UCLouvain

epl

École polytechnique de Louvain

# WatCam: mesure non-intrusive du débit des rivières au moyen d'une vidéo de l'écoulement

Auteurs: **Xavier Sanchez-Rivas, Kilian Ury**
Promoteur: **Sandra Soares-Frazão**
Lecteurs: **Kim Mens, Pierre-Yves Gousenbourger**
Année académique 2024–2025
Master [60] in Computer Science

## Remerciements

Avant d'entrer dans le vif du sujet, nous souhaitons exprimer notre profonde gratitude à toutes les personnes qui ont contribué, de près ou de loin, à la réalisation de ce mémoire.

Nous tenons tout d'abord à remercier chaleureusement notre promotrice, la Professeure Sandra Soares-Frazao, pour son accompagnement bienveillant, la clarté de ses conseils, ainsi que sa disponibilité tout au long de ce travail. Son expertise et son exigence scientifique ont constitué un cadre stimulant qui nous a permis de progresser tant sur le plan méthodologique que technique. Nous remercions également Monsieur Pierre-Yves Gousenbourger pour sa supervision, ses retours pertinents et son regard critique, qui ont joué un rôle essentiel dans l'amélioration de la qualité de l'application et de notre raisonnement global.

Nous souhaitons également adresser nos remerciements à l'ensemble de l'équipe GCE (Génie Civil et Environnemental) pour leur soutien, leurs retours constructifs, et les nombreuses discussions enrichissantes qui ont jalonné cette année. Un merci tout particulier à Masoumeh, Zoé, Martin, Erwan, Florian, Arthur et Cyril, pour avoir accepté de mettre à disposition leurs expérimentations, leurs résultats ou leur temps, ce qui nous a permis de confronter les résultats de WatCam à des cas réels et d'en tester concrètement les fonctionnalités. Un remerciement spécial à Rotchild pour son aide logistique et technique liée à l'utilisation du drone, élément central dans l'acquisition de nos vidéos sur le terrain.

Enfin, nous adressons toute notre reconnaissance à Samuel Jean : grâce à son travail d'automatisation des tests, nous avons pu valider nos avancées de manière rapide et efficace.

Nous remercions également Monsieur Kim Mens et Monsieur Pierre-Yves Gousenbourger d'avoir accepté de faire partie de notre jury de mémoire. Leur implication et leur regard extérieur seront, sans nul doute, précieux pour enrichir notre réflexion et porter une analyse critique sur notre travail.

Enfin, nous souhaitons remercier nos familles, nos amis et nos compagnes pour leur soutien indéfectible tout au long de ce projet. Leur patience, leurs encouragements et leur compréhension ont été des piliers sur lesquels nous avons pu nous appuyer pendant les moments de doute ou lors des périodes de forte charge de travail.

Ce mémoire est le fruit d'un travail d'équipe, mais aussi d'un environnement humain stimulant, dont nous mesurons aujourd'hui toute la richesse.

## Résumé

Depuis toujours, la mesure du débit des cours d'eau constitue une donnée essentielle dans de nombreux domaines, tels que la gestion des ressources hydriques, l'aménagement du territoire ou encore la surveillance environnementale. Ce dernier aspect revêt une importance croissante face aux bouleversements climatiques actuels.

Ces dernières années, le monde a été marqué par des épisodes extrêmes, alternant entre longues périodes de sécheresse et précipitations intenses provoquant des crues soudaines des rivières et des fleuves. Dans ce contexte de changement climatique accéléré, où la fréquence et l'intensité des événements hydrologiques extrêmes tendent à s'amplifier, la connaissance précise et actualisée des débits fluviaux devient un enjeu crucial pour une gestion durable des cours d'eau et des infrastructures associées.

