

École polytechnique de Louvain

RiverApp : mesure non-intrusive du débit des rivières au moyen d'une vidéo de l'écoulement, développement de l'application

Auteur: **Andréas BOMBAERT**
Promoteur: **Sandra SOARES-FRAZAO**
Lecteurs: **Lionel DRICOT, Pierre-Yves GOUSENBOURGER**
Année académique 2023–2024
Master [60] en sciences informatiques

Remerciements

Premièrement, je souhaiterais remercier Sandra Soares-Frazao et Pierre-Yves Gousenbourger pour leur supervision durant la réalisation de ce mémoire. Leurs conseils et pistes d'améliorations m'ont permis de continuer d'avancer durant l'année et de clarifier ma vision du projet. Ils m'ont aussi permis d'avoir deux perspectives différentes des objectifs à réaliser.

Deuxièmement, je voudrais remercier Lionel Dricot d'avoir accepté d'être membre de mon jury pour ce travail.

Ensuite, je tiens à remercier sincèrement Alexandre Mignot pour notre collaboration pendant l'année qui vient de s'écouler. Je lui suis reconnaissant de m'avoir éclairé sur les notions d'hydrologie dont j'avais besoin pour la bonne compréhension de l'application.

Je voudrais également remercier mes parents pour leur soutien indéfectible durant la réalisation de ce travail, et plus récemment pour la relecture assidue de celui-ci.

Merci à Louis pour le squelette de la mise en page de ce document, me faisant gagner un temps précieux.

Gil et Manu pour le prêt de leur drone pour la prise de vidéos au dessus de la rivière.

Pierre et Benoît pour le prêt de leurs voitures respectives pour se déplacer aux sites de prises de mesure.

Finalement, je voudrais remercier les membres du projet Rivialis pour leur pédagogie lors des visites de sites, ainsi que pour les images de la prise de mesure du Petit Bocq.

Résumé

Dans bien des domaines, la connaissance du débit d'un cours d'eau est indispensable. Pensons entre autres à la navigation, la conception d'ouvrages hydrauliques ainsi que la surveillance des cours d'eau.

Dans le contexte des événements climatiques, (e.g. les inondations de 2021), la surveillance accrue des cours d'eau est essentielle pour la prévention de catastrophes, ainsi que la gestion des dégâts liés à celles-ci. Cette surveillance est basée sur plusieurs critères dont la connaissance du débit.

Il existe deux types de méthodes pour calculer celui-ci : les méthodes intrusives et non-intrusives. L'utilisation des méthodes intrusives n'étant pas possible durant les périodes de crue ou d'étiage, il est important de développer des techniques non-intrusives efficaces.

Ce mémoire rentre dans la continuité des mémoires réalisés depuis quelques années sur l'étude des analyses PIV (Particle Image Velocimetry) et LSPIV (Large Scale Particle Image Velocimetry), et se concentre plus précisément sur le développement d'une application de calcul de débit.

Là où les mémoires réalisés ces dernières années sur le calcul de débit par imagerie se concentraient surtout sur l'analyse des technologies utilisées, le présent travail est principalement axé sur le développement de la nouvelle interface et des outils qui l'entourent. Ces outils consistent à améliorer la qualité de développement et d'utilisation de l'application, cela passe par un déploiement de la charge des calculs lourds sur serveur, une stratégie de testing et une documentation complète et fiable.

Avertissements

L'intelligence artificielle a été utilisée à quelques reprises dans ce mémoire. L'utilisation principale de celle-ci s'est faite via GitHub Copilot, qui est une intelligence artificielle d'autocomplétion du code.

La deuxième utilisation de l'IA a été celle de Google Gemini pour la rédaction de tableaux LaTeX et de documentation python (l'utilisation pour la documentation sera rappelée plus tard dans ce document).

La dernière utilisation de logiciel externe est l'utilisation du site scribens pour la correction de certains paragraphes de ce travail.

Table des matières

Acknowledgements	1
Résumé	1
1 Introduction	5
2 Etat de l'art	7
2.1 Calculs de débit	7
2.1.1 Mesures intrusives	7
2.1.2 Limites des mesures intrusives	9
2.1.3 Mesures non-intrusives	10
2.1.4 Préprocessing et données nécessaires	11
2.2 Solutions existantes	11
2.3 RiverApp v1 & PyORC (PyOpenRiverCam)	12
2.4 Nouvelle interface	13
3 Etat de l'application	15
3.1 Etat de l'application reçue en septembre	15
3.2 Fonctionnalités manquantes pour terminer le mémoire précédent . .	16
3.3 Analyse critique et erreurs dans l'implémentation initiale	16
3.3.1 La PIV ne s'effectue pas	17
3.3.2 Mauvaise conversion de la position des balises	17
3.3.3 Paramètres hardcodés	18
3.3.4 Association incorrecte des distances entre les balises	19
3.4 Qualité du code	20
4 Développement du MVP	22
4.1 Définition du MVP	22
4.2 Correction des erreurs bloquantes	23
4.3 Développement des fonctionnalités nécessaires	23
4.4 Testing	25

4.5	Constitution d'un dataset (prise de mesures)	26
4.5.1	Petit Bocq	27
4.5.2	Limelette	27
4.5.3	Qualité du code	27
5	Déplacement de l'effort de calcul sur serveur distant	28
5.1	API	28
5.1.1	Nécessité de l'API pour utiliser l'application sur téléphone	28
5.1.2	Quelles fonctionnalités migrer vers l'API?	28
5.1.3	Testing	29
5.2	Solution de déploiement	29
5.3	Git flow et automatisation	30
5.3.1	Git flow	30
5.3.2	Automatisation	30
6	Résultats	32
6.1	Résultats des anciennes versions	32
6.2	Résultats de la nouvelle application	33
6.2.1	Débits obtenus et observations	33
6.2.2	Paramètres d'influence	38
6.2.3	Temps d'exécution	40
6.2.4	Détection des balises	41
6.3	Comparaison & Conclusion	42
7	Documentation	44
7.1	Guide développeur	44
7.2	Guide utilisateur	45
8	Pistes d'amélioration	47
8.1	Développement de l'application	47
8.2	API	50
8.3	Prises de mesures	51
9	Conclusion	53
	Appendices	55
A	Prise de débit au courantomètre	56
B	Tableaux de résultats de la nouvelle interface	58
B.1	Débits limelette - vidéo du milieu du pont	58
B.2	Débits limelette - vidéo de la gauche du pont	59

B.3	Débits limelette - vidéo de la droite du pont	59
B.4	Débit Petit Bocq - vidéo à 90°	59
B.5	Débit Petit Bocq - vidéo à 45°	60
B.6	Débit VGC	60
C	Diagramme du fonctionnement de l'application	61
D	Protocole de prise de mesures	63

Chapitre 1

Introduction

Dans bien des domaines, la connaissance du débit d'un cours d'eau est indispensable. Pensons entre autres à la navigation, la conception d'ouvrages hydrauliques ainsi que la surveillance des cours d'eau.

Dans le contexte des événements climatiques, (e.g. les inondations de 2021), la surveillance accrue des cours d'eau est essentielle pour la prévention de catastrophes, ainsi que la gestion des dégâts liés à celles-ci. Cette surveillance est basée sur plusieurs critères dont la connaissance du débit.

Il existe deux types de méthodes pour calculer celui-ci : les méthodes intrusives et non-intrusives. L'utilisation des méthodes intrusives n'étant pas possible durant les périodes de crue ou d'étiage (pour des raisons expliquées au chapitre 2), il est important de développer des techniques non-intrusives efficaces.

Ce mémoire rentre dans la continuité des mémoires réalisés depuis quelques années sur l'étude des analyses PIV (Particle Image Velocimetry) et LSPIV (Large Scale Particle Image Velocimetry), et se concentre plus précisément sur le développement d'une application de calcul de débit. La première itération de ce développement date de 2020 dans le mémoire de S. Renard[1]. Pour des raisons explicitées par la suite dans ce travail, ainsi que dans les mémoires de G. Debray et D. Hay [2], une nouvelle interface a vu le jour plus récemment, en 2023, dans le travail de fin d'études de A. Meunier[3]. Là où les mémoires réalisés ces dernières années sur le calcul de débit par imagerie se concentraient surtout sur l'analyse des technologies utilisées, le présent travail est principalement axé sur le développement de la nouvelle interface et des outils qui l'entourent.

L’objectif global de ce travail est d’améliorer la qualité actuelle de l’application, ainsi que des outils informatiques qui gravitent autour.

La priorité est d’achever le développement du MVP (Minimum Viable Product), afin de permettre à un utilisateur lambda d’effectuer le parcours complet de l’application et d’obtenir un débit théorique. Celui-ci est défini plus en profondeur dans la sous-section 4.1. Le projet actuel est uniquement constitué du code de l’application, et, dans l’objectif de continuer le développement (et d’ouvrir ce dernier à des contributeurs externes), il est nécessaire de munir le projet d’une stratégie de testing, ainsi que d’un contrôle continu de la qualité du code.

Pour améliorer ce processus de vérification de la qualité du projet, il est souhaitable d’automatiser ces étapes via des outils d’intégration continue¹. Premièrement pour faciliter le développement du projet, ensuite pour garantir que cette étape est bien effectuée.

La nouvelle interface vise à être plus intuitive pour l’utilisateur. Il est néanmoins obligatoire de fournir une documentation complète à ce dernier afin de permettre une utilisation optimale de l’application. Le but ici est donc de rédiger un guide à la fois pour l’utilisateur, et pour le développeur, de manière à ce que tout type d’utilisateur se retrouve dans le projet. Toujours dans le but de simplifier l’utilisation de l’application par un utilisateur lambda, un transfert de la charge de calcul sur serveur doit être effectué avant de déployer l’application sur téléphone.

La première partie de ce travail développe le calcul de débit de rivière, ainsi que la situation actuelle en termes de calcul via méthodes non-intrusives. Un chapitre est dédié à l’état de l’application telle que développée dans le travail de fin d’études de A. Meunier[3]. Les autres parties sont consacrées au développement des différents objectifs cités précédemment, et seront conclues par les pistes d’améliorations identifiées durant la réalisation de ce mémoire.

1. L’intégration continue est une méthode de développement de logiciel DevOps par laquelle les développeurs intègrent régulièrement leurs modifications de code à un référentiel centralisé, suite à quoi des opérations de création et de test sont automatiquement exécutées.[4]

Chapitre 2

Etat de l'art

Ce chapitre est une introduction aux calculs de débits de rivières et au passé du projet RiverApp. Il contient un résumé des différentes méthodes classiques de calcul, des méthodes informatisées développées à cet effet, ainsi que l'évolution de RiverApp ces dernières années. La majorité des informations concernant le calcul de débit sont tirées des analyses de E. de Grand Ry et C. Everaert, ainsi que G. Debray et D. Hay dans leurs travaux de fin d'études de 2021[5] et 2023[2].

2.1 Calculs de débit

2.1.1 Mesures intrusives

Par soucis de place et du fait que ceux-ci ne sont pas l'objet de ce mémoire, ils ne seront que brièvement développés dans cette section, de plus amples informations sont disponibles dans les mémoires cités ci-dessus.

Capteur à moulinet Celui-ci porte également le nom de courantomètre mécanique. A l'intérieur de l'appareil, un signal est généré et produit la révolution d'une hélice. Le nombre de tours de cette hélice (déterminé par un compteur) sur une certaine période de temps permet de calculer la vitesse angulaire due à la vitesse de l'eau. Voir fig. 2.1a pour une illustration..

Courantomètre électromagnétique Le courantomètre électromagnétique utilise une sonde qui génère un champ magnétique. D'après la loi de Faraday, un conducteur en mouvement dans un champ magnétique génère une différence de tension qui varie linéairement avec la vitesse du conducteur. Les conducteurs sont ici les particules d'eau chargées qui bougent selon le courant. Celles-ci vont donc

générer une force électromagnétique qui pourra être mesurée par des électrodes présents dans la sonde. Voir fig. 2.1b pour une illustration..

Vélocimètre acoustique Les appareils acoustiques utilisés pour la mesure de vitesse sont aussi appelés Acoustic Doppler Velocimeter (ADV) ou Acoustic Doppler Current Profiler (ADCP). Ces appareils émettent une onde acoustique dans l'eau. Celle-ci est réfléchiée par les particules du cours d'eau et analysée selon le principe de Doppler.

Cependant, les vitesses ne peuvent être mesurées que sur une partie de la profondeur d'eau. En effet, à cause de la réflexion des ondes sur le fond du cours d'eau, les vitesses de la partie inférieure de la profondeur (environ six pour cents de la hauteur) ne sont pas mesurables [6]. De plus, comme l'appareil est légèrement enfoncé dans l'eau, les vitesses de la zone sous la surface d'eau ne sont pas non plus mesurables. Voir fig. 2.1d pour une illustration..

Méthode des flotteurs Dans cette méthode, des traceurs sont dispersés sur la surface du cours d'eau. Le temps mis par les particules pour parcourir une distance déterminée est mesuré. A partir de ces données, les vitesses de surface sont obtenues et ensuite transformées en vitesses moyennes sur la profondeur grâce à un certain coefficient. Pour obtenir des résultats corrects, il faut que les particules soient bien réparties sur toute la largeur du cours d'eau [2]. Cette méthode est généralement utilisée si aucune des méthodes citées précédemment n'est utilisable.

Jaugeage par perche transparente Le jaugeage par perche transparente est une méthode peu coûteuse et efficace dont le principe de mesure est de mesurer la différence de niveau d'eau entre l'amont et l'aval d'une planche placée perpendiculairement face à l'écoulement du cours d'eau. En plaçant la perche de cette manière face au flux d'eau, on peut déterminer la force dynamique liée à la vitesse moyenne de l'écoulement[7]. Grâce à une feuille de calcul prédéfinie on peut obtenir un débit théorique. Voir fig. 2.1c pour une illustration.

Jaugeage par dilution de sel On injecte une masse M de chlorure de sodium dans un cours d'eau, puis on observe en aval le passage du nuage salin (figure 2). La méthode de jaugeage consiste à mesurer l'évolution de la conductivité électrique de l'eau à l'aide d'un conductimètre[8].

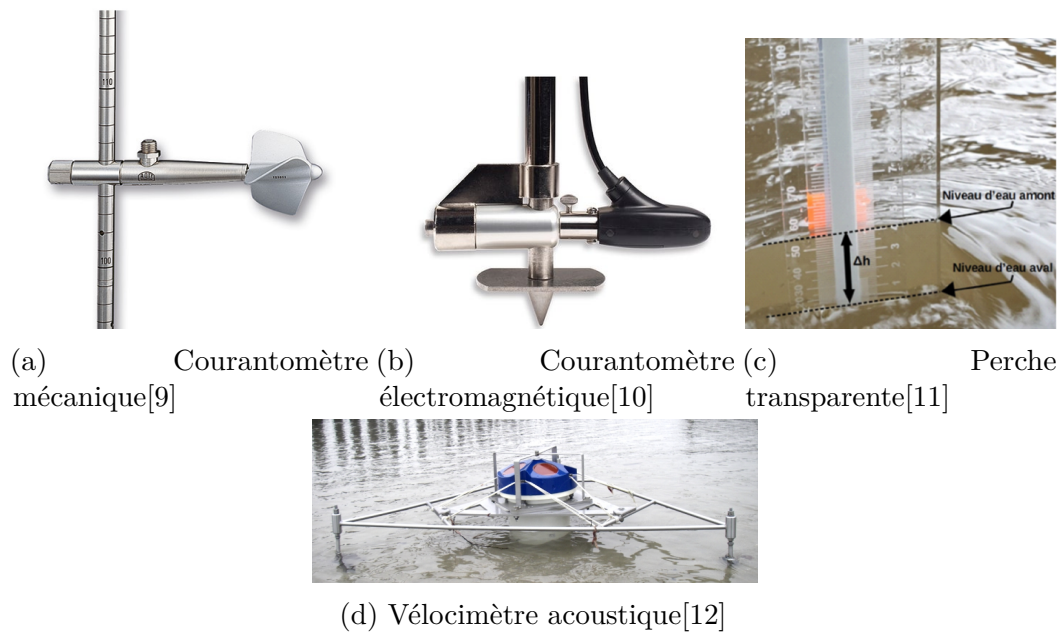


FIGURE 2.1 – Illustrations de quelques solutions présentées ci-dessus

Illustrations des solutions présentées

2.1.2 Limites des mesures intrusives

Les mesures intrusives rencontrent des obstacles lors des périodes de crue ou d'étiage. Or, les mesures de débit sont indispensables, que ce soit pour la prédiction de crue ou encore pour le design d'ouvrages hydrauliques.

