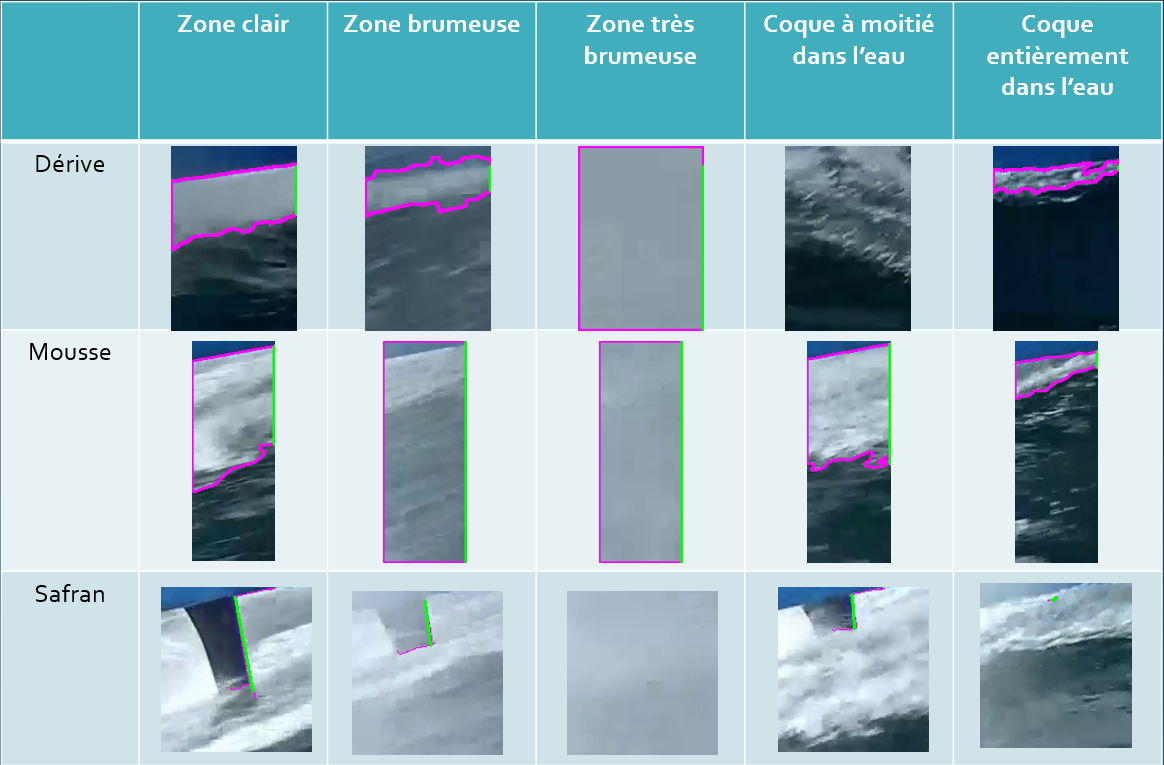
Après avoir configuré les zones d’analyse l’utilisateur lance le programme à partir de la classe Main, image par image les zones sont analysées et les résultats enregistrés. Pendant l’exécution toutes les images sont affichées avec les dessins des mesures, le contour détecté en rose et la hauteur en vert. Plusieurs informations sont aussi indiquées, le numéro de l’image, les indices qualité et le nombre d’image stocké dans le buffer, comme visible sur l’image ci-dessous.



A la fin du programme les courbes de hauteurs et d’indice qualités sont affichés.