Les méthodes de mesure traditionnelles, utilisées depuis plus d'un siècle, ont connu des améliorations notables en termes de fiabilité et de précision. Toutefois, leur caractère intrusif limite leur applicabilité dans certaines conditions. En particulier, lors des crues, ces techniques deviennent difficilement exploitables, que ce soit pour des raisons de sécurité ou en raison de la dégradation de leur précision. Dans de telles situations, les opérations de terrain peuvent même représenter un danger réel pour les opérateurs. C'est pourquoi nous souhaitons apporter notre pierre à l'édifice en améliorant une solution accessible à tous afin de contribuer, à notre échelle, au monde de demain.

Il convient de souligner que la mesure du débit en période de crue est non seulement utile, mais cruciale. Elle permet, d'une part, d'alimenter les systèmes de prévision et d'alerte précoce contre les inondations, réduisant ainsi les risques pour les populations. C'est donc dans cette optique qu'un outil tel que WatCam, non-intrusif, portable et accessible, prend tout son sens.

Ce mémoire s'inscrit dans la continuité des travaux précédemment menés autour de l'application initialement nommée RiverApp, renommée WatCam afin d'éviter toute confusion avec une autre application existante portant le même nom. Les études antérieures ont permis de poser les fondations d'un outil capable d'estimer le débit des rivières à partir de simples vidéos, en s'appuyant sur la vélocimétrie d'image (*PIV*). Notre mission, dans le cadre de ce travail, a été de faire évoluer l'application sur le plan fonctionnel et technique, en mettant l'accent sur l'amélioration de l'expérience utilisateur. Concrètement, cela s'est traduit par l'implémentation de nouveaux filtres d'analyse vidéo visant à renforcer la précision des calculs dans des conditions variables, ainsi que par une refonte partielle du code (*refactoring*) afin d'en améliorer la lisibilité, la maintenabilité et la modularité. Un objectif complémentaire de notre travail était également de faciliter la prise en main de l'outil par des utilisateurs non experts via une automatisation des processus, une séparation des différents services tout en garantissant la fiabilité des estimations produites et l'ajout de nouveaux boutons facilitant la prise en main.

Ce travail marque ainsi une nouvelle étape dans le développement de WatCam, en la transformant en une application plus cohérente, évolutive et prête à être déployée dans diverses conditions. En bonus, nous nous sommes permis de proposer un nouveau logo (Figure 1) afin de créer une vraie identité à l'application.



FIGURE 1 – Le nouveau Logo de WatCam

## Avertissements

L'intelligence artificielle a été utilisée dans ce mémoire. GitHub Copilot a été employé pour explorer différentes possibilités d'implémentation des filtres, tandis que ScholarGPT a été utilisé pour corriger certaines phrases afin d'en améliorer la clarté et le vocabulaire. Nous avons également eu recours à DALL · E pour proposer une nouvelle icône de WatCam.

# Table des matières

**1 Introduction** **8**

**2 État de l'art** **10**
2.1 Principes de l'hydrométrie et de l'estimation du débit . . . . . 10
2.1.1 Débit : définition et fondement physique . . . . . 11
2.1.2 La mesure de la bathymétrie . . . . . 13
2.1.3 Mesure de la vitesse . . . . . 14
2.1.4 Synthèse . . . . . 18
2.2 Développement des approches basées sur la vision par ordinateur . . . . . 18
2.2.1 Facteurs influençant la qualité des mesures hydrauliques . . . . . 21
2.2.2 Applications existantes et positionnement de *WatCam* . . . . . 23
2.2.3 Conclusion . . . . . 26