Le premier problème est que les instruments de mesure intrusifs ne sont pas prévus pour être utilisés en période de crue. En effet, ils pourraient être abîmés par le transport de sédiment, ou tout simplement par la force du débit.

De plus, ces instruments ne sont pas adaptés pour les mesures de faibles débits. Les mesures obtenues lors des périodes d'étiage ne sont pas toujours fiables.

Enfin, les mesures intrusives nécessitent la majorité du temps l'intervention humaine. En cas de crue, ou de rivière inaccessible, cette intervention est alors dangereuse, voir impossible. Les possibilités de mesures sont donc limitées avec l'utilisation de ces techniques, et ne conviennent pas à toutes les situations.

2.1.3 Mesures non-intrusives

PIV L'analyse PIV (Particle Image Velocimetry) est une méthode de mesures optique qui permet de calculer le champ de vitesses instantanées d'une surface à partir d'images consécutives d'un cours d'eau. Le cours d'eau estensemencé¹ de particules traceuses qui sont supposées suivre l'écoulement naturel de celui-ci.

La PIV analyse les images par paire afin de déterminer le suivi des particules d'image en image, ce qui permet de déterminer un vecteur par pattern, et ensuite le champ de vecteurs sur la région analysée. Le suivi des particules est analysé majoritairement via **autocorrélation**. Le principe de l'autocorrélation est de comparer un groupe des particules dans deux images consécutives, séparées par un temps connu Δt . Cette comparaison s'effectue en évaluant la fonction d'autocorrélation et permet de déterminer le déplacement entre deux images Δx . De cette façon, la vitesse peut être déduite.

A partir de ce champ de vecteurs de vitesse et en connaissant la bathymétrie, ainsi que le coefficient de surface², le débit du cours d'eau peut être obtenu.

LSPIV Alors que la PIV à été développée pour être utilisée en laboratoire, la méthode LSPIV (Large Scale Particle Image Velocimetry) correspond à l'utilisation de la PIV à grande échelle, dans des conditions moins contrôlées et sur des objets plus grands. Les particules d'ensemencement doivent être plus grandes pour pouvoir être détectées sur un champ de vision agrandi. Cependant, l'ensemencement n'est plus nécessaire sur un cours d'eau à grand débit où les remous peuvent être suivis. La méthode suit les quatre étapes suivantes :

1. **Acquisition** des images dont la différence de temps Δt est connue entre chacune d'elle.
2. **Orthorectification** des images, c'est la rectification géométrique qui permet de corriger les distorsions dues à la perspective.
3. **Calcul** des déplacements Δx entre chaque image, basés sur les corrélations entre celles-ci.
4. **Post-processing** des données de vitesses obtenues. Il s'agit des moyennes de vitesses et de l'estimation du débit.

1. L'ensemencement consiste à disposer des copeaux de bois (ou autres particules flottantes) pour simuler du remous sur la vidéo du cours d'eau

2. Le profil de vitesse n'est pas constant sur toute la profondeur du cours d'eau. Or le débit est calculé à partir des vitesses en surface, il faut donc corriger la surestimation de la vitesse en déterminant un coefficient de correction de la vitesse. Ce facteur dépend du profil de la rivière étudiée.

2.1.4 Préprocessing et données nécessaires

Préprocessing L'utilisation de ces méthodes non-intrusives nécessite en général plusieurs étapes de pré-processing servant à préparer la vidéo pour l'analyse :

- **Stabilisation** Compensation des mouvements de la caméra sur les images d'un écoulement.
- **Calibration** Compensation des effets de distorsion des images dus au type de lentille de caméra utilisée.
- **Orthorectification** : Compensation des effets de distorsion des images dus à la perspective.
- **Application de filtres** : Pendant l'analyse PIV, la valeur de chaque pixel va influencer le résultat du calcul de la cross-corrélation. Il est parfois préférable d'appliquer un filtre aux images de la vidéo de manière à amplifier les contrastes et rendre plus précis la détection des patterns d'une image à l'autre.

Données nécessaires Pour fonctionner correctement, et cela sera rappelé fréquemment dans les prochains chapitres, la PIV a besoin des paramètres suivants :

- Vidéo de l'écoulement pour permettre le calcul du champ de vitesse en surface
- Bathymétrie³
- Hauteur d'eau sur la section analysée
- Coefficient de surface
- Résolution de la vidéo, afin de pouvoir effectuer une conversion pixel-mètre

2.2 Solutions existantes

Le calcul du débit d'un cours d'eau via des techniques d'imagerie existant depuis déjà quelques temps, il existe quelques solutions prévues à cet effet. Cependant, la plupart d'entre elles sont payantes, et les autres ne sont pas open-source, les rendant inutilisables dans le cadre du projet RiverApp.

Fudaa-LSPIV Fudaa[13] est un logiciel libre co-développé et diffusé par EDF et Irstea avec la société DeltaCAD. Il utilise la LSPIV (Large Scale PIV) pour calculer le débit. Bien qu'il offre des fonctionnalités intéressantes et qu'il corresponde aux besoins de RiverApp, il n'est pas intégrable dans une autre application. De plus,

3. Mesures du relief d'un cours d'eau, dans notre cas sur la section de la rivière que l'on souhaite analyser

son utilisation n'est pas aisée pour un utilisateur lambda, plus particulièrement dans le choix des paramètres qui peuvent influencer considérablement le résultat.

PIVLab PIVLab[14] est une routine d'analyse PIV dans MatLab qui permet d'appliquer la technique PIV pour obtenir les vitesses moyennes de surface en analysant les images par paires. Cette routine possède une interface graphique qui rend son utilisation plus visuelle pour l'utilisateur.

Cependant, étant développée sur MatLab (logiciel dont la licence est payante), elle rend son utilisation impossible dans le cadre du projet RiverApp.

RIVeR RIVeR[15] est une application gratuite permettant de calculer le débit d'une rivière à partir d'images de celle-ci. Elle utilise deux techniques non-intrusives pour analyser les images : Particle Image Velocimetry (PIV) et Particle Tracking Velocimetry (PTV). Aux premier abord, l'application a tout ce qu'il faudrait pour être utilisée dans le projet RiverApp. Cependant, encore une fois, elle utilise des routines MatLab (PIVLab & PTVLab) pour performer ces analyses, ce qui l'élimine de la liste des candidats potentiels pour RiverApp.

Discharge Discharge[16] est une application Android permettant de déterminer rapidement la vitesse et le débit d'une rivière à partir d'une vidéo. L'avantage de l'application est la vitesse de celle-ci, en revanche, elle ne permet pas de réaliser l'analyse sur une vidéo prise au préalable. Elle n'est pas non plus open-source, la modification de celle-ci aux fins de RiverApp n'est pas envisageable. De plus, son utilisation nécessite des conditions particulières (e.g. les particules présentes naturellement doivent être suffisamment visibles pour être tracées).

Tenevia Camflow Tenevia CamFlow[17] est une application propriétaire destinée à la surveillance et à la prévision d'informations environnementales. Elle permet de surveiller le débit d'un cours d'eau en direct. Cependant, elle nécessite que la mesure se fasse sur la même configuration du cours d'eau en permanence, et est payante et donc non modifiable aux fins de RiverApp. Ce qui la raye également des candidats potentiels.

2.3 RiverApp v1 & PyORC (PyOpenRiverCam)

Une première version de RiverApp a vu le jour en 2019-2020[1], et a été améliorée en 2021 par E. de Grand Ry et C. Everaert[5]. Les résultats de cette version seront apportés par P. Laurent et C. Dufromont en 2022[18]. Ces résultats se rapprochent du débit réel calculé par Aqualim, toujours en gardant une marge d'erreur non

négligeable. Le moteur de calcul de cette application a été codé manuellement et ne fait pas appel à une librairie spécialisée dans les analyses PIV ou autres analyses de débit.

PyORC quant à elle, est une librairie python permettant de réaliser une analyse LSPIV sur une vidéo donnée. Elle est avantageuse dans le sens où elle permet d'effectuer à la fois cette analyse, mais également les étapes de pré-processing citées auparavant. Elle facilite aussi l'affichage des résultats de l'analyse. PyORC fournit également une documentation complète, ainsi que des notebooks pour pouvoir tester le code localement.

Une comparaison entre RiverApp v1 et PyORC a été réalisée (encore une fois, dans le mémoire de 2023[2]), et PyORC offre de meilleurs résultats, autant dans la précision de ceux-ci, que dans leur constance. Selon les analyses réalisées en 2023, PyORC gère mieux les variations d'images, ainsi que les jeux de données moins précis, ce qui en fait le candidat idéal pour la suite du projet RiverApp.

2.4 Nouvelle interface

L'interface de RiverApp v1 n'étant pas intuitive et les résultats de cette application se montrant moins satisfaisants que la librairie PyORC, une nouvelle interface autour de PyORC a été développée durant l'année académique 2022-2023 par Arnaud Meunier dans son travail de fin d'études[3]. Cette interface doit faire le lien entre un utilisateur lambda et l'utilisation de la librairie PyORC afin de rendre le calcul du débit le plus accessible (et open-source) possible. L'interface est sensée combler les besoins fonctionnels tels que décrits dans ce travail :

- L'ergonomie du projet en modifiant la structure de celui-ci ainsi qu'en retravaillant son interface
- Les bonnes pratiques de programmation en revoyant les concepts et le code de l'application pour les rendre conformes aux pratiques actuelles
- Le testing et la gestion d'erreurs qui sont totalement absents dans le projet
- L'automatisation des processus nécessaires à l'intégration continue ;
- La collaboration entre développeurs à l'aide d'outils GitLab et d'une documentation complète ;
- L'amélioration technique du projet avec des corrections et des ajouts de nouvelles fonctionnalités comme :
 - un module d'adaptation de la résolution des flux vidéo afin d'en avoir une idéale
 - une correction et une amélioration pertinentes des filtres de l'application
 - une optimisation des phases intensives de calculs.

Seul un faible pourcentage du cahier des charges a réellement été honoré. Cela sera développé dans le chapitre 3.

Chapitre 3

Etat de l'application

3.1 Etat de l'application reçue en septembre

L'application RiverApp est le produit d'une succession de travaux de fin d'études réalisés sous la supervision de la Prof. Sandra Soares-Frazao. Après plusieurs itérations sur une première version de test (incluant un moteur de calcul artisanal et une interface graphique minimaliste), il a été décidé, en 2022-2023, de refondre entièrement l'interface graphique, mais aussi de remplacer le moteur de calcul par le module PyORC (Py Open River Cam). L'application telle que reçue contient les écrans suivants :

- **Configuration de la vidéo** : L'enregistrement de la vidéo de l'écoulement ainsi que les différents paramètres de la vidéo (temps en secondes de début et de fin de l'analyse PIV, nombre d'images par seconde que l'on souhaite analyser).
- **Bathymétrie** : L'enregistrement du fichier de la bathymétrie, ainsi qu'un graphique représentant cette dernière pour que l'utilisateur puisse vérifier qu'elle correspond aux mesures observées. L'utilisateur doit également rentrer la hauteur de l'eau dans le champ prévu à cet effet.
- **Zone de recherche** : L'écran contient le placement des balises servant à définir la zone de recherche de l'analyse PIV ainsi que l'encodage des distances entre chaque balise. Ces distances sont utilisées pour l'orthorectification de l'image lors de l'analyse et de l'affichage des vecteurs de vélocimétrie.
- **Filtres** : Un écran servant à appliquer des filtres à l'image avant d'analyser ladite vidéo.
- **PIV** : Le lancement de l'analyse PIV utilisant la configuration du projet préparée à travers les étapes précédentes.

Le projet reçu contient également un début de documentation pour ce qui est de la procédure d'installation, ainsi qu'une dizaine de tests unitaires offrant une

couverture largement insuffisante pour assurer le maintien efficace du projet.

L'application actuelle offre un début de processus fonctionnel. Cependant, il manque les parties les plus importantes et la documentation de l'application. Nous verrons dans le chapitre 7 que le code n'est pas non plus documenté correctement et qu'il faut y remédier.

3.2 Fonctionnalités manquantes pour terminer le mémoire précédent

La principale fonctionnalité manquante concerne la finalisation de l'analyse PIV et l'affichage du résultat obtenu.

Une fois la PIV terminée, on doit utiliser le résultat obtenu par celle-ci pour afficher les transects sur la vidéo afin de permettre à l'utilisateur de visualiser le courant en surface. On utilise alors le résultat de l'analyse PIV pour donner à l'utilisateur le débit théorique calculé, qu'il reste encore à implémenter.

Une autre fonctionnalité ayant son importance est l'application de filtres sur l'image, ce qui pourrait amener (ou non) à des résultats plus proches de la réalité. L'utilisateur pourrait donc vouloir les utiliser dans son analyse. Cela dit, cette fonctionnalité est secondaire et n'est pas obligatoire pour obtenir un premier débit.

Comme mentionné précédemment, la couverture de tests unitaires est très faible et ne convient pas pour tester efficacement l'application actuelle. Nous verrons dans la sous-section suivante que ce ne sont pas les seuls obstacles rencontrés avec le projet reçu.

Pour permettre l'utilisation et la transmission de savoir quant à l'application, il faut documenter toutes les étapes citées auparavant.

3.3 Analyse critique et erreurs dans l'implémentation initiale

La plupart des fonctionnalités développées fournissent déjà un état satisfaisant à l'application. Cependant, elle donne une fausse impression de fonctionnement global. On remarque rapidement plusieurs points critiques nécessitant un remaniement.

Notons que ces points auraient pu être détectés plus efficacement si la couverture de tests avait été étendue.

Après analyse du code, il s'avère que l'auteur n'a pas testé le fonctionnement de ces fonctionnalités, ou du moins n'en a pas compris l'essence. Il est alors important de préciser que lors de la transmission du projet, il a été expliqué à l'oral que toutes les fonctionnalités présentes étaient bien fonctionnelles. Les sous-sections suivantes en démontrent le contraire.

Les erreurs d'implémentations suivantes ont été détectées, rendant l'analyse PIV incorrecte voire impossible.

3.3.1 La PIV ne s'effectue pas

Fonctionnement attendu Pour effectuer une PIV avec PyORC, on donne la configuration de la caméra telle qu'utilisée pour les mesures sur lesquelles on effectue l'analyse (configuration contenant la position des balises et la distance entre elles, la position de la caméra, la résolution de celle-ci). On donne également les secondes auxquelles on veut démarrer et finir la PIV, ainsi que le nombre d'images que l'on souhaite analyser par seconde.

Problème soulevé Ici, la configuration de la caméra n'est jamais assignée aux données du projet en cours, la PIV se contente de planter instantanément sans prévenir l'utilisateur qui attend indéfiniment.

Etat du problème Le problème est critique car la suite du projet dépend uniquement de cette PIV. Trouver ce bug a été relativement aisé grâce aux retours d'erreur de la console python.

3.3.2 Mauvaise conversion de la position des balises

Fonctionnement attendu Comme expliqué dans la sous-section précédente, la PIV a besoin de la position des balises pour s'effectuer correctement. Cependant, dans l'application, les positions sont calculées par rapport à la hauteur en pixel de la fenêtre de l'application. Il y a lieu de les convertir en fonction de la taille en pixels de l'image originale.

Problème soulevé La conversion des pixels pour l'axe vertical donne un y inversé comme le montre la figure 3.1.