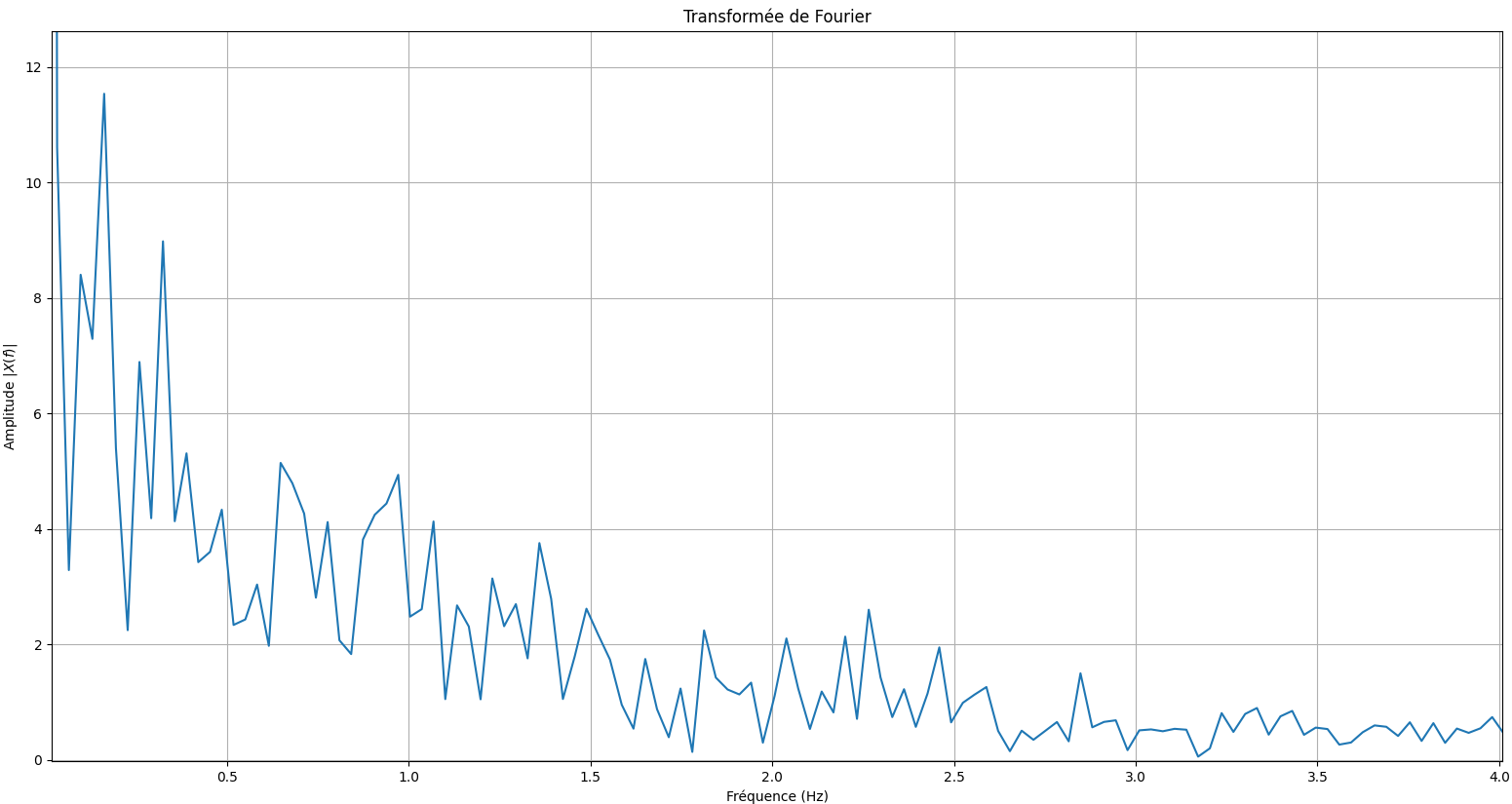
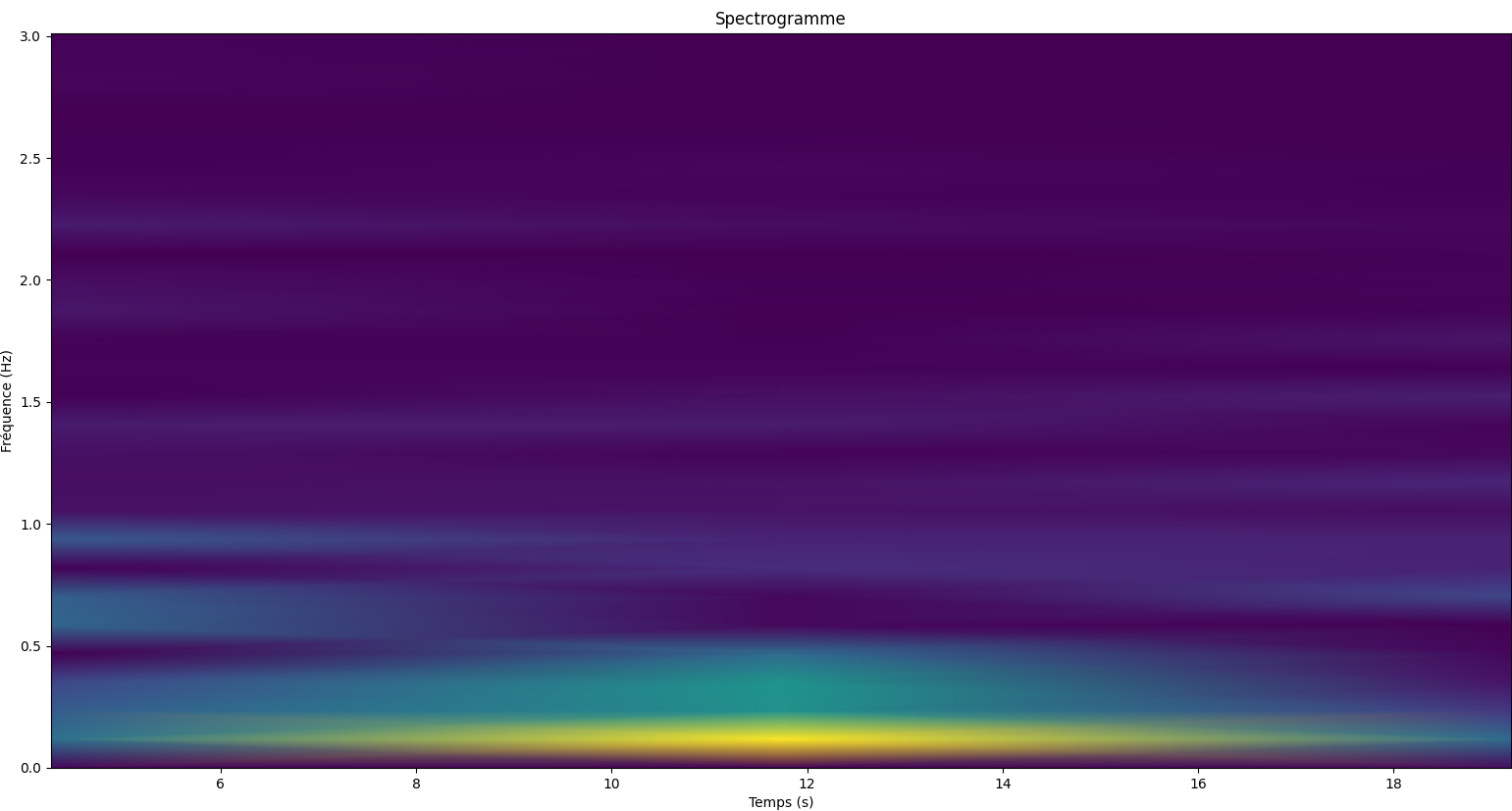