**3 Présentation de WatCam** **27**
3.1 Genèse du projet et héritage de WatCam . . . . . 27
3.2 Architecture Générale de l'Application WatCam . . . . . 29
3.2.1 Interface Utilisateur . . . . . 29
3.2.2 Traitement des Données . . . . . 30
3.2.3 Utilitaires Partagés et Configuration . . . . . 30
3.2.4 Organisation et Flux de Projet . . . . . 30
3.3 Analyse PIV dans WatCam . . . . . 33
3.3.1 Rappel du contexte méthodologique . . . . . 33
3.3.2 Implémentation dans WatCam . . . . . 33
3.3.3 Calibration et conversion métrique . . . . . 34
3.4 Système de stockage des données et persistance des projets . . . . . 34
3.5 Gestion des performances et optimisation des calculs . . . . . 35
3.6 User Stories . . . . . 36
3.6.1 Introduction . . . . . 36
3.6.2 Interface générale et navigation . . . . . 36
3.6.3 Création d'un nouveau projet . . . . . 37
3.6.4 Bathymétrie . . . . . 37
3.6.5 Beacons . . . . . 37
3.6.6 Filtres . . . . . 38

3.6.7 Post-Process . . . 38
3.6.8 Résultats PIV . . . 38
3.6.9 Synthèse . . . 38

**4 Développement et apports techniques 40**
4.1 Rappel des objectifs . . . 40
4.2 Implémentation des filtres . . . 41
4.2.1 Choix d'une image optimale pour l'analyse PIV . . . 45
4.3 Analyse de sensibilité . . . 46
4.3.1 Conclusion . . . 47
4.4 Refonte logicielle et architecture MVC . . . 48
4.4.1 Pourquoi une refonte ? Les limites du code initial . . . 48
4.4.2 Principe de l'architecture MVC . . . 48
4.4.3 Étude de cas : refonte du module `projects.py` . . . 48
4.4.4 Pourquoi cette architecture n'a pas été appliquée partout . . . 51
4.4.5 Pourquoi MVC plutôt qu'une autre architecture ? . . . 51
4.4.6 Un retour d'expérience à la fois technique et pédagogique . . . 52
4.5 Améliorations interface utilisateur . . . 53
4.5.1 Problèmes initiaux . . . 53
4.5.2 Ecran des projets . . . 54
4.5.3 Ecran des filtres . . . 55
4.5.4 Écran des résultats . . . 57
4.5.5 Vérification des fichiers bathymétriques . . . 58
4.6 Application du principe DRY dans le code . . . 59
4.7 Création d'un exécutable (.exe) et limites rencontrées avec Kivy . . . 60
4.8 Conclusion . . . 62

**5 Expériences et validations 64**
5.1 Campagnes en laboratoire . . . 64
5.1.1 Expérience du *Dike Breaching* . . . 64
5.1.2 Expérience de rupture de barrage . . . 66
5.2 Expérience en extérieur . . . 69
5.3 Bilan . . . 77

**6 Perspectives d'amélioration et développements futurs 78**
6.1 Filtres pour l'analyse PIV . . . 78
6.1.1 Vue d'ensemble des filtres . . . 78
6.1.2 Optimisation des combinaisons de filtres . . . 80
6.1.3 Défis et considérations . . . 81
6.2 Autres travaux futurs . . . 81

**7 Conclusion 83**

**8 Annexe** **85**
8.1 Techniques de vélocimétrie optique . . . . . 86
8.2 Solutions existantes . . . . . 87
8.3 Effet des filtres . . . . . 87
8.4 Analyse de sensibilité des différents paramètres . . . . . 88
8.4.1 Durée de la vidéo . . . . . 88
8.4.2 FPS . . . . . 89
8.4.3 Filtres . . . . . 90
8.5 Extraits de code . . . . . 94
8.5.1 Gestion simplifiée des états dans le lecteur vidéo . . . . . 94
8.5.2 Factorisation de la mise en forme du temps . . . . . 94
8.5.3 Centralisation des messages d'erreur . . . . . 94
8.6 Annexes techniques . . . . . 95
8.6.1 Code pour la génération de l'exécutable Windows . . . . . 95

# Chapitre 1

# Introduction

Ce mémoire s'inscrit dans la continuité des travaux précédemment menés autour de l'application RiverApp, aujourd'hui renommée WatCam, afin d'éviter toute confusion avec une autre application du même nom, Riverapp, une application mobile conçue pour le suivi en temps réel des niveaux d'eau et des débits des rivières.