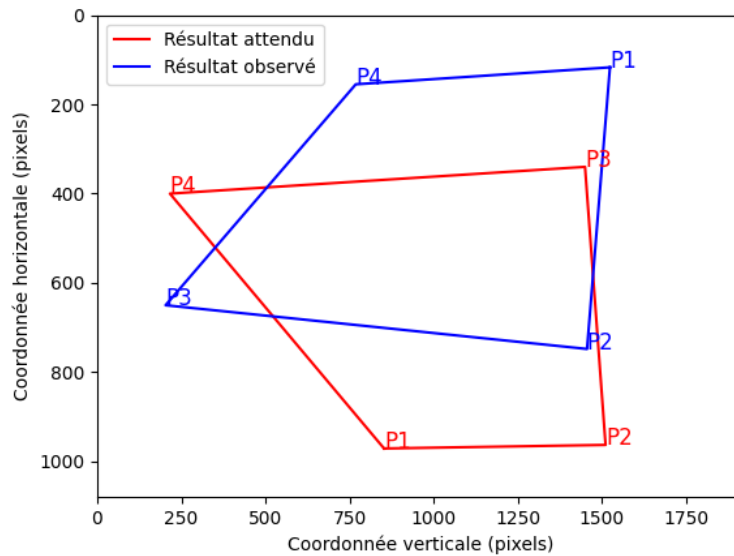


FIGURE 3.1 – Comparatif des deux dernières versions

Etat du problème Ce problème se révèle également critique car il empêche la PIV de fournir les résultats attendus. En revanche, trouver ce bug fut moins aisé que le précédent étant donné qu'il a fallu retracer tout le chemin depuis l'affichage des transects pour comprendre d'où venait l'erreur et les résultats faussés.

3.3.3 Paramètres hardcodés

Fonctionnement attendu Pour utiliser la librairie PyORC correctement, il existe plusieurs paramètres qui varient pour chaque configuration de prises de mesures. Si ceux-ci ne sont pas changés pour chaque vidéo différente, les résultats de la PIV et de l'affichage des transects seront mal calculés.

Problème soulevé Certains de ces paramètres sont hardcodés dans l'application. Ces paramètres sont valables uniquement pour les mesures d'une des vidéos du Video Global Challenge. Le problème apparaît dès qu'on essaie d'utiliser une autre configuration (PIV qui ne détecte aucune vélocimétrie en surface, Affichage des transects faussé).

Il est nécessaire que les paramètres suivants soient passés via l'interface par l'utilisateur de l'application.

- La position de la caméra, sous la forme d'une liste de coordonnées $[x, y, z]$. Cette position est utilisée pour la définition de la zone de recherche de la PIV.

Si ce paramètre n'est pas configuré correctement, la PIV ne fonctionnera pas et il ne sera pas possible d'afficher les transects.

- La valeur du coefficient de surface
- La position du vecteur d'affichage des transects (voir fig. 3.2), ces deux coordonnées $[x, y]$ définissent la ligne sur laquelle on va calculer le débit et afficher les vecteurs de débit après l'analyse PIV. Si ce vecteur n'est pas défini spécifiquement pour chaque configuration, l'affichage et le calcul du débit sont également faux.

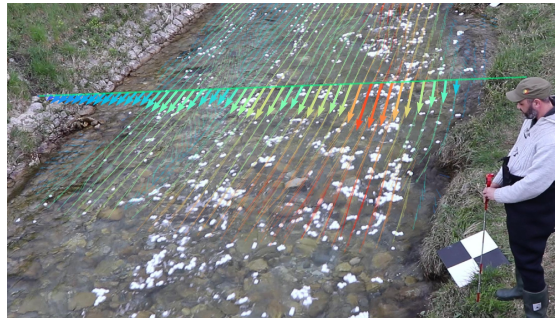


FIGURE 3.2 – Vecteur d'affichage des transects (en vert)

Etat du problème Le bug est également critique car il empêche aussi l'utilisateur d'avoir des résultats sensés.

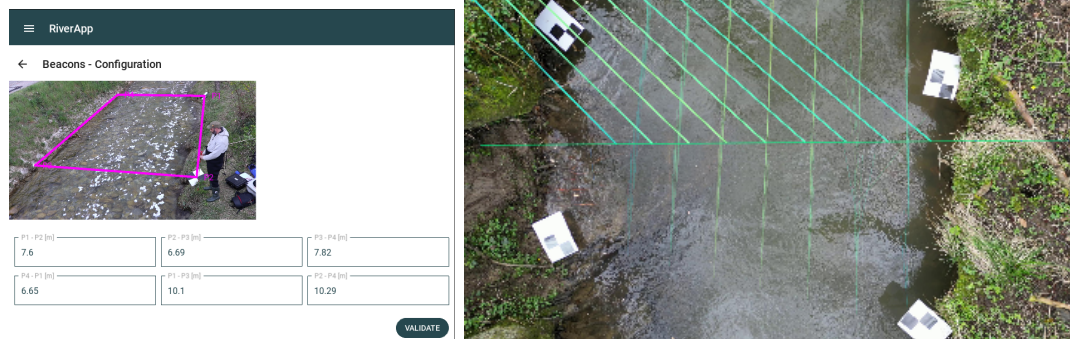
3.3.4 Association incorrecte des distances entre les balises

Fonctionnement attendu Sur l'écran de configuration des balises, on doit rentrer les distances entre chaque balise pour que PyORC puisse rectifier l'image et l'analyser proprement. Chaque entrée de distance doit donc associer la valeur au bon lien de balises pour assurer un fonctionnement correct du projet.

Problème soulevé Les entrées de l'écran de configuration n'associent pas les bonnes valeurs aux bonnes balises, la PIV reçoit alors une zone de recherche inversée. Sur la fig. 3.3a on peut voir les titres de chaque entrée "P1 - P2 [m]", ceux-ci n'associaient pas les valeurs aux bonnes distances, ici "P1 - P2 [m]" associait la distance entrée au vecteur P4 - P1, et aucune case n'était bien paramétrée.

Etat du problème Le problème ici est que l'étudiant en charge de l'application entrait les valeurs de la configuration du Video Global Challenge sans savoir quelle valeur était réellement associée à quelle distance. Les résultats de la PIV sont corrects avec le Video Global Challenge mais dès qu'on change de configuration, la

PIV reçoit des distances incorrectes et produit un résultat faussé comme le montre l'exemple de la fig. 3.3b.



(a) Illustration de la définition des valeurs des distances (b) Exemple d'un résultat faussé par une mauvaise association des distances

FIGURE 3.3 – Illustrations du problème soulevé

3.4 Qualité du code

Les prochains points n'impactent pas le fonctionnement du code en lui-même, mais ils offrent une lecture et une compréhension plus difficile du code. La dernière version de pylint¹ évalue l'état de l'application reçue à 1.34/10 avec environ 250 avertissements quant à la propreté du code.

Respect des conventions On attend d'un projet python qu'il respecte certaines conventions de base [19], voici quelques unes des plus importantes pour avoir un code clair :

- Lignes de longueur maximale de 79 caractères
- Gestion des espaces, les opérateurs doivent être entourés d'espaces et ils doivent également suivre les virgules
- Nommage des variables et des fonctions, utiliser du snake_case²
- Utiliser des noms de variables et fonctions explicites

Dans le code, on remarque que ces bonnes pratiques n'ont pas été respectées, on y trouve des dizaines de lignes dépassant 120 caractères, des noms de variables

1. Librairie python permettant l'évaluation de la qualité du code python en se basant sur les bonnes pratiques de programmation et conventions pythons

2. Convention de nommage des variables et fonctions consistant à mettre un underscore entre chaque mot du nom, exemple : position_camera

contenant des typos, du variable shadowing³.

Organisation Concernant l'organisation du code, on attend intuitivement de celle-ci qu'elle soit structurée par écran/fonctionnalités. Actuellement, le front-end est organisé par écran contenant chacun certaines fonctionnalités. En revanche, l'organisation du back-end est structurée par fichiers regroupant des fonctionnalités de plusieurs écrans. Ce choix donne lieu à un fonctionnement contre-intuitif de l'application, par exemple l'analyse PIV est lancée dans le fichier qui gère la sauvegarde des données de chaque projet, ou encore la conversion des pixels des balises qui se fait dans ce même fichier. Or, ces opérations devraient être réalisées dans le même fichier que leurs écrans respectifs.

Conclusion Un refactoring est nécessaire sur l'organisation des fichiers et fonctions du back-end. Une partie des fonctions doit retourner dans le front-end et d'autres fonctions devraient avoir des fichiers propres à leur portée. Concernant la propreté du code on peut aisément refactorer les fichiers concernés grâce au rapport fourni par pylint (ou autre utilitaire), qui offre un détail ligne par ligne de chaque avertissement.

3. Dans ce cas-ci, lorsqu'une variable contient le même nom qu'une fonction innée de python (ex : max, sum, ...) [20]

Chapitre 4

Développement du MVP

4.1 Définition du MVP

Le MVP, Minimum Viable Product doit permettre à l'utilisateur d'effectuer le parcours complet d'un utilisateur qui souhaite obtenir le débit d'une section d'une rivière, dont il connaît les informations nécessaires à l'analyse PIV. Cette version de l'application doit contenir les fonctionnalités utilisateurs suivantes :

- **Gestion des projets** : l'utilisateur doit pouvoir créer, modifier et supprimer des projets. L'application doit gérer la sauvegarde des informations de chaque projet pour que l'utilisateur puisse y revenir après un redémarrage.
- **Configuration de la vidéo** : l'utilisateur doit pouvoir entrer une vidéo en choisissant parmi les fichiers de son appareil : les secondes de démarrage et de fin de l'analyse, la fréquence d'échantillonnage ainsi que la position de la caméra à travers des champs de saisie.
- **Configuration de la bathymétrie** : l'utilisateur doit pouvoir entrer la bathymétrie de la section qu'il veut analyser, ainsi que la hauteur de l'eau au moment de la mesure. L'utilisateur doit aussi pouvoir choisir le coefficient de surface qu'il souhaite appliquer, le tout à travers des champs de saisie.
- **Configuration des balises et distances** : l'utilisateur doit pouvoir sélectionner l'emplacement des balises sur l'image pour définir la zone de recherche. Pour ce faire, l'utilisateur doit pouvoir déplacer des points sur l'image pour définir ces emplacements. Il doit aussi pouvoir définir les distances entre chaque balises à travers des champs de saisie.
- **Affichage des transects et du débit** : L'utilisateur doit pouvoir sélectionner le vecteur d'affichage des transects sur l'image, et recevoir l'image traitée avec le vecteur à l'endroit sélectionné, ainsi que le débit théorique obtenu par l'application.

Du côté du développement, l'application doit contenir une procédure d'installation

pour que la prise en main du projet soit aisée pour quiconque souhaiterait développer sur le projet.

4.2 Correction des erreurs bloquantes

Comme expliqué dans le chapitre précédent, certaines erreurs ont été détectées lors du développement de l'application.

La PIV ne s'effectue pas Pour que l'analyse PIV puisse se terminer, il faut lui fournir la configuration de la caméra et des balises telle que l'utilisateur l'aura paramétré dans l'application. Ici, la configuration de la caméra n'était jamais donnée en argument à la PIV. Il a été nécessaire d'associer la configuration donnée par l'utilisateur du projet en cours à la PIV pour que celle-ci puisse démarrer.

Mauvaise conversion des balises Après analyse, la PIV recevait des coordonnées faussées de la zone de recherche. Afin de solutionner ce problème, il y a eu lieu d'inverser la coordonnée verticale pour obtenir les bonnes coordonnées en pixels à donner à PyORC pour l'analyse PIV.

Association des balises Deux choix sont possibles pour solutionner cette erreur :

- Inverser les placeholders¹ entre les champs de saisie
- Inverser l'ordre des distances passées en paramètre pour la PIV

Ici le deuxième choix est le plus judicieux car il ne modifie pas l'interface qui est ainsi plus intuitive, et au niveau du code il n'implique qu'un seul changement là où la première solution implique une refonte des composants de l'écran et de toutes les variables liées.

4.3 Développement des fonctionnalités nécessaires

Après correction des erreurs bloquantes, l'application ne constituait toujours pas un MVP. Pour en arriver à un MVP tel que souhaité, les fonctionnalités suivantes devaient être implémentées.

Paramètres hardcodés Comme développé dans le chapitre précédent, trois paramètres étaient encore entrés en dur, alors que l'utilisateur doit pouvoir les modifier en fonction de la configuration de sa prise de mesures. Ces paramètres ont été intégrés dans l'interface comme suit :

1. Indication textuelle qui apparaît dans un champ de saisie vide pour indiquer à l'utilisateur ce qu'il doit entrer comme valeur[21]

- **Position de la caméra** : Ajout d'un champ sur l'écran de la configuration de la vidéo, ainsi que dans le fichier de configuration de chaque projet.
- **Vecteur d'affichage des transects** : Ajout d'un widget de sélection du vecteur à l'aide du même système que pour les balises, l'utilisateur doit déplacer les deux extrémités d'un vecteur pour le définir. Les coordonnées du vecteur sont maintenant enregistrées sous le nom "local_points" dans le fichier de configuration de chaque projet.
- **Coefficient de surface** : Ajout d'un champ sur l'écran de configuration de la bathymétrie, ainsi qu'un champ dans le fichier de configuration de chaque projet.

Affichage des transects Une fois la PIV terminée, il faut utiliser le fichier qui résulte de celle-ci pour afficher les vecteurs de vélocimétrie sur l'image de la rivière. Il faut aussi calculer le débit théorique sur le vecteur défini par l'utilisateur. Pour ce faire on va utiliser les fonctions développées lors de la première utilisation de PyORC. A travers ces fonctions on va lire le dataset fourni par le fichier "piv.nc" produit par la PIV, et on va appliquer des filtres de manière à obtenir et afficher le champ de vecteurs (voir fig. 4.1a).

Le débit est également calculé grâce à la bathymétrie mise en parallèle avec la vélocimétrie observée sur le vecteur des transects. Pour que cela soit affiché pour l'utilisateur, le contenu de l'écran est remplacé par l'image affichant les débits, ainsi que le débit obtenu, comme illustré par la fig. 4.1b



(a) Champ de vecteurs tel que calculé par PyORC (b) Écran de fin contenant le vecteur et le débit obtenu

FIGURE 4.1 – Illustrations du champ de vecteur et de l'écran de fin de RiverApp

4.4 Testing

Le testing est une partie fondamentale dans le cadre du développement d'une application, il permet d'assurer une qualité constante durant le développement et de garantir que des changements au code n'ont pas impacté son bon fonctionnement. La couverture de test étant presque inexistante (se référer au chapitre 3), il était impératif de munir RiverApp d'un testing concret et applicable dans le cadre d'intégration et de déploiement continus.

Tests unitaires Les tests unitaires ont été réalisés avec la librairie python pytest. Celle-ci nous permet de tester efficacement chaque fichier, ainsi que de lancer tous les tests d'une seule commande. Les fonctions suivantes sont maintenant testées :

- Fonctions de gestion des données des projets
- Fonction de détection automatique des balises
- Fonctions de calcul des vecteurs et de débit
- Fonctions utilitaires de conversion de format d'image

Ces tests offrent une couverture satisfaisante des fonctions énoncées ci-dessus, pour une couverture globale de 75% des fonctions du projet, comme le montre le tableau 4.1. Le reste des fonctions n'étant pas testées soit à cause des librairies utilisées pour le front-end et l'api, soit parce que ces fonctions ne sont pas utilisées dans l'application.

Fichier	Couverture (%)
src/api/app.py	44
src/back/gcp_detection/gcp_detection.py	84
src/back/mask.py	100
src/back/project_data_management.py	72
src/back/saving_project_data.py	97
src/back/transect.py	100
src/back/video_filters.py	0
src/utils/image_generator.py	53
src/utils/utils.py	70
Total	75

TABLE 4.1 – Pourcentages de couverture des tests

Tests end-to-end Le fonctionnement de l'application dépendant majoritairement des fonctions utilisant PyORC et des fonctions "back-end", l'accent a été mis sur les tests unitaires de celles-ci.