L’image ci-dessus montre les courbes de hauteurs (en haut) ainsi que les courbes d’indices qualités (en bas) mesuré au niveau du safran (violet) et de la dérive (rouge). On remarque que ces courbes suivent à peu près la même forme avec moins d’amplitude pour la dérive étant donné que le centre de la coque est bombé. On observe aussi des pics de hauteur qui correspondent à des erreurs de mesures, principalement pendant des moments où l'indice qualité est faible.

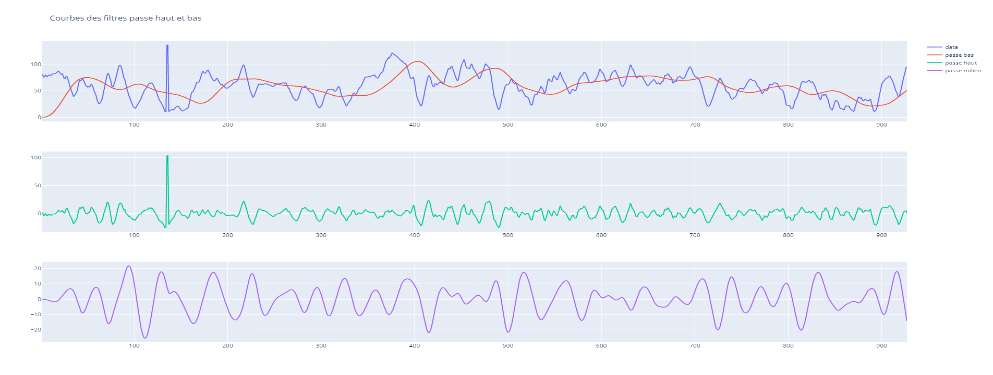
   

Sur ce tableau de droite, on retrouve les analyses de 3 zones dans différentes conditions. On peut noter que l’analyse de la mousse est la moins précise, particulièrement en zone brumeuse. L’algorithme n’est pas parfait et il existe des moments ou la mesure est incorrect, le taux d’erreur est situé entre 5% et 10%, les trois images au-dessus à droite montre les erreurs les plus communes sur la zone du safran, la non détection quand le safran visible est trop petit, l’indice qualité ne détecte pas que la zone est trop brumeuse, la mesure est trop longue car l’ombre est pris en compte.

Pour lisser la courbe et enlever le bruit causé par les clapot, les vagues et les erreurs de mesures, il est possible d'appliquer une moyenne glissante sur les données, une moyenne glissante de 50 donne de bon résultats. Mais avec cette méthode on perd l’information des vagues, une idée est d'appliquer des filtres passe haut et passe bas pour pouvoir séparer les différentes fréquences du bruit, des vagues et du bateau.