Les études antérieures ont permis de poser les bases d'un outil capable d'estimer le débit des rivières à partir de simples séquences vidéo, en s'appuyant sur les principes de vélocimétrie d'image ($PIV$).

Dans un premier temps, notre travail a consisté à nous familiariser avec l'architecture du code existant. Celui-ci, bien que perfectible, reposait sur des fondations solides, offrant un socle pertinent pour envisager des améliorations concrètes. Avec l'appui de notre promotrice, nous avons identifié et hiérarchisé les axes d'amélioration à prioriser afin de renforcer l'utilité et l'efficacité du projet.

Notre mission, dans le cadre de ce mémoire, a été de faire évoluer l'application tant sur le plan technique que fonctionnel, avec une attention particulière portée à l'amélioration de l'expérience utilisateur. Concrètement, cela s'est traduit par l'implémentation de filtres de traitement vidéo supplémentaires, conçus pour améliorer la robustesse des analyses dans des conditions d'éclairage ou de contraste variables. En parallèle, une opération de *refactoring* a été menée afin d'améliorer la lisibilité, la maintenabilité et la modularité du code. En complément, nous avons également proposé une nouvelle identité visuelle à travers la conception d'un logo.

En parallèle du développement logiciel, nous avons réalisé plusieurs séances de capture vidéo sur le terrain et en laboratoire. L'objectif était d'enrichir la base de données de WatCam en couvrant une diversité de configurations. Ces vidéos ont servi de cas d'étude pour éprouver le fonctionnement de l'application dans des conditions variées, évaluer sa fiabilité et identifier des pistes d'amélioration futures.

Nous avons également pris soin de formaliser de manière claire et structurée les *user stories* de l'application WatCam. Ces récits utilisateurs définissent les besoins fonctionnels du projet du point de vue des usagers. Ils constituent un cadre de référence essentiel pour guider les développements futurs et assurer la cohérence avec la vision initiale de l'outil. Cette documentation contribuera à maintenir une trajectoire claire pour les évolutions à venir, tout en facilitant la prise en main du projet par de futurs contributeurs.

Enfin, nous avons proposé plusieurs pistes d'amélioration ainsi que les problèmes encore à résoudre pour le développement futur de WatCam, à la manière de A.Bombeart [1].

# Chapitre 2

# État de l'art

## 2.1 Principes de l'hydrométrie et de l'estimation du débit

L'hydrométrie regroupe l'ensemble des méthodes et outils destinés à mesurer les caractéristiques physiques des écoulements d'eau, qu'il s'agisse de milieux naturels tels que les rivières et les fleuves, ou d'aménagements artificiels comme les canaux ou les réseaux hydrauliques.

Parmi les paramètres mesurés, le débit — exprimé en mètres cubes par seconde ($m^3/s$) — occupe une place centrale. Il constitue un indicateur fondamental pour la gestion durable des ressources en eau, la prévention des crues et des sécheresses, ainsi que pour le dimensionnement et la sécurité des infrastructures hydrauliques.

La mesure du débit permet d'évaluer, à un instant donné, les volumes d'eau transitant dans un cours d'eau. Cette information est essentielle pour planifier l'approvisionnement en eau potable, encadrer les prélèvements agricoles et industriels, et maintenir un débit écologique minimal garantissant la préservation des écosystèmes aquatiques.

D'un point de vue environnemental, le suivi régulier du débit contribue à l'élaboration des bilans hydrologiques, à l'analyse des régimes fluviaux et à la surveillance des effets du changement climatique. Il est également utilisé pour la calibration de modèles de qualité de l'eau, dans les études de transport sédimentaire, ainsi que pour la prévision de la disponibilité des habitats de la faune aquatique. Ainsi, une connaissance fine et actualisée du débit s'impose comme un prérequis incontournable pour une gestion intégrée et responsable des systèmes hydriques.