4.5 Constitution d'un dataset (prise de mesures)

Pour pouvoir tester le bon fonctionnement de l'application, ainsi qu'établir sa précision et les résultats que l'on peut attendre de celle-ci, il est nécessaire d'établir un dataset de mesures pour tester l'application dans différents contextes. Un set de mesures doit comprendre les informations suivantes :

- Vidéo d'un angle quelconque, duquel il est possible de voir la zone de recherche dans sa totalité et sans obstacle entre la caméra et la rivière. Un set contient idéalement plusieurs points de vue pour tester la cohérence des différents résultats
- Balises GCP (Ground Control Point) au nombre de 4
- Distances entre chaque balise, ou position GPS des balises pour pouvoir calculer ces distances
- Position de la caméra par rapport au point bas gauche de la zone de recherche, ou position GPS pour pouvoir calculer la distance
- Mesure de la bathymétrie sur la largeur de la rivière, au plus de points au mieux
- Mesure du débit sur la section que l'on analyse, à des fins de comparaison²

La prise de mesure nécessite actuellement plusieurs conditions pour être menée à bien et être utilisable pour l'application :

- Être en possession d'un outil de prise de position GPS
- Conditions climatiques : il n'est pas possible de prendre des mesures durant l'hiver sans matériel adapté, ni de prendre des mesures s'il pleut, la pluie perturbant la surface et donc l'analyse PIV
- Être en possession d'un courantomètre ou d'un débit calculé en temps réel par un organisme tiers (exemple : l'hydrométrie de la Fédération Wallonie-Bruxelles)
- Obligation d'être à plusieurs pour la prise de mesures. L'installation du setup et la prise de mesures en elle-même n'est réalisable avec une seule personne que pour des cours d'eau à très faible débit.

Trois sets de mesures ont été réalisés lors de cette année scolaire 2023-2024. Les résultats de ceux-ci seront abordés plus tard dans le chapitre 6.

Un protocole de prise de mesure plus précis avait été mis en place par P. Laurent et C. Dufromont, et a été mis à jour par A. Mignot pour inclure le kit d'orthorectification développé par ses soins. Celui ci est disponible à l'annexe D.

2. Attention de bien respecter les règles de prises de mesures avec points fixes en fonction de la largeur du cours d'eau analysé. Voir Annexe A : Prise de débit au courantomètre pour plus de plus amples informations

4.5.1 Petit Bocq

Lors d'une visite de sites avec le projet Rivialis³ et les étudiants du cours d'hydraulique fluviale, deux visites ont été réalisées sur le site du Petit Bocq à Hamois, la première consistant en du repérage pour le projet, et la deuxième en une prise de mesures durant laquelle un premier set a été réalisé.

Ce set contient une vidéo prise au drone du dessus de la rivière. Il est intéressant car il permet de tester la sensibilité du paramètre de la position de la caméra (ce paramètre étant encore inconnu et codé en dur au moment de la prise de mesures). Il est également utile pour tester une vidéo 4K sur l'application et sur une vidéo plus rapprochée de ce qui a déjà pu être abordé auparavant sur le projet RiverApp.

4.5.2 Limelette

Deux sets ont été réalisés à Limelette à des heures différentes de la journée, un le matin, l'autre le soir, ce qui permettra plus tard d'analyser la différence de résultats en fonction de l'ombre et des reflets sur l'eau.

Ces sets à Limelette contiennent aussi des tests sur un kit de mesure actuellement en développement.

4.5.3 Qualité du code

Un premier nettoyage du code a été effectué à partir des résultats fournis par pylint. Ce premier nettoyage a principalement consisté à supprimer du code non-utilisé (imports, variables, paramètres), supprimer des retours vides inutiles (return dans une fonction void) et adapter la longueur de lignes de code pour respecter les conventions de programmation. Comme expliqué, ces changements n'ont en rien impacté le fonctionnement du code, mais ils offrent une lecture et une compréhension plus aisée pour quiconque reprendra le projet. Le score pylint est maintenant à 9.81/10, et la plupart des alertes restantes concernent des "TODO"⁴.

3. Projet regroupant plusieurs startups (Kapta survey, Skalup, S&R), ainsi que plusieurs membres du personnel de l'UCLouvain et de l'ULiège. Le projet ayant la vocation de permettre de visualiser des cours d'eau en 3D, ainsi que d'autres informations sur celui-ci, pour permettre aux petites communes d'assainir les cours d'eau pour favoriser les écosystèmes de ceux-ci.

4. Commentaire à même le code, spécifiant un changement/ajout à faire dans le code à cet endroit.

Chapitre 5

Déplacement de l'effort de calcul sur serveur distant

5.1 API

Afin d'assurer la compréhension de tout lecteur de ce travail, voici une définition d'une API : "Interface de programmation à travers laquelle un logiciel offre des services à d'autres logiciels".[22]

5.1.1 Nécessité de l'API pour utiliser l'application sur téléphone

Une des vocations de l'application RiverApp est d'être finalement déployée sur téléphone dans le but de permettre à un utilisateur lambda de prendre une vidéo d'un cours d'eau et de l'analyser pour obtenir à terme un historique des débits de certains cours d'eau.

Les calculs effectués lors de l'analyse d'une vidéo sont lourds, que ce soit l'analyse PIV comme le calcul et l'affichage des transects. Pour un pc moderne (i5-1135G7, 12go de RAM) le calcul peut prendre jusqu'à plusieurs minutes pour la PIV, et quelques dizaines de secondes pour le calcul du débit et des transects. A partir de ce constat, il est évident qu'il ne sera pas possible d'effectuer ces calculs sur un téléphone moyen, ou en tout cas pas dans un temps raisonnable.

5.1.2 Quelles fonctionnalités migrer vers l'API ?

Comme expliqué dans la sous-section précédente, les calculs les plus lourds sont effectués lors de la PIV et du calcul des transects. Toutes les fonctions appelées lors

de ces opérations doivent être migrées sur serveur pour permettre une utilisation acceptable par un utilisateur. Ces fonctions sont les suivantes :

- Normalisation de la vidéo fournie à PyORC (partie de la vidéo sélectionnée par l'utilisateur)
- Génération du fichier de la PIV
- Ouverture du dataset généré par la PIV
- Application des masques sur le dataset
- Génération de l'image finale
- Calcul du débit

5.1.3 Testing

L'API consistant en un serveur Flask, le testing de celle-ci nécessite un environnement spécifique à rajouter aux tests actuels de l'application. Étant donné que les fonctions utilisées par l'API sont déjà testées et couvertes à près de 100% (Voir section 4.4), il a été jugé que les tests de l'API ne sont pas une priorité pour l'instant.

5.2 Solution de déploiement

Par facilité de déploiement, de documentation et de ressources, le choix de déploiement s'est porté sur un container Docker¹ contenant l'API Flask. La modularité de Docker constitue son avantage majeur dans le cadre du projet RiverApp, permettant à n'importe quel développeur de déployer l'API sans devoir installer manuellement tous les prérequis. Le dossier racine du projet contient un Dockerfile et un docker-compose qui, après utilisation de la commande suivante, créent un conteneur Docker comprenant l'application Flask allumée sur le port 5000 de la machine sur laquelle est hébergée l'API :

```
docker compose up --build -d
```

Le conteneur tourne sur un système d'exploitation Linux, distribution Debian 12 sur lequel python 3.10 est installé (l'application RiverApp et ses fonctions n'acceptent pas d'autre version de python). Sur ce conteneur, sont installées les différentes bibliothèques susceptibles d'être utilisées par PyORC. Le même fichier d'installation que celui d'une installation classique est employé pour l'automatisation

1. Docker est un outil qui permet de créer des conteneurs, qu'on peut comparer à des machines virtuelles mais en plus léger. Ces conteneurs permettent un déploiement automatisé sur n'importe quelle machine supportant Docker. Docker peut aussi être utilisé pour simuler des réseaux et des systèmes complexes. [23]

de l'installation. La librairie waitress est ajoutée pour servir de passerelle Web entre le serveur où est hébergé l'application et l'application elle-même.

L'application utilisant le port 5000 (modifiable si nécessaire, à déterminer en fonction du serveur sur lequel l'application Flask est hébergée), il est obligatoire que la machine hôte ait ouvert ce port. Si tel n'est pas le cas, l'utilisateur n'arrivera pas à se connecter et devra alors effectuer les calculs lourds localement.

5.3 Git flow et automatisatisation

Le contrôle de source et l'automatisation de certains processus font partie de la bonne réalisation d'un projet de développement. Les outils suivants ont été mis en place pour assurer une qualité continue de développement.

5.3.1 Git flow

Un des objectifs de ce mémoire est la mise en place d'outils pour faciliter et améliorer le développement de l'application (testing, documentation, runners). Il était donc logique d'instaurer un git flow cohérent avec celui-ci. Le projet n'a jamais eu de logique dans ses branches si ce n'est une branche par TFE, avec parfois quelques dizaines de commits sur la même branche. Le git flow instauré pour RiverApp est le suivant :

- **Branche main** : dev/main
- **Branches features** : dev/features/nom_de_la_fonctionnalité
- **Branches testing** : tests/nom_du_test
- **Branches documentation** : dev/docs/nom_de_la_documentation
- **Branches bugfix** : dev/fixes/nom_du_bugfix

Une fois qu'une fonctionnalité/test/bugfix/doc est implémentée et testée, la branche est mergée avec dev/main.

5.3.2 Automatisatisation

Concernant l'API, qui tourne sur un script python, celui-ci est lancé avec l'utilitaire Linux 'nohup', qui lui permet de se lancer en arrière-plan. Cela permet que le docker compose puisse se terminer sans attendre la fin de l'exécution du script.

Après quelques exécutions, il a été constaté que le serveur Flask produit une fuite de mémoire². Après quelques recherches sur les forums[25][26], il semble que les fuites de mémoire avec un serveur Flask sont fréquentes lors de grosses opérations.

Afin de régler cette fuite de mémoire, plusieurs tentatives ont été effectuées (différent type d'ouverture de fichiers, utilisation de la librairie gc - garbage collection), mais aucune de ces solutions n'a porté ses fruits.

Pour pallier à cette augmentation de l'utilisation de la mémoire, un cron³ a été configuré pour terminer le processus python en cours et le redémarrer toutes les heures, ce qui règle temporairement ce problème.

Le dernier point concernant l'automatisation n'est pas le moindre : un runner gitlab⁴ a été installé sur un serveur de l'EPL afin d'automatiser certaines tâches. Ces tâches sont les suivantes :

- **Exécution du linter Pylint** : l'exécution de pylint à chaque modification du code permet d'assurer que la qualité de celui-ci dépasse bien un certain seuil (défini dans la configuration du runner. Il est donc modifiable si l'on souhaite augmenter la qualité d'un cran)
- **Exécution de la suite de tests Pytest** : l'exécution des tests est encore plus importante que celle du linter, étant donné que ceux-ci vérifient le fonctionnement du code en lui-même. Si les tests ne passent pas, le développeur sait que son code ne fonctionnera pas en situation réelle. Il est donc nécessaire de fixer celui-ci avant de continuer à développer.

Le runner s'exécute actuellement en deux étapes, une pour le linter (rate si le score du code est en dessous de 9.5) et une pour les tests (rate si un seul test échoue). La figure 5.1 illustre la bonne exécution de ces étapes.

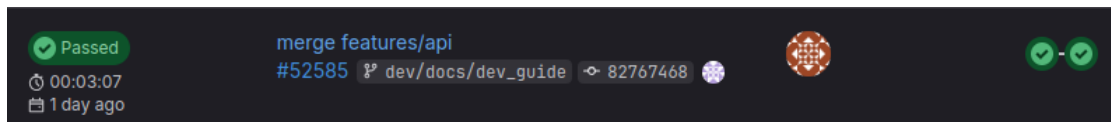


FIGURE 5.1 – Illustration d'une analyse positive du code après un commit quelconque

2. La fuite de mémoire est due à un manque de désallocation de la mémoire suite à une opération. Celle-ci cause un manque de mémoire pour les autres opérations en cours.[24]

3. Utilitaire linux permettant à l'utilisateur d'exécuter une commande donnée à intervalle régulier, configuré par le même utilisateur.[27]

4. Un runner est une application qui permet d'effectuer un travail donné à chaque fois que le code est modifié sur le repository gitlab. Ces travaux peuvent être constitués de scripts en tout genre, comme ceux développés dans ce paragraphe. [28]

Chapitre 6

Résultats

6.1 Résultats des anciennes versions

Le mémoire de Grégoire Debray et Dorian Hay[2] en 2022-23 mentionne les résultats de la première application RiverApp dont le moteur avait été développé par des étudiants[1], ainsi que les résultats de la première utilisation de la librairie PyORC. Ceux-ci sont repris dans la table 6.1 :

Jeu de données	Débit réel [m ³ /s]	Débit riverApp [m ³ /s]	Débit Py0rc [m ³ /s]
Noirath [1/1]	0.203	0.2285	0.246
VGC_1 [1/5]	1.26	0.0282	0.032
VGC_1 [1/3]	1.26	0.0036	1.081
VGC_1 [1/1]	1.26	0.0031	1.46

TABLE 6.1 – Débits estimés par les anciennes versions de l'application

On peut observer dans ceux-ci que le moteur de calcul développé par les étudiants produit des résultats qui ne sont pas assez consistants et trop écartés de la réalité lorsque le point de vue de la caméra demande une d'orthorectification trop importante pour fonctionner correctement.

En revanche, les quelques tests lancés avec PyORC semblent se rapprocher de la réalité lorsqu'on analyse une plus grande portion des images fournies à la librairie.

6.2 Résultats de la nouvelle application

6.2.1 Débits obtenus et observations

Toutes les mesures chiffrées sont disponibles dans l'annexe B, qui regroupe tous les tableaux de données utilisés pour produire les graphes présentés ci-dessous.

Note : tous les débits calculés par RiverApp ont été comparés soit avec des mesures prises par dilution de sel, soit sur le site de l'hydrométrie en Wallonie[29] (Aqualim + Infocruce, portail de l'hydrométrie en Wallonie[30])

Limelette - vidéo prise du milieu du pont Lors des mesures effectuées à Limelette, les vidéos du milieu du pont donnent les résultats les plus satisfaisants (pour ce setup de mesures). On peut observer ici que le débit calculé par RiverApp est par défaut proche de la réalité, et tend à s'en rapprocher encore lorsqu'on fournit plus d'images à l'analyse PIV, voir figure 6.1 :

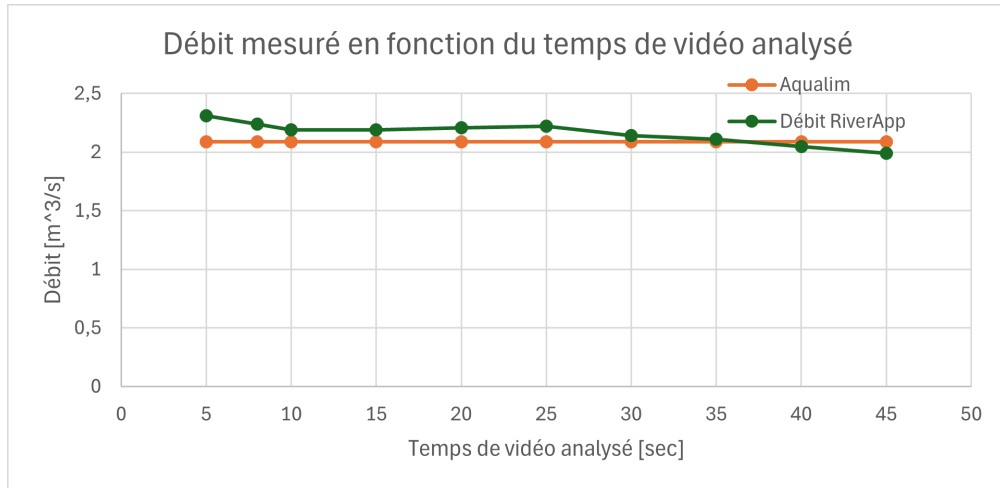


FIGURE 6.1 – Débit en fonction du temps de vidéo analysé - Limelette milieu

Limelette - vidéo prise de la gauche du pont

Comme le montre la figure 6.3, les résultats sont moins encourageants pour les vidéos prises de points de vue moins centraux. La précision du débit calculé par RiverApp semble régresser (ou du moins ne pas évoluer) lorsqu'on donne plus d'images à analyser à la PIV. (Illustration du point de vue sur la figure 6.2)



FIGURE 6.2 – Point de vue de la gauche du pont

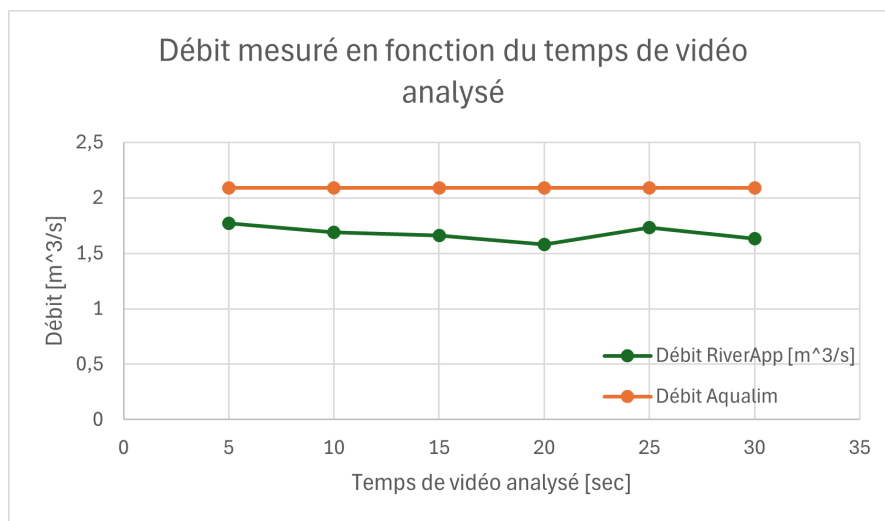


FIGURE 6.3 – Débit en fonction du temps de vidéo analysé - Limelette gauche

Un point non négligeable de cette vidéo est qu'une partie de la rivière est dans l'ombre, ce point sera abordé dans la sous-section 6.2.2 - Paramètres d'influence.