On retrouve sur les images ci-dessus la transformée de Fourier et sont spectrogramme sur des hauteur mesuré avec la maquette et un papier en forme de vagues sinusoïdale, simulant le mouvement du bateau et le mouvement des vagues. On peut séparer le spectrogramme en 3 zones, une première entre 0 et 0.5Hz qui correspond au mouvement du bateau, une deuxième entre 0.5Hz et 1Hz qui correspond aux vagues et une dernière tout ce qui est au-dessus de 1Hz qui correspond au bruit.



Les trois courbes ci- dessus montrent dans l’ordre le mouvement du bateau avec filtre passe bas (en rouge) avec en fond les données brutes, le bruit avec un filtre passe haut (en vert) et le mouvement des vagues avec un filtre passe haut puis passe bas (en violet).

# Discussion

Ce programme a permis de montrer qu’il était possible d’utiliser les caméras pour prendre des mesures. Il existe plusieurs pistes d’amélioration possible pour ce projet,

* Il pourrait être intéressant d’améliorer l’interface utilisateur pour automatiser certaines tâches comme la configuration.
* Ajouter des mesures, en particulier la surface de la coque en contact avec l’eau.
* Améliorer la précision des mesures et le temps de traitement.
* Utiliser deux caméras synchronisées pour corroborer les résultats.

Plusieurs points de vigilance sont à noter, malgré le fait que l’analyse s’adapte à la luminosité il peut arriver que dans certains cas il faut modifier les paramètres manuellement pour que la mesure soit parfaite.

Concernant l’analyse en direct le nombre d'images dans le buffer permet de savoir si le programme est assez rapide pour tenir la cadence de 25 images/s de la caméra. Sur l’appareil utilisé le programme était assez rapide pour afficher les images en direct, mais cette performance dépend beaucoup de la capacité du processeur.

Concernant l’analyse de deux caméras en simultané, il est noté à chaque fois un décalage dans le nombre d'images analysées qui varie constamment, entre 0 et 50 environ. Malgré le fait que le buffer soit vide il est donc possible que des images soient perdues entre la capture de l’image et la réception par le programme.

# Références

|  |  |
| --- | --- |
| [1] | «Python,» [En ligne]. https://www.python.org/. |
| [2] | «opencv,» [En ligne]. https://opencv.org. |
| [3] | «scipy,» [En ligne]. https://scipy.org. |
| [4] | «plotly,» [En ligne]. https://plotly.com/python. |
| [5] | «numpy,» [En ligne]. https://numpy.org. |
| [6] | «pandas,» [En ligne]. https://pandas.pydata.org. |
| [7] | «opencv tutorial contour features,» [En ligne]. https://docs.opencv.org/3.4/dd/d49/tutorial\_py\_contour\_features.html. |

# Summary

The objective of the undertaking was to assess the application of camera footage obtained from the SVR-Lazartigue, a high-speed racing sailboat, in order to gather data on the boat's flight elevation and ocean conditions. To accomplish this aim, a range of image processing methods and libraries were utilized, including OpenCV, SciPy, Plotly, NumPy, and Pandas.

The first stage of the venture was to determine the most appropriate image processing library that was applicable to the specific problem at hand. OpenCV was selected due to its open-source nature, real-time processing capabilities, and popularity within the image processing community. In addition, OpenCV has a wealth of documentation and tutorials available, making it easier to locate resources for specific tasks, such as identifying contours in images.

To measure the height of the trimaran during flight, the project employed three measurable components: the rudder, the keel, and the visible foam beneath the hull, which increases in size when the boat is in flight. The program examined only the defined regions of interest, which were selected manually by the user, to improve processing speed. The project had two separate classes of programs, depending on the area being analyzed: the "AnalyseMousse" class, which used the foam to determine height, and the "AnalyseSafran" class, which utilized the rudder to determine height.

To transform the image and analyze the areas of interest, the program executed a series of image processing methods, including a black and white filter, a Gaussian blur to reduce noise, and a binary image transformation. Additionally, the program employed thresholding algorithms to differentiate between the foam and water, and the rudder and water. Finally, the program calculated the size of the areas of interest and determined the height of the boat based on that data.

The program exhibits various depictions, including measurement drawings, pink outlines denoting detected contours, and green lines representing elevation. The outcomes manifest as height and quality index curves, which exhibit similar forms. The program experiences an error rate of approximately 5-10%, with blunders frequently arising in misty regions, diminutive objects, or long shadows. Albeit a moving average can alleviate the curves' roughness, it can potentially forfeit pertinent details. A Fourier transform with low and high pass filter can differentiate boat movements, waves, and noise.

In conclusion, the undertaking demonstrated the practicality of using camera footage to gather critical data on a high-speed racing sailboat, such as flight elevation and ocean conditions.