Généralement, les mesures de débit sont réalisées à l'aide de techniques dites intrusives, reposant sur un contact direct avec l'eau via des instruments immergés. Si ces approches restent fiables, elles ne peuvent toutefois pas toujours être mises en œuvre, notamment en contexte de crue ou dans les zones difficiles d'accès.

L'évolution récente des technologies numériques a permis l'émergence de méthodes non intrusives, qui exploitent l'imagerie (caméras fixes, drones), les capteurs acoustiques ou encore la télédétection satellitaire pour estimer les paramètres hydrauliques à distance, de manière plus sûre et parfois en temps réel.

Ces avancées rendent possible l'observation fine de phénomènes hydrologiques tels que la répartition spatiale des vitesses d'écoulement, la dynamique des crues ou encore la morphologie du lit fluvial (bathymétrie), sans perturber le milieu. Elles ouvrent ainsi la voie à des solutions plus accessibles, automatisables et adaptées aux enjeux climatiques actuels.

### 2.1.1 Débit : définition et fondement physique

La mesure du débit des cours d'eau constitue un enjeu fondamental dans de nombreux domaines liés à la gestion de l'eau et de l'environnement. Elle permet d'évaluer la quantité d'eau transitant dans les rivières, les fleuves et les canaux, et sert ainsi de base à la gestion des ressources hydriques, à la planification des usages agricoles, industriels ou domestiques, ainsi qu'au suivi écologique des milieux aquatiques.

Sur le plan de la sécurité, connaître le débit en temps réel est particulièrement utile pour anticiper les crues, gérer les périodes de sécheresse et prévenir les inondations.

À une époque où le changement climatique intensifie les épisodes extrêmes (précipitations intenses, étiages prolongés), disposer d'outils fiables et accessibles pour mesurer le débit devient un indicateur précieux pour appuyer les décisions des gestionnaires de bassins versants, des autorités publiques et des acteurs locaux.

Formellement, le débit l'exprime par la relation :

$$Q = A \cdot V$$

où :
— $A$ est la surface de la section mouillée (en $\mathrm{m}^2$) ;
— $V$ est la vitesse moyenne de l'écoulement sur toute la section (en m/s).

La Figure 2.1 illustre simplement et efficacement le principe de débit.

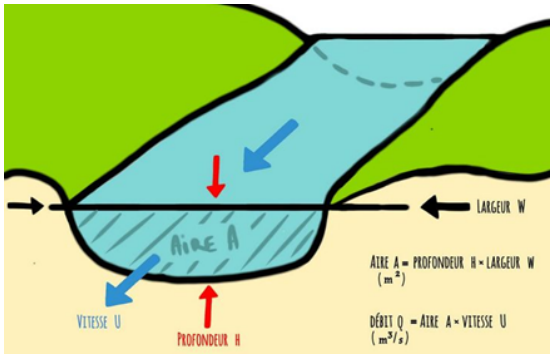



FIGURE 2.1 – Illustration simplifiée du débit

Pour déterminer le débit d'un cours d'eau, deux informations fondamentales sont donc requises : l'aire de la section transversale du lit, et la vitesse moyenne de l'écoulement.

L'aire de la section est obtenue à partir de la bathymétrie, c'est-à-dire la mesure de la forme et de la profondeur du fond du cours d'eau à différents points. Sans ces données, il est impossible d'estimer avec précision la surface mouillée par l'eau.

Quant à la vitesse moyenne, elle ne peut pas être mesurée directement : on accède généralement à une vitesse de surface, qu'il faut ensuite corriger à l'aide de facteurs empiriques ou de modèles afin d'estimer la vitesse moyenne sur toute la profondeur, comme illustré à la Figure 2.2.