Limelette - vidéo prise de la droite du pont

Pour contraster avec les résultats observés jusqu'ici, les résultats affichés sur la figure 6.5 s'éloignent du débit réel lorsqu'on analyse plus d'images, ce qui paraît contre-intuitif. En effet, on pourrait croire qu'en analysant plus d'images, le résultat serait plus proche de la réalité. (Illustration du point de vue sur la figure 6.4)



FIGURE 6.4 – Point de vue de la droite du pont

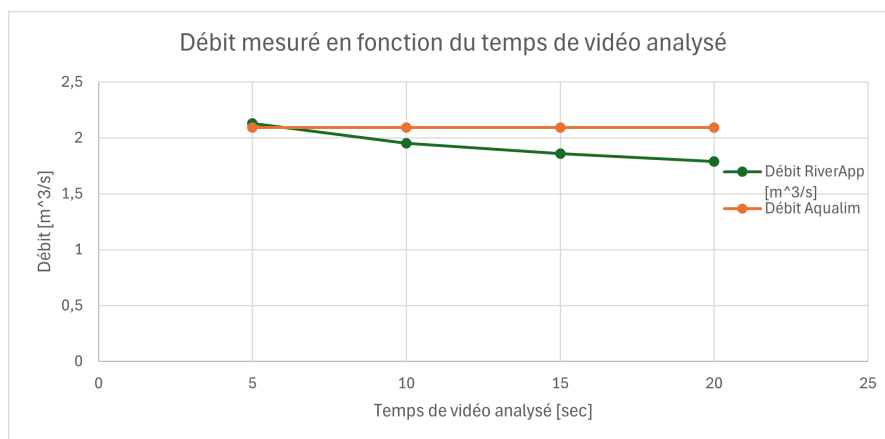


FIGURE 6.5 – Débit en fonction du temps de vidéo analysé - Limelette droite

Petit Bocq - vidéo à 90° au dessus de la rivière

Cette vidéo montre également des résultats satisfaisants par rapport au débit réel (Comme l'illustre la figure 6.7). Ces débits ont été rapportés par Martin Petitjean, assistant chercheur à l'iMMC, et ont été calculés par dilution de sel. Celle-ci a donné les résultats suivants : 363[l/s], 372[l/s], 377[l/s]. Cela donne environ 0.37m³/s. (Illustration du point de vue sur la figure 6.6)



FIGURE 6.6 – Point de vue à 90° au dessus du Petit Bocq

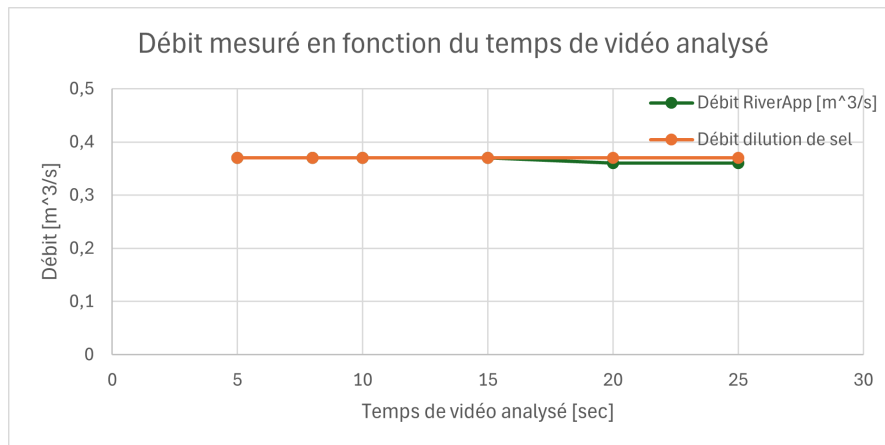


FIGURE 6.7 – Débit en fonction du temps de vidéo analysé - Petit Bocq 90°

Ceux-ci offrent une constance et une exactitude qui n'a pas été observée dans les tests sur les vidéos de la Dyle à Limelette.

Petit Bocq - vidéo à 45° au dessus en retrait de la rivière

Les tests de l'application sur la vidéo au dessus du Petit Bocq à 45° en retrait de la zone de recherche montrent également des résultats encourageants (fig. 6.9). Il semble toutefois qu'une analyse plus longue des données donne un résultat plus proche de la réalité. (Illustration du point de vue sur la figure 6.8)



FIGURE 6.8 – Point de vue à 90° au dessus du Petit Bocq

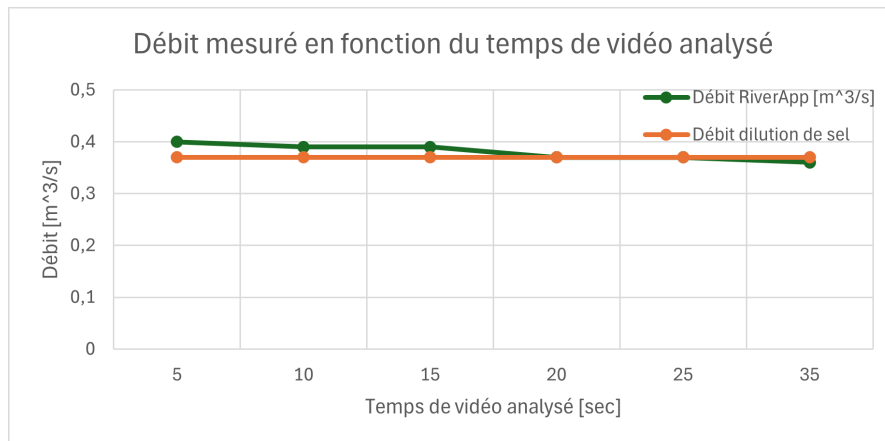


FIGURE 6.9 – Débit en fonction du temps de vidéo analysé - Petit Bocq 45°

Video Global Challenge Disposant de la vidéo du Video Global Challenge sur laquelle les étudiants du mémoire précédent avaient fourni les résultats repris dans la section 6.1, celle-ci a été testée avec la nouvelle interface. Ceux-ci portent sur moins de valeurs car la section de la vidéo comportant de l'ensemencement est trop courte pour obtenir plus de résultats. Cependant, certains de ces résultats ne sont pas mauvais pour autant, comme le montre la figure 6.10 :

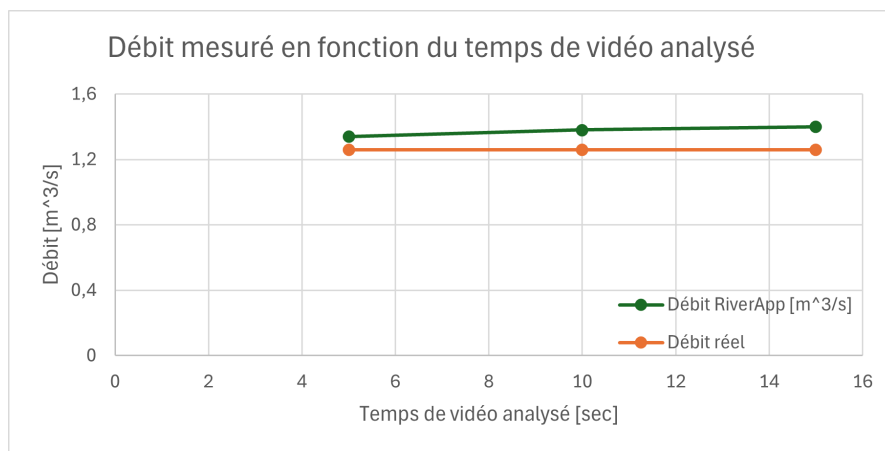


FIGURE 6.10 – Débit en fonction du temps de vidéo analysé - Video Global Challenge

Il est important de souligner que, pour cette vidéo, l'endroit de la rivière où a été pris la bathymétrie est inconnu, il a fallu choisir un endroit où placer le vecteur de calcul du débit. Ces résultats donnent donc une indication du débit calculé par

RiverApp sur cette vidéo, mais ne sont pas aussi précis que les débits cités plus haut.

6.2.2 Paramètres d'influence

Ensemencement L'ensemencement¹ est un des paramètres cruciaux pour le bon fonctionnement de l'analyse PIV. Des analyses d'échantillons sans ensemencement ont été réalisées sur les vidéos du Petit Bocq (pour rappel, le Petit Bocq est le meilleur résultat obtenu à ce jour avec RiverApp), et ceux-ci ont donné des résultats décevants. Pour avoir un ordre d'idée, les tests sur 15 et 35 secondes ont donné respectivement des marges d'erreur de 24% et 35%.

Les résultats sont légèrement meilleurs lorsqu'on teste la vidéo de la Dyle à Limelette. Mais les résultats ne sont pas suffisamment satisfaisants pour prétendre à une approximation du débit réel (20% de marge d'erreur pour 10 et 30 secondes analysées).

Résolution de la vidéo Le graphique représenté sur la figure 6.11 montre des résultats quelque peu surprenants quant à la précision de l'analyse en fonction de la qualité de la vidéo. En effet, on pourrait s'attendre à ce que la précision augmente en fonction de la hauteur de la résolution, mais il semblerait que cela ne soit pas tout le temps le cas. Lorsqu'on analyse une vidéo sur une courte période de 5 secondes, seules les vidéos de basse et très hautes résolutions donnent un résultat très proche de la réalité. En revanche, et cela semble plus logique, les résultats sont bien plus précis pour les hautes résolutions lorsqu'on analyse un échantillon plus long de la vidéo, ici 35 et 40 secondes (et inversement pour les basses résolutions).

Ce point sera abordé dans la suite de ce chapitre, mais il est important de garder en tête que plus l'échantillon est grand, plus l'analyse est chronophage.

Il est aussi utile d'insister sur le fait que si la résolution est trop basse, il n'est alors pas possible d'obtenir des résultats avec RiverApp. Le cas du drone Suptain SP650 le prouve. Sur deux vidéos, la résolution trop basse et de qualité médiocre n'a produit aucun résultat, si ce n'est un champ de vecteurs désorganisé et ne correspondant en rien à la réalité.

1. Pour rappel, l'ensemencement consiste à disposer des copeaux de bois (ou autres particules flottantes) pour simuler du remous sur la vidéo du cours d'eau

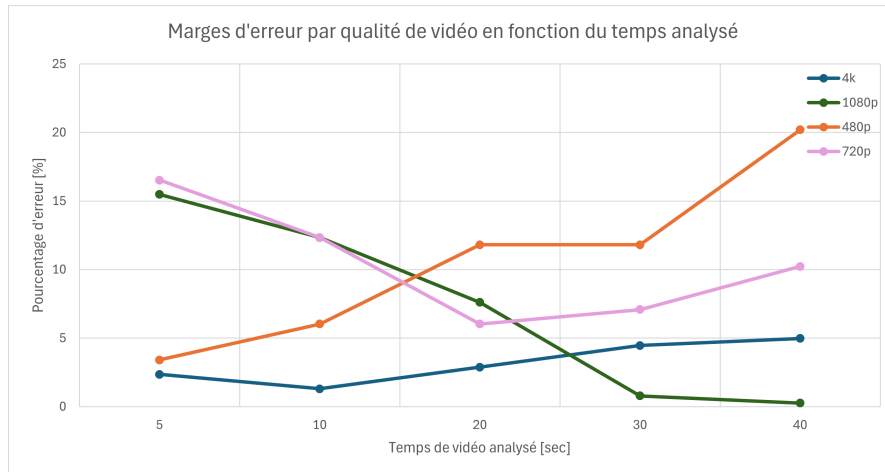


FIGURE 6.11 – Marges d’erreur par résolution en fonction du temps d’exécution sur la vidéo de Limelette - milieu

Éclairage Comme le montrent les figures 6.12a et 6.12b, PyOrc semble éprouver des difficultés à réaliser une analyse PIV correcte lorsqu’une zone de la vidéo est trop sombre. Il se pourrait que celle-ci soit alors considérée comme une partie de la berge ou comme une zone immobile du cours d’eau (Rocher, branche, ...). Cette zone non prise en compte pourrait expliquer le résultat plus faible pour un même débit réel (zone délimitée en rouge sur la fig. 6.12b).



(a) Résultat d’une piv sans ombre

(b) Résultat avec zone d’ombre

FIGURE 6.12 – Comparaison de deux résultats avec différents éclairages

Position de la caméra La position de la caméra par rapport à la rivière influence le résultat de l’analyse PIV de manière significative. On peut observer l’effet de ce paramètre sur les mesures de Limelette. Le paragraphe précédent explique pourquoi la vidéo prise de la gauche du pont n’est peut être pas à prendre en compte. En revanche, la vidéo prise de la droite du pont remplit les conditions d’une bonne

prise de mesure, mais les résultats diffèrent tout de même d'une vidéo centrée sur la rivière.

Précision de la `lens_position` Le paramètre "`lens_position`" est directement lié au paragraphe précédent. En effet, il permet à l'application de rectifier une certaine partie de l'image. Si celui-ci n'est pas correctement défini, l'image n'est pas rectifiée correctement et l'analyse PIV ne se fait pas proprement. Un résultat semblable peut être retrouvé sur la figure 3.4 dans le cas où le paramètre est faussé.

Précision des distances Encore une fois, dans le cadre de la rectification de l'image par PyORC, les distances entre les balises sont nécessaires afin d'obtenir le résultat le plus proche possible de la réalité. Si celles-ci ne sont pas correctes, ou diffèrent trop de la réalité, le résultat sera faussé comme le montre encore une fois la figure 3.4. Pour la précision des distances, il est recommandé de prendre la position GPS de chaque balise pour avoir un résultat plus précis qu'une mesure à la main (sur les mesures effectuées cette année, une différence jusqu'à 20cm a été observée).

Position du vecteur de la bathymétrie Le positionnement du vecteur sur lequel RiverApp va calculer le débit final est un paramètre décisif. En effet, il est obligatoire de placer ce vecteur à l'endroit de l'image où la bathymétrie a été calculée, sans quoi les calculs seront faussés et le débit obtenu ne sera pas celui que l'on cherche.

Distance par rapport au cours d'eau La précision des résultats sur les tests des vidéos du Petit Bocq soulèvent la question de savoir si une vidéo plus proche du cours d'eau produit de meilleurs résultats. Sur le petit échantillon disponible avec la Dyle et le Petit Bocq, il semblerait que cela soit le cas.

Cette constatation semble logique lorsqu'on part du principe que l'analyse parcourt chaque pixel de la zone de recherche. Si on est plus proche de la surface, les patterns de remous contiennent plus de pixels, ce qui entraîne une analyse plus précise de la vélocimétrie en surface.

6.2.3 Temps d'exécution

Les différents paramètres peuvent potentiellement tous faire varier le temps d'exécution de l'analyse PIV. Les plus importants sont le temps de vidéo analysé et la résolution de la vidéo. La figure 6.13 montre les différences significatives entre les différentes résolutions. Les temps d'exécutions suivent une certaine logique dans

le sens où plus la résolution est haute, plus il y a de pixels à analyser et plus l'exécution sera longue.

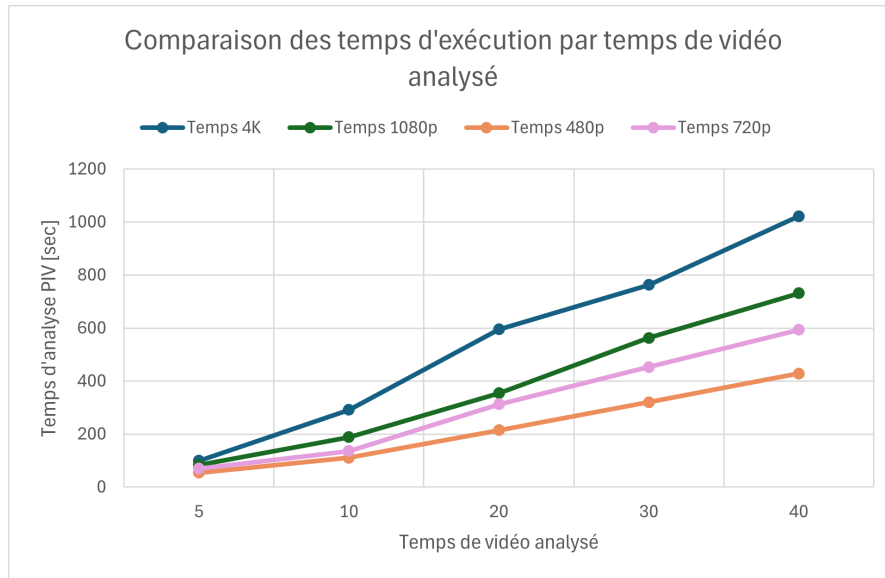


FIGURE 6.13 – Comparaison des temps d'exécution en fonction des temps d'exécution sur la vidéo de Limelette - milieu

6.2.4 Détection des balises

La détection des balises est une fonctionnalité qui doit manifestement être revue. Celle-ci ne fonctionne correctement que lorsque la vidéo est prise du dessus de la rivière et que les balises sont visibles à 100%. La figure 6.14 illustre les deux cas de figures en question.

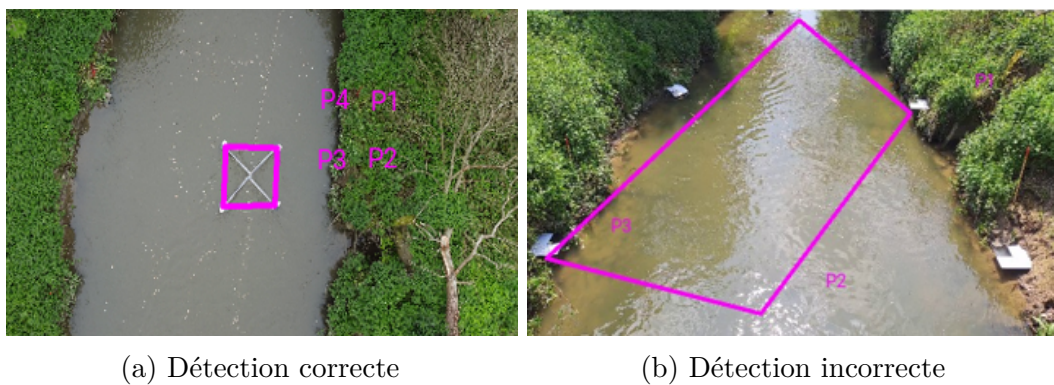


FIGURE 6.14 – Illustrations de la détection des balises

Lors des nombreuses analyses lancées pour obtenir les résultats explicités dans ce chapitre, la détection des balises a montré des faiblesses lorsqu'on tente d'analyser une vidéo dans laquelle il n'y a pas de balises. Le comportement attendu lorsqu'on fournit une vidéo sans balises est de faire apparaître 4 points sur l'écran, de manière à ce que l'utilisateur puisse configurer la zone à la main. Cependant, il arrive que la détection ne fasse pas apparaître ces points et provoque un arrêt de l'application.

6.3 Comparaison & Conclusion

Comparaison avec les anciens résultats Les résultats de la nouvelle interface montrent des chiffres moins volatiles que les résultats exprimés dans la section 6.1. En effet, pour le moteur de calcul artisanal, on peut voir que les débits varient de manière extrême en fonction du jeu de données fourni. Il en va de même pour PyORC, contrairement aux résultats explicités dans la section 6.2, qui montrent des résultats constants, pouvant donner une idée assez précise du débit réel. Ce qui n'était pas possible auparavant au vu des résultats des versions précédentes.

Résultats encourageants Les résultats de l'application actuelle permettent dans tous les cas d'avoir une idée se rapprochant de la réalité, même dans le cas où tous les paramètres d'influence ne sont pas optimaux. Le fait d'avoir pu obtenir exactement le même résultat que le débit réel sur plusieurs cours d'eaux (moyennant l'utilisation des bonnes vidéos) affirme le bon fonctionnement de l'application, ainsi qu'une nette amélioration par rapport au moteur de calcul du mémoire de 2022-2023[2]. L'intérêt de ces résultats est surtout de savoir quels paramètres font diverger les résultats d'une analyse.

Il est tout de même nécessaire de mener des analyses plus approfondies de la nouvelle version afin de déterminer comment contrer ces paramètres ayant une influence négative sur les résultats.

La vidéo "parfaite" La vidéo qui permettrait la meilleure analyse est celle qui contiendrait les facteurs suivants :

- Résolution en 1080p : C'est la résolution qui offre le meilleur rapport résultats/temps d'exécution
- Vidéo prise du dessus de la rivière, ou qui est centrée par rapport à celle-ci
- Pas de zones d'ombres ni de reflets
- Balises visibles à 100%
- Distance minimale entre la caméra et le cours d'eau, idéalement toute la surface visible sur la vidéo est la zone de recherche
- Distances entre les balises mesurées au GPS (lens_position comprise)

- La vidéo contient de l'ensemencement (copeaux de bois, frigolite - à éviter pour des questions environnementales, autres particules)

Ces facteurs semblent offrir les meilleurs résultats avec l'application. Il est bien sûr possible d'obtenir des résultats tout à fait cohérents si ceux-ci ne sont pas respectés à la lettre, mais la marge d'erreur sera plus grande.

Temps d'exécution Il est clair que le rapport temps d'exécution / précision des résultats doit rester le plus rentable possible pour l'utilisateur final. Il est donc hors de question de proposer à l'utilisateur des analyses durant plusieurs dizaines de minutes. Il est nécessaire d'établir la limite maximale d'exécution autorisée pour un utilisateur, ou du moins de le prévenir du temps qu'il risque d'attendre en fonction de sa demande.

Par exemple si l'on veut un résultat rapide et que la marge d'erreur acceptable est plus flexible, il est judicieux de conseiller à l'utilisateur une vidéo de plus petite résolution et de courte durée et inversement si l'utilisateur souhaite un résultat précis.

Conclusion La nouvelle version de l'application fournit des résultats satisfaisants si les bons paramètres sont utilisés. De nouvelles analyses des différents facteurs critiques devraient encore permettre d'améliorer ces résultats.

Chapitre 7

Documentation

La documentation d'une application a deux objectifs :

- Assurer que l'utilisateur se sert de l'application comme les développeurs l'ont conçue.
- Assurer que la transmission de savoir entre développeurs se fasse correctement. Cela permet à quiconque souhaiterait modifier ou ajouter des fonctionnalités à l'application de le faire le plus aisément possible.

En l'état de sa réception en septembre, l'application possédait de la documentation comprenant seulement la procédure d'installation de l'application ainsi que quelques bases sur l'utilisation de Kivy. Cette documentation est générée grâce à l'extension python Sphinx, qui permet de convertir des fichiers RST (reStructured-Text) en fichiers html. Cela offre un formatage automatique de la documentation, respectant les conventions et faisant gagner un temps non négligeable sur la rédaction[31].

La documentation a maintenant été étayée avec des guides pour le développement et l'utilisation de l'application. Ces parties sont détaillées ci-dessous. Ces nombreux ajouts à la documentation ont également été développés avec Sphinx.

7.1 Guide développeur

Documentation de l'API La documentation de l'API contient dans un premier temps les explications par requêtes tel que l'on a l'habitude de le voir dans une documentation REST¹, à savoir :

- Exemple de l'url de la route

1. Representational State Transfer : ensemble de contraintes architecturales qu'une API doit respecter. [32]

- Définition des différents paramètres
- Exemple de paramètres
- Réponses attendues et significations

Vient ensuite la procédure d'installation de l'API sur une machine quelconque. Cette procédure a été testée (et corrigée) en conditions réelles lors de l'installation sur le serveur de l'EPL. Elle est donc correcte si l'on suit les consignes à la ligne.

La dernière section de cette partie de la documentation porte sur la sécurité de l'API, et contient quelques recommandations.

Installation du runner Comme expliqué dans la sous-section 5.3.2, un runner gitlab a été installé cette année pour assurer une qualité de développement supérieur à ce que le projet a pu connaître dans le passé. Toute la procédure d'installation est expliquée étape par étape. Elle permet à n'importe quel utilisateur disposant des droits de responsable du projet d'installer un runner sur sa machine.

Documentation par écran Chaque écran est documenté de manière à ce qu'un nouveau développeur du projet puisse comprendre rapidement le fonctionnement global de l'application. Ce guide contient une page par écran qui détaille chaque fonctionnalité de l'écran, les interactions entre les différentes fonctions de celui-ci et d'autres fichiers potentiels. Ainsi qu'un récapitulatif de chaque fonction (les détails de chaque fonction sont spécifiés directement en commentaire dans le code). Ce guide est destiné à être étayé pour chaque ajout dans le code de l'application.

Diagramme du fonctionnement de l'application Afin de faciliter la compréhension de l'application pour les futures personnes devant développer dessus, la documentation dispose d'un diagramme représentant le fonctionnement de l'application. Celui-ci montre les écrans et le cheminement logique entre eux, ainsi que les fichiers dont le front va appeler les fonctions pour son bon fonctionnement. Le diagramme est affiché à l'annexe C pour avoir une idée de sa représentation. Il est visualisable sur la documentation de l'application en ligne.²

7.2 Guide utilisateur

Documentation de l'application Cette partie de la documentation est destinée à un utilisateur lambda, pour qui l'application serait déjà installée, mais également au développeur qui découvrirait l'application. Celle-ci reprend chaque écran et

2. https://sites.uclouvain.be/hydraulics-group/riverapp/dev_guide/diagram.html

décrit son fonctionnement, ainsi que les informations attendues par l'utilisateur pour obtenir une analyse du débit théorique de la rivière. Chaque écran est illustré afin d'offrir à l'utilisateur un exemple d'utilisation et rendre la documentation claire.

Guide de prises de mesures Un utilisateur lambda voudra plus que probablement tester l'application avec son propre set de mesures. La documentation lui fournit un guide reprenant le matériel nécessaire, ainsi que les étapes de prise de mesures nécessaires pour obtenir toutes les informations dont l'application a besoin pour obtenir un débit cohérent. Le guide contient également une checklist plus concise pour la prise de mesures.

Docstrings Les docstrings sont des chaînes de documentations à même le code qui permettent de comprendre facilement un morceau de code (généralement classes & fonctions). Elles sont généralement utilisées pour le partage de code et les travaux en groupe, mais elles permettent aussi au développeur de se remémorer son code lorsqu'il doit revenir dessus.[33]

Le projet tel que reçu en septembre dispose de quelques fonctions comportant des docstrings (majoritairement le fichier contenant les fonctions de génération d'images et quelques classes du front-end), mais une grande partie demeure non documentée (il manque précisément 33 docstrings selon pylint).

Au vu de ce manque de documentation du code en lui-même, les fonctions du back du projet ont été munies de docstrings pour toutes les fonctions qui n'en disposaient pas encore. Sphinx permet de générer automatiquement de la documentation à partir de ces docstrings, offrant une interface plus lisible qu'à même le code dans un environnement de développement. Cette auto-documentation a également été mise en place pour toutes les docstrings écrites.

Les docstrings ajoutées à l'application ont été générées via l'intelligence artificielle Google Gemini³. Celles-ci ont ensuite été corrigées et formatées pour correspondre à la réalité de l'application, et permettre de générer automatiquement la documentation citée ci-dessus.

3. Google Gemini est une intelligence artificielle (IA), générative et multimodale, de type transformeur, créée par Google.[34] Disponible au lieu suivant : gemini.google.com

Chapitre 8

Pistes d'amélioration

8.1 Développement de l'application

Le développement de l'application a été effectué jusqu'à compléter le stade de MVP, ce qui implique qu'il reste de nombreuses améliorations à effectuer pour arriver à une version livrable au grand public. Voici une liste d'idées pouvant améliorer l'expérience de l'utilisateur, ainsi que les résultats de l'application.

Bugs graphiques Il existe plusieurs bugs graphiques sur l'application actuelle, voici ceux qui ont été notés durant la réalisation de ce mémoire :

- **Étirement de la fenêtre** : lors de l'étirement de la fenêtre de l'application, les écrans contenant des images déplacent les composants de manière désordonnée, laissant apparaître des composants cachés, etc... Pour les composants cachés, une solution simple est de les supprimer de l'écran lorsqu'ils ne sont plus utilisés.

Concernant la disposition des composants lorsqu'on varie la taille de fenêtre, il est possible de jouer avec la positions de ceux-ci dans le fichier de configuration Kivy.

Pour l'écran des balises, les 4 points se déplacent en dehors de l'image et sont alors difficilement atteignables par l'utilisateur

- **Clic droit** : lorsque l'utilisateur effectue un clic droit sur n'importe quel écran de l'interface, un rond rouge transparent apparaît sur l'écran et ne disparaît qu'après un redémarrage de l'application. (cela n'est probablement pas un bug mais plutôt une fonctionnalité non désirée de kivy). Une simple recherche permet de trouver des solutions à ce problème : il s'agit de la "multi-touch emulation"[35], la solution se trouve dans la configuration de Kivy.
- **Gestion des projets** : lorsqu'il n'existe encore aucun projet, que l'on

créée un projet et qu'on revient aussitôt en arrière, le label expliquant qu'il n'existe pas encore de projet est encore affiché (alors qu'il existe bel et bien un projet). Encore une fois, il suffit de faire disparaître le label de l'écran via un appel à une fonction Kivy lorsqu'on crée un projet.

Différentes tailles d'écran Kivy offre 3 tailles d'écrans possibles afin que l'application développée soit graphiquement adaptative (responsive¹) en fonction de l'appareil utilisé : mobile, tablette et ordinateur. L'application n'est développée que pour correspondre à la taille tablette. Il est donc nécessaire de développer (au moins) la taille mobile si l'objectif est de déployer l'application sur mobile dans le futur.

Gestion des threads avec Kivy L'interface Kivy est sensible à l'utilisation des threads. En effet, comme utilisé dans l'application, un thread s'occupe de la gestion de l'interface, il est alors possible d'utiliser d'autres threads pour effectuer des tâches lourdes. Le problème soulevé ici est que le thread de l'interface va attendre la fin de l'exécution des autres threads pour continuer sa propre exécution. Pour la PIV et les transects, la fenêtre Kivy va se geler en attendant, laissant croire à un crash de l'application. Si l'on veut déployer l'application sur mobile, il sera nécessaire de corriger cette utilisation avant le déploiement. Pour cela, il faut gérer la gestion d'événement autrement qu'en appelant un autre thread, sous peine de bloquer le thread principal de Kivy. Si ce n'est pas possible, l'utilisation de Kivy pourrait-être remise en question pour l'interface.

Filtres PIV L'application dispose déjà de l'écran pour afficher les prévisualisations des applications de filtres sur la vidéo, avant d'exécuter la PIV. Cependant, ces filtres ne sont pas fonctionnels. Il serait intéressant de développer cette fonctionnalité pour analyser à quel point ces filtres influencent en bien, ou en mal, les résultats de l'analyse menée par PyORC. Un exemple de filtre qui pourrait donner un résultat différent est un filtre de contraste. En faisant ressortir les contrastes des patterns de particules mobiles sur l'eau, l'analyse pourrait potentiellement arriver à mieux les suivre et donc à donner des résultats plus précis. Chaque filtre qui pourrait modifier la perception de l'image par l'analyse PIV pourrait alors affiner (ou inversement dégrader) les résultats de celle-ci.

Améliorations de l'interface L'interface actuelle est fonctionnelle (c'est ce qu'on attend en premier lieu d'un MVP), mais elle pourrait être plus intuitive.