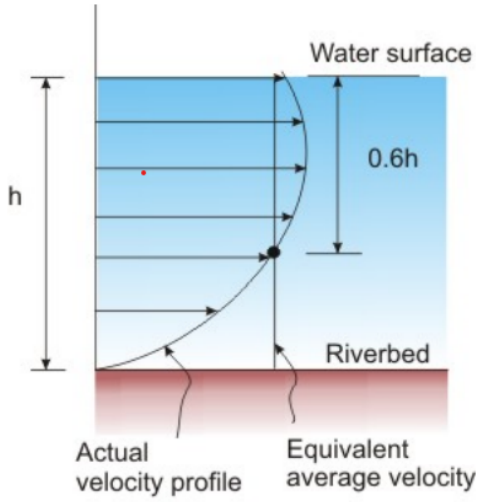



FIGURE 2.2 – Illustration du profil de vitesse en fonction de la profondeur de l'eau. La vitesse de surface est multipliée par un facteur de correction appelé *coefficient de surface* pour obtenir la vitesse moyenne équivalente. [2]

### 2.1.2 La mesure de la bathymétrie

La bathymétrie est la mesure de la topographie du lit fluvial. La connaissance de la géométrie de la rivière, en particulier de la forme du lit, est indispensable pour estimer correctement la section mouillée $A$, surtout dans les approches non intrusives où aucune sonde ne mesure directement la profondeur. Plusieurs techniques permettent aujourd'hui d'obtenir des profils bathymétriques, parmi lesquelles :

— **Le sonar multi-faisceaux** (*multibeam echosounder*), qui envoie des ondes acoustiques et mesure le temps de retour pour calculer la profondeur.
— **Le LIDAR bathymétrique**, qui utilise un laser vert embarqué sur un drone ou un avion pour pénétrer la surface de l'eau et cartographier le fond, représente une technologie prometteuse. Cependant, cette technique n'est pas encore pleinement opérationnelle : la pénétration du laser dans l'eau, surtout en milieux turbides, constitue une limite technique majeure. Elle relève donc davantage du domaine prospectif ou expérimental à ce stade, et ne peut être considérée au même niveau de maturité que d'autres méthodes d'acquisition bathymétrique actuellement fonctionnelles.
— **La photogrammétrie aérienne**, qui permet de reconstruire en 3D les zones peu profondes ou émergées à partir d'images en haute résolution.

Ces différentes techniques sont illustrées à la Figure 2.3.

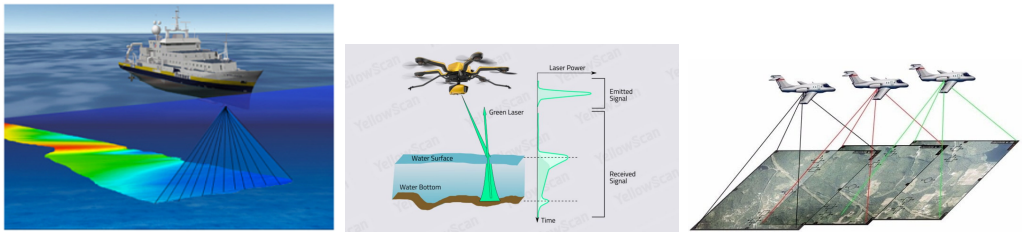



FIGURE 2.3 – Illustrations : Sonar multi-faisceaux, Lidar bathymétrique et photogrammétrie aérienne

Dans le cadre de ce mémoire, et plus largement dans l'usage actuel de l'application *WatCam*, nous avons dû faire l'hypothèse que la bathymétrie du site était connue à l'avance. En effet, nos travaux se concentrent principalement sur le développement et l'amélioration de l'outil en tant que tel, et non sur l'acquisition des données bathymétriques.

En conséquence, tout utilisateur souhaitant estimer le débit d'un cours d'eau à partir d'une vidéo à l'aide de *WatCam* doit impérativement disposer d'une connaissance préalable, fiable et suffisamment détaillée de la bathymétrie de la section analysée.