1. Une application dite "responsive" signifie que la taille de la fenêtre est adaptative en fonction de l'écran de l'appareil sur lequel elle est affichée.

Voici quelques propositions d'ajouts qui amélioreraient et fluidifieraient l'utilisation de l'interface :

- Pouvoir passer de case en case en utilisant la touche TAB du clavier, sur les écrans contenant plusieurs champs où il faut entrer des valeurs
- Pouvoir retourner en arrière sur les dernières phases de l'application
- Pouvoir relancer le calcul du débit et l'affichage du transect, en modifiant le vecteur via l'écran dédié
- Laisser à l'utilisateur le choix de lancer la PIV sur le serveur ou non (et griser le bouton serveur si Internet n'est pas disponible, ou si l'API n'est pas joignable)
- Lorsqu'il faut choisir un fichier, démarrer la recherche de fichier depuis la racine du dossier où est installée l'application
- Pouvoir annuler l'analyse PIV une fois qu'elle est lancée, et retourner directement à l'écran précédent
- Pouvoir mettre les points des balises dans n'importe quel sens sans que cela fausse le résultat de l'analyse

Qualité du code le code actuel contient de la programmation défensive (assertions², exceptions) pour une bonne partie des fonctions. Cependant, il existe des fonctions qui ne sont pas du tout protégées, et les exceptions existantes ne couvrent pas tous les cas possibles. Il serait intéressant d'uniformiser le code à ce niveau pour prévenir des erreurs lors du développement.

Organisation des fichiers de l'application comme expliqué dans la section 3.4, l'organisation des fichiers nécessite un remaniement pour être plus cohérente. Certaines fonctions du back-end devraient soit avoir leur propre fichier soit retourner dans le front-end. Il est clair qu'elles ne doivent pas être là où elles sont. Une analyse et un refactoring sont nécessaires.

Exemple : la fonction "generate_piv" se trouve actuellement dans le fichier "saving_project_data", fichier sensé gérer la sauvegarde des données dans le fichier de configuration du projet concerné. Il est évident que cette fonction n'a rien à faire dans ce fichier et doit-être déplacée soit dans le fichier de l'écran en question, soit dans un fichier dédié dans le back-end.

Détection des balises La détection des balises ne fonctionne actuellement à 100% que dans le cas où la vidéo est prise à 90° au dessus de la zone de recherche, et que les balises sont totalement visibles. Si aucune balise n'est présente sur l'image, il y a de grandes chances pour que l'application cesse de fonctionner, il

2. Une assertion consiste à vérifier que l'information que l'on va traiter est au bon format, avant de lancer le traitement, pour prévenir d'un bug

faut impérativement régler ce problème, quitte à ce que l'utilisateur doive mettre les balises à la main. Le problème est dû au fait que si la détection renvoie une liste de balises en contenant moins que 4, l'application s'arrête, il faut donc au moins renvoyer une liste de bonne taille (et ce quitte à renvoyer les mauvaises position).

8.2 API

L'API n'étant qu'à la première version de son développement, elle n'est pas encore totalement au point. Voici une liste de suggestions et de problèmes à corriger pour un fonctionnement optimal de l'API.

Memory leak Comme expliqué dans la sous-section 5.3.2, l'utilisation de Flask entraîne une fuite de mémoire non contrôlée. L'API nécessite une analyse approfondie de cette fuite de mémoire afin de pouvoir cerner d'où elle provient et s'en débarrasser. Il reste une piste qui n'a pas encore été testée, celle de supprimer le fichier vidéo directement après l'analyse. En effet, il est probable que celui-ci reste dans le cache de l'application Flask, entraînant un memory leak conséquent pour chaque opération réalisée sur l'API. Si cela ne résout pas le problème, il est possible que le problème vienne de la technologie de Flask en elle-même, un développement sur une autre technologie sera alors nécessaire.

Testing Actuellement, l'API en elle même n'est pas testée. Les fonctions utilisées par l'API pour le calcul de la PIV et l'affichage des vecteurs sont testées, mais le processus d'appel aux routes depuis l'application RiverApp ne l'est pas. Afin de garantir la qualité du développement de l'API, il est important de mettre du testing en place. Le testing d'une application Flask est possible via pytests, librairie déjà utilisée pour le testing de l'application client de RiverApp.³

Chiffrement de l'API key La sécurité de l'API est uniquement assurée par une clé API qui permet à Flask d'authentifier l'utilisateur qui souhaite utiliser ses routes. Étant donné que l'API est uniquement en HTTP pour l'instant, quiconque étant sur le réseau peut intercepter la clé API et l'utiliser à ses fins. Dans un premier temps, il est donc nécessaire de chiffrer la clé lors de l'envoi sur le réseau et de la déchiffrer lors de la réception dans l'API.

Actuellement, l'API est hébergée sur un serveur de l'UCL qui n'est accessible que depuis le réseau de l'Université, ce qui signifie qu'il faut des accès spécifiques

3. Voici des exemples de tutoriels pour réaliser ce testing : testdriven.io[36], Documentation Flask[37]

pour joindre ce réseau. L'ajout du chiffrement n'a donc pas été la priorité car le réseau de l'UCL est considéré comme fiable pour notre application.

Certificat SSL Afin d'améliorer la sécurité d'un cran supplémentaire, il serait souhaitable de passer le serveur sur lequel est hébergé l'API en HTTPS. Cela implique d'obtenir un certificat SSL et de l'installer sur la machine. Ce certificat permettra de sécuriser la connexion entre l'utilisateur et le serveur, empêchant quiconque de tenter d'écouter la communication.

Mode offline Pour l'instant, au moment des calculs lourds, l'application vérifie si elle est connectée à Internet. Si elle l'est, elle va utiliser l'API pour effectuer l'analyse PIV et le calcul des transects. Cependant, si l'API n'est pas disponible, l'application va tenter de s'y connecter. Il serait alors judicieux de créer une route qui permettra à l'application coté client de vérifier si l'API est bien apte à être appelée.

Optimisation de l'envoi de fichiers L'envoi du fichier de la vidéo à analyser est la partie la plus longue et lourde de l'exécution de la PIV sur serveur, notamment parce que le fichier est envoyé dans son entièreté. Il serait plus efficace d'envoyer uniquement la section de la vidéo qui sera utilisé pour l'analyse PIV, ce qui ferait gagner énormément de temps, surtout pour les utilisateurs ne disposant pas d'une connexion Internet suffisante pour l'envoi de gros fichiers. Cela permettrait également de gagner en espace de stockage sur le serveur hébergeant l'API.

8.3 Prises de mesures

L'application a été testée sur des cours d'eau de tailles semblables (quelques mètres de largeur, quelques dizaines de centimètres de profondeur maximale). Il serait intéressant de comparer ses résultats sur des cours d'eau de tailles plus extrêmes, comme des ruisseaux ou des fleuves, afin de tester les limites de PyORC.

Un autre point à améliorer est le matériel de prise de mesure. Actuellement, l'EPL fournit un drone Snaptain SP650 pour la prise d'images, ce drone est instable et la qualité d'image est médiocre, des tests ont été effectués sur les images de celui-ci et elles sont inutilisables. Il existe actuellement des drones accessibles qui fournissent une stabilité largement suffisante, ainsi qu'une qualité d'image allant jusqu'à 4K pour moins de 500€ (DJI Mini 2SE, DJI Mini 3). L'achat d'un tel drone permettrait d'obtenir des images de qualité pour chaque setup, moyennant des conditions météo correctes.

Le mémoire ayant lieu pendant l'année académique, les températures ne sont donc pas favorables à des prises de mesures en rivière. L'achat d'une paire de cuissardes de pêche (avec plastron) permettrait aux étudiants chargés des prises de mesures de les effectuer pendant une plus grande période de l'année, en dépit de la température de l'eau.

Les prises de mesures sont développées plus en détail dans le mémoire de A. Mignot[7]. Il y décrit la procédure étape par étape dans la section "4.2 - Mesures sur le terrain", et fourni des exemples pour faciliter la compréhension de ces étapes.

Chapitre 9

Conclusion

Cela fait maintenant des années que les techniques d'imagerie sont utilisées à des fins de mesures de débit pour permettre ces mesures en toutes circonstances (crues, étiages). C'est pourquoi, depuis quelques années maintenant, des étudiants réalisent des recherches sur les analyses PIV et LSPIV dans le cadre de leur mémoire à l'UCLouvain. Ce mémoire fait suite à la deuxième itération de la création d'une application de calcul de débit au sein de l'EPL.

L'application telle que reçue en septembre constitue une bonne base, malgré les problèmes de conception dans l'organisation des fichiers du projet, le fait qu'elle ne respectait pas le cahier des charges initial et qu'elle comprenait un nombre d'erreurs bloquantes non négligeable. La première (et majeure) partie de ce travail a consisté en l'analyse de cette version de l'application afin de cibler et régler ces erreurs, laissant moins de temps pour le développement à proprement parler.

Une fois l'analyse PIV et le calcul du débit développés, ces parties ont été portées sur serveur afin de pouvoir directement libérer nos machines de la puissance de calcul nécessaire pour réaliser ces opérations.

Le MVP étant fonctionnel, l'application a été testée sur les différentes mesures effectuées cette année académique afin de pouvoir comparer les résultats entre eux et ceux de RiverApp v1. Ces résultats s'avèrent être plus qu'encourageants et il est certain qu'ils peuvent être améliorés, moyennant l'analyse des différents paramètres critiques énoncés dans ce travail.

Dans un second temps, le développement d'une série de tests afin d'établir une couverture maximale sur l'entièreté du projet a été effectuée, dans le but de les appliquer automatiquement à chaque ajout de code sur le projet GitLab, ainsi que

l'analyse de la qualité du code.

Le projet a été doté d'une documentation, la plus complète possible, qui est vouée à évoluer avec l'application. Cette documentation fournit des guides destinés à l'utilisateur lambda, ainsi qu'au développeur, et contient toutes les informations pour installer et utiliser l'application. Quiconque souhaiterait modifier l'application pour une utilisation bien précise possédera les outils en main pour y arriver.

Les objectifs de développement cités dans l'introduction ayant été remplis, le projet est maintenant muni des outils nécessaires pour continuer un développement de qualité. Le projet dispose également de quelques sets de mesures permettant à la personne qui reprendra le développement d'effectuer des tests comparatifs avec la réalité sans devoir se mouiller.

Le développement n'est pas terminé. En effet il reste un certain chemin à parcourir pour que l'application soit entièrement fonctionnelle et utilisable par un utilisateur lambda depuis son téléphone (Voir Chapitre 9). Cependant, tout ce qui a été fait cette année devrait permettre que la suite de ce chemin soit sans embûche.

Il est important de noter que ce mémoire a été réalisé en collaboration avec Alexandre Mignot, qui travaille sur le développement d'un kit de mesures pour prendre des mesures avec RiverApp. Alexandre a fait figure d'utilisateur critique à de nombreuses reprises, permettant de gérer des cas qui n'auraient pas été détectés sans son aide (e.g. application utilisée sur différents systèmes d'exploitation, générant des erreurs).

Annexes

Annexe A

Prise de débit au courantomètre

Cette Annexe est tirée du Mémoire de A. Mignot[7], qui a basé ses recherches sur les pages 77 à 86 du livre Hydrometry de Boiten. [38]

Dans le cadre d'une prise de débit avec courantomètre, la précision de celui-ci dépend du nombre de mesures prise par rapport à la morphologie de la rivière. Dans le cas où la bathymétrie est trop irrégulière, un espacement plus adapté doit être réalisé mais en temps normal, les mesures verticales à prendre doivent être équidistantes sur la largeur.

Largeur du cours d'eau [m]	Nombre de mesures verticales
$b < 0.5$	3
$0.5 < b < 1.0$	4 à 5
$1.0 < b < 3.0$	5 à 8
$3.0 < b < 6.0$	8 à 12
$6.0 > b$	12 ou plus

TABLE A.1 – Nombres de points horizontaux à prendre lors d'une prise de débit au courantomètre

Concernant les mesures sur le plan vertical, le tableau est le suivant :

Hauteur d'eau [m]	Nombre de points de mesure
$d < 0.25$	1
$0.25 < d < 0.5$	2
$0.5 > d$	3 ou plus

En fonction des critères cités dans les tableaux ci-dessus, du temps disponible et de la précision souhaitée il existe plusieurs méthodes permettant de trouver la vitesse moyenne d'écoulement de l'eau.

1. Méthode à 1 point :

$$\bar{v} = v_{0.6} \quad (\text{A.1})$$

avec $v_{0.6}$ qui est la vitesse mesurée à 60% de la hauteur à partir de la surface de l'eau.

2. Méthode à 2 points :

$$\bar{v} = 0.5(v_{0.2} + v_{0.8}) \quad (\text{A.2})$$

avec $v_{0.2}$ et $v_{0.8}$ qui sont respectivement les vitesses mesurées à 20% et 80% de la hauteur à partir de la surface de l'eau.

3. Méthode à 3 points :

$$\bar{v} = 0.25v_{0.2} + 0.5v_{0.6} + 0.25v_{0.8} \quad (\text{A.3})$$

avec $v_{0.2}$, $v_{0.6}$ et $v_{0.8}$ qui sont respectivement les vitesses mesurées à 20%, 60% et 80% de la hauteur à partir de la surface de l'eau.

4. Méthode à 5 points :

$$\bar{v} = 0.1(v_s + 3v_{0.2} + 2v_{0.6} + 3v_{0.8} + v_b) \quad (\text{A.4})$$

avec v_s et v_b qui sont respectivement les vitesses mesurées à la surface de l'eau et au fond de l'eau.

Annexe B

Tableaux de résultats de la nouvelle interface

B.1 Débits limelette - vidéo du milieu du pont

Temps de vidéo analysé [sec]	Débits calculés par RiverApp [m ³ /s]	Débit Aqualim
5	2,31	2,09
8	2,24	2,09
10	2,19	2,09
15	2,19	2,09
20	2,21	2,09
25	2,22	2,09
30	2,14	2,09
35	2,11	2,09
40	2,05	2,09
45	1,99	2,09

TABLE B.1 – Débits calculés en commençant l'analyse au début de l'ensemencement

B.2 Débits limelette - vidéo de la gauche du pont

Temps de vidéo analysé [sec]	Débits calculés par RiverApp [m ³ /s]	Débit Aqualim
5	1,77	2,09
10	1,69	2,09
15	1,66	2,09
20	1,58	2,09
25	1,73	2,09
30	1,63	2,09

TABLE B.2 – Débits calculés par RiverApp sur la Dyle à Limelette, depuis la gauche du pont

B.3 Débits limelette - vidéo de la droite du pont

Temps de vidéo analysé [sec]	Débits calculés par RiverApp [m ³ /s]	Débit Aqualim
5	2,13	2,09
10	1,95	2,09
15	1,86	2,09
20	1,79	2,09

TABLE B.3 – Caption

B.4 Débit Petit Bocq - vidéo à 90°

Temps de vidéo analysé [sec]	Débits calculés par RiverApp [m ³ /s]	Débit dilution
5	0,37	0,37
8	0,37	0,37
10	0,37	0,37
15	0,37	0,37
20	0,36	0,37
25	0,36	0,37

TABLE B.4 – Débits calculés par RiverApp sur la vidéo à 90° au dessus du Petit Bocq

B.5 Débit Petit Bocq - vidéo à 45°

Temps de vidéo analysé [sec]	Débits calculés par RiverApp [m ³ /s]	Débit dilution
5	0,4	0,37
8	0,4	0,37
10	0,39	0,37
15	0,39	0,37
20	0,37	0,37
25	0,37	0,37
35	0,36	0,37

TABLE B.5 – Débits calculés par RiverApp sur la vidéo à 45° au dessus du Petit Bocq

B.6 Débit VGC

Temps de vidéo analysé [sec]	Débits calculés par RiverApp [m ³ /s]	Débit réel
5	1,34	1,26
10	1,38	1,26
15	1,4	1,26

TABLE B.6 – Débits calculés par la nouvelle interface de RiverApp

Diagramme du fonctionnement de l'application



FIGURE C.1 – Diagramme du fonctionnement de RiverApp

Annexe D

Protocole de prise de mesures

Fiche d'information générale

Nom du projet :
Lieu :
Cours d'eau :
Date :/...../..... Heure :
Prise de vue :
Condition Météorologiques :

Mesure de la bathymétrie

La section suivante présentera les étapes générales de la mesure de la bathymétrie, ainsi que les équipements requis :

- 2 pieux en bois
- une bombe de couleur/peinture
- un mètre ruban (20m)
- un mètre gradué ou perche transparente à tête de vitesse
- Cuissarde de pêche

1. Sélection de l'emplacement :

Choisissez l'emplacement où vous souhaitez effectuer la mesure de la bathymétrie. Délimitez la section à étudier en posant un pieu en bois de part et d'autre de la rivière et colorez l'extrémité du pieu à l'aide de la bombe de couleur. Il est important de mettre en évidence le pieu à l'image afin de faciliter sa détection à l'oeil nu lors de l'utilisation de l'application.

2. Mise en place :

Attacher le mètre ruban à un pieu en bois et le dérouler jusqu'au pieu de l'autre côté de la rivière en veillant à ce que le mètre soit bien tendu,

de sorte à pouvoir lire la distance directement sur le mètre tendu. Il vous faut ensuite, déterminer l'intervalle à partir duquel vous allez prendre les mesures. Il est conseillé d'ajuster cet intervalle en fonction de la largeur de la rivière car cette étape peut être très lourde. Par exemple, lors des prises de mesures sur la Dyle (largeur de rivière 8m) un intervalle de 20 cm a été utilisé, tandis que sur le petit Bocq (largeur de rivière de 3.6m), un intervalle de 10 cm a été utilisé.

3. **Mesure** : Pour mesurer la bathymétrie, il est possible d'utiliser un mètre gradué ou une perche transparente à tête de vitesse pour mesurer la profondeur.

La perche transparente est plus efficace dans le sens où l'on peut mesurer la profondeur et s'assurer que celle-ci est parfaitement à la verticale à l'aide d'un niveau à bulle tridimensionnelle. De plus, elle permet également de mesurer la vitesse de l'écoulement comme le montre la figure 2.1c.

4. **Encoder la bathymétrie** :

Pour encoder la bathymétrie, il faut créer un fichier *.txt* et y insérer les coordonnées. Les abscisses commencent au point le plus à gauche de la rivière, tandis que les ordonnées commencent au point le plus profond mesuré.

Pose des balises

Pour réaliser une analyse PIV correcte, il est nécessaire de réaliser une orthorectification. Afin de préparer ces étapes, il est nécessaire d'avoir le matériel suivant :

- 4 balises (GCPs)
- Station dGPS Leica1200
- Kit d'orthorectification (4.2)
- Corde

Pour ces étapes, deux approches sont maintenant envisageables. La méthode est disponible avec ou sans kit d'orthorectification.

- **Méthode sans kit d'orthorectification** :

1. **Pose de balise** :

Dans un premier temps, il faut placer les balises. Pour ce faire, il faut les placer, sur les berges, aux 4 coins de la zone que vous souhaitez traiter. Ces balises doivent être visibles depuis la caméra qui sera utilisée.

2. **Mesurer la distance entre les balises** :

Ensuite, il vous faudra mesurer la distance entre les balises comme l'indique la figure D.1. Si vous avez uniquement de la corde, il vous faudra mesurer les distances les séparant à l'aide de celle-ci au risque

d'avoir des mesures imprécises dues à la végétation des berges ou d'autres obstacles.

Cependant, si vous savez vous procurez une station dGPS Leica1200, il est possible de prendre les coordonnées GPS de chaque balise avec une précision de $\pm 2[\text{cm}]$ et à partir de ces coordonnées d'utiliser la formule de distance entre 2 points fonctionnent :

$$L_{A-B} = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2 + (z_B - z_A)^2} \quad (\text{D.1})$$

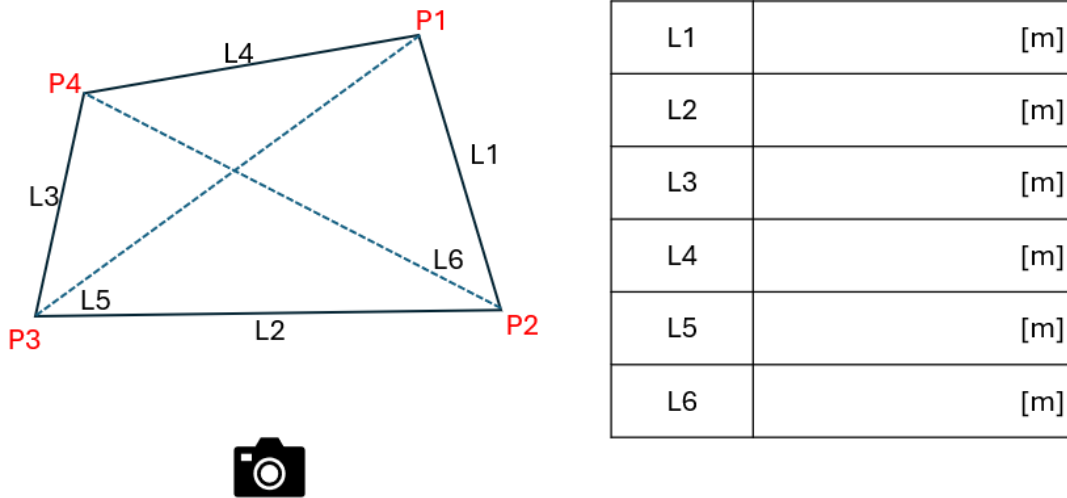


FIGURE D.1 – Teampate mesure des balises

3. Mesurer la Lens position :

Il est désormais nécessaire à l'application de connaître la position de la caméra. De la même manière que dans le point précédent, il est possible de mesurer ces distances avec une corde ou avec une station dGPS.

— Méthode avec kit d'orthorectification :

Dans le cas où un kit d'orthorectification est utilisé, on épargne un bon nombre de manipulation laborieuse.

Dans ce cas-ci, les balises sont directement fixées sur le kit et les distances les séparant sont connues.

Il suffit donc seulement de mettre le kit à l'eau et de mesurer la *lens position*, pour ce dernier, il est possible de procéder de la même manière que sans kit d'orthorectification.

Prise de vidéo

Dernièrement, il nous faudra prendre les vidéos de l'écoulement et pour se faire le matériel suivant est nécessaire :

- une caméra (GoPro ou smart phone)
- Un trépied
- un drone avec caméra intégrée
- des traceurs (copeaux de bois 10/40[mm])
- un filet 2x10[m], maillage 1.8[mm]

1. Position de la caméra :

- Dans le cas où l'on utiliserait une caméra, il faut placer et orienter celle-ci de manière à avoir sur l'image les 4 balises ainsi que les pieux en bois de la bathymétrie. On peut stabiliser les caméras à l'aide d'un trépied si le terrain le permet.
- Si l'on utilise un drone, il faut s'assurer que les balises ainsi que les pieux en bois de la bathymétrie soit visible depuis celui-ci et que le drone reste au maximum stationnaire le temps de la vidéo.

2. Lancement de la vidéo :

Avant de lancer la vidéo, 2 derniers préparatifs sont nécessaires, premièrement l'ensemencement. Pour ensemer la rivière, il faudra se munir des traceurs et se placer suffisamment en amont de la zone d'intérêt pour assurer un ensemencement recouvrant une zone assez grande et de les jeter dans l'eau en nombre suffisant.

Deuxièmement, il faudra placer le filet en aval de la rivière pour récupérer les traceurs lancés en amont. Ce filet permet de faire des économies et de ne pas polluer la rivière.

Maintenant, que tous les préparatifs sont faits, la vidéo peut être lancée. Il est important de noter que la vidéo prise doit être sous le format *.mp4*

Bibliographie

- [1] RENARD S. Development of an application for river discharge measurement [Master's thesis]. Louvain-la-Neuve, BE; 2020. Available at <https://dial.uclouvain.be/memoire/ucl/object/thesis:26501>.
- [2] Debray G, Hay D. Mesure de débit en rivière : analyse et optimisation du projet riverApp [Master's thesis]. Louvain-la-Neuve, BE; 2023. Available at <https://dial.uclouvain.be/memoire/ucl/fr/object/thesis%3A40737>.
- [3] Arnaud M. Amélioration et correction du code source de RiverApp [Bachelor's thesis]. Louvain-la-Neuve, BE; 2023.
- [4] Qu'est-ce que CI? - Explication de l'intégration continue - AWS;. (Accessed on 05/23/2024). [https://aws.amazon.com/fr/devops/continuous-integration/#:~:text=L'int%C3%A9gration%20continue%20d%C3%A9signe%20g%C3%A9n%C3%A9ralement,exemple%20apprendre%20%C3%A0%20int%C3%A9grer%20fr%C3%A9quemment\)](https://aws.amazon.com/fr/devops/continuous-integration/#:~:text=L'int%C3%A9gration%20continue%20d%C3%A9signe%20g%C3%A9n%C3%A9ralement,exemple%20apprendre%20%C3%A0%20int%C3%A9grer%20fr%C3%A9quemment)).
- [5] de Grand Ry E, Everaert C. Mesures de débit en rivière à partir de techniques d'imagerie : développement d'une application [Master's thesis]. Louvain-la-Neuve, BE; 2021. Available at <https://dial.uclouvain.be/memoire/ucl/object/thesis:30607>.
- [6] Daubagnan L, Hauet A, Morlot T. Velocity profile and depth-averaged to surface velocity in natural streams : A review over a large sample of rivers. E3S Web Conf. 2018;40 :6-15.
- [7] Mignot A. RiverApp : Application de mesure non-intrusive du débit des rivières basée sur l'analyse PIV d'une vidéo de l'écoulement [Master's thesis]. Louvain-la-Neuve, BE; 2024.
- [8] Jaugeage du débit d'un cours d'eau par dilution de chlorure de sodium – Sciences de la Vie et de la Terre;. (Accessed on 05/22/2024). <https://www.pedagogie.ac-nice.fr/svt/?p=1744>.
- [9] OTT C2 - OTT France;. (Accessed on 05/23/2024). <https://www.ott.com/fr-fr/produits/le-debit-deau-70/ott-c2-463/>.
- [10] OTT MF Pro Velocity and Depth Sensor, Cable 6 m | Hach - Overview;. (Accessed on 05/23/2024). <https://my.hach.com/>

- ott-mf-pro-velocity-and-depth-sensor-cable-6-m/product?id=59429668076#.
- [11] revue-set.fr/article/download/7293/26106?inline=1;. (Accessed on 05/23/2024). <https://revue-set.fr/article/download/7293/26106?inline=1>.
 - [12] Courantomètre - Unité Littoral;. (Accessed on 05/23/2024). <https://littoral.ifremer.fr/Laboratoires-Environnement-Ressources/LER-Normandie/Infrastructures-et-Moyens/Engins-de-prelevements-et-de-mesures/Courantometre>.
 - [13] Fudaa-LSPIV – River Hydraulics;. (Accessed on 05/21/2024). <https://riverhydraulics.inrae.fr/outils/logiciels/fudaa-lspiv/>.
 - [14] PIVlab - particle image velocimetry (PIV) tool with GUI - File Exchange - MATLAB Central;. (Accessed on 05/21/2024). <https://nl.mathworks.com/matlabcentral/fileexchange/27659-pivlab-particle-image-velocimetry-piv-tool-with-gui>.
 - [15] Rectification of Image Velocity Results (RIVeR) : A simple and user-friendly toolbox for large scale water surface Particle Image Velocimetry (PIV) and Particle Tracking Velocimetry (PTV) - ScienceDirect;. (Accessed on 05/21/2024). <https://www.sciencedirect.com/science/article/pii/S0098300417307045?via%3Dihub>.
 - [16] DISCHARGE;. (Accessed on 05/21/2024). <https://discharge.ch/>.
 - [17] Capteur CAM-FLOW : mesure de vitesses, jaugeages et débits par caméra;. (Accessed on 05/21/2024). <https://www.tenevia.com/fr/sensor/cam-flow-mesure-de-debit-par-camera-lspiv-discharge/>.
 - [18] Célestin DUFROMONT PL. Mesures de débit en rivière à partir de techniques d'imagerie : Développement d'une application [Master's thesis]. Louvain-la-Neuve, BE; 2022. Available at <https://dial.uclouvain.be/memoire/ucl/object/thesis:37914>.
 - [19] 15. Bonnes pratiques en programmation Python - Cours de Python;. (Accessed on 05/06/2024). https://python.sdv.univ-paris-diderot.fr/15_bonnes_pratiques/.
 - [20] Variable shadowing - Wikipedia;. (Accessed on 05/06/2024). https://en.wikipedia.org/wiki/Variable_shadowing.
 - [21] Le champ de saisie - PC SOFT - Online documentation;. (Accessed on 05/28/2024). https://help.windev.com/en-US/?1013145&name=Edit_control&lf=fr#:~:text=Le%20texte%20d%27indication%20permet,de%20saisie%20de%20type%20texte.

- [22] Interface de programmation — Wikipédia;. (Accessed on 05/20/2024). https://fr.wikipedia.org/wiki/Interface_de_programmation.
- [23] Docker (logiciel) — Wikipédia;. (Accessed on 05/20/2024). [https://fr.wikipedia.org/wiki/Docker_\(logiciel\)](https://fr.wikipedia.org/wiki/Docker_(logiciel)).
- [24] Fuite de mémoire — Wikipédia;. (Accessed on 05/13/2024). https://fr.wikipedia.org/wiki/Fuite_de_m%C3%A9moire#:~:text=En%20informatique%2C%20une%20fuite%20de,la%20m%C3%A9moire%20de%20l%27ordinateur.
- [25] `file.read()` memory leak when reading in "r" mode but not "rb" mode, and only inside of Flask.. : r/flask;. (Accessed on 05/13/2024). https://www.reddit.com/r/flask/comments/wa2fyr/file_read_memory_leak_when_reading_in_r_mode_but/.
- [26] python - Flask Application Memory Leak - Stack Overflow;. (Accessed on 05/13/2024). <https://stackoverflow.com/questions/76846960/flask-application-memory-leak>.
- [27] cron — Wikipédia;. (Accessed on 05/13/2024). <https://fr.wikipedia.org/wiki/Cron>.
- [28] GitLab Runner | GitLab;. (Accessed on 05/15/2024). <https://docs.gitlab.com/runner/>.
- [29] L'hydrométrie en Wallonie - Débit;. (Accessed on 05/21/2024). <https://hydrometrie.wallonie.be/home/observations/debit.html?mode=announcement>.
- [30] Aqualim + InfoCrue = le nouveau Portail de l'Hydrométrie en Wallonie - Inondations en Wallonie | Inondations en Wallonie;. (Accessed on 05/21/2024). <https://inondations.wallonie.be/home/actualites/actualites/aqualim--infocrue--le-nouveau-portail-de-lhydrometrie-en-wallonie.html>.
- [31] Welcome — Sphinx documentation;. (Accessed on 05/30/2024). <https://www.sphinx-doc.org/en/master/>.
- [32] API REST : définition;. (Accessed on 05/20/2024). <https://www.redhat.com/fr/topics/api/what-is-a-rest-api>.
- [33] DocString : Qu'est-ce que c'est? À quoi ça sert?;. (Accessed on 05/17/2024). <https://datascientest.com/docstring-tout-savoir#:~:text=Les%20docstrings%20sont%20des%20cha%C3%A9nes,d%C3%A9tection%20d'erreur%20entre%20autres>.
- [34] Google Gemini — Wikipédia;. (Accessed on 05/30/2024). https://fr.wikipedia.org/wiki/Google_Gemini.

- [35] python - Why does right-clicking create an orange dot in the center of the circle? - Stack Overflow;. (Accessed on 05/28/2024). <https://stackoverflow.com/questions/12692851/why-does-right-clicking-create-an-orange-dot-in-the-center-of-the-circle>.
- [36] Testing Flask Applications with Pytest | TestDriven.io;. (Accessed on 05/16/2024). <https://testdriven.io/blog/flask-pytest/>.
- [37] Testing Flask Applications — Flask Documentation (3.0.x);. (Accessed on 05/16/2024). <https://flask.palletsprojects.com/en/3.0.x/testing/>.
- [38] Boiten W. Hydrometry. 3rd ed. IHE Delft Lecture Note Series. CRC Press; 2020.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN

École polytechnique de Louvain

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | www.uclouvain.be/epl