### 2.1.3 Mesure de la vitesse

Le dernier composant nécessaire à l'estimation du débit est, bien évidemment, la vitesse d'écoulement. La mesure de cette vitesse constitue un pilier fondamental dans le calcul du débit fluvial. Toutefois, il est important de souligner que ce n'est pas la vitesse instantanée ou ponctuelle qui nous intéresse directement, mais bien la vitesse moyenne de l'écoulement sur l'ensemble de la section transversale du cours d'eau.

Nous allons maintenant détailler les différentes techniques permettant d'estimer cette vitesse dans un cours d'eau, en gardant à l'esprit que leur objectif commun est de fournir une estimation fiable de la vitesse moyenne, indispensable au calcul du débit.

#### Via des méthodes intrusives

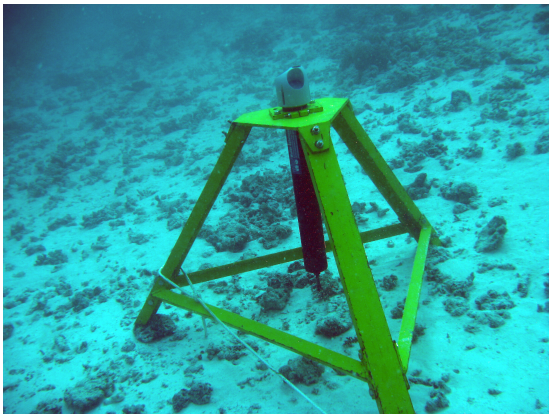
Qu'appelle-t-on une méthode intrusive dans le cadre des cours d'eau ? Une méthode intrusive est une technique de mesure qui nécessite un contact physique direct avec le milieu aquatique. En hydrométrie, cela signifie que les capteurs ou instruments sont immergés dans le cours d'eau afin de mesurer la vitesse, la profondeur ou d'autres paramètres hydrauliques. Ces méthodes, bien que considérées comme précises, doivent être mises en œuvre dans des conditions bien contrôlées pour garantir leur fiabilité.

Les techniques classiques reposent notamment sur l'utilisation de moulinets hydrométriques, qui permettent de mesurer la vitesse de l'eau en différents points d'un profil transversal. Ces dispositifs, malgré leur fiabilité, nécessitent un contact direct avec l'écoulement et sont relativement longs à mettre en œuvre.

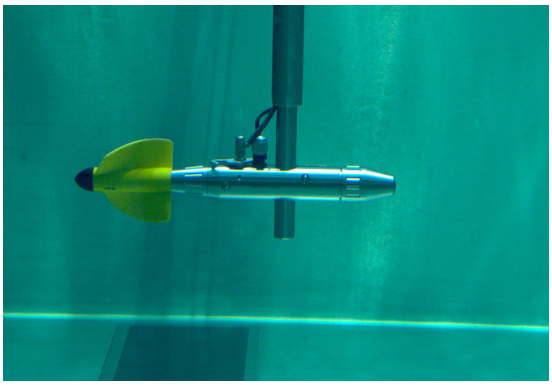
L'introduction des profileurs de courant acoustiques Doppler (ADCP) a marqué une avancée significative [3]. Grâce à l'envoi d'ondes acoustiques dans l'eau, ces appareils permettent d'obtenir des mesures détaillées du débit en temps réel. Toutefois, leur efficacité est conditionnée à une profondeur minimale du cours d'eau, nécessaire pour assurer une bonne propagation du signal.

Voici quelques exemples :

— **Moulinet hydrométrique** (Figure 2.4b) : utilisé pour mesurer la vitesse moyenne à différentes profondeurs, puis estimer le débit total par intégration de surface.
— **ADCP (Acoustic Doppler Current Profilers)** (Figure 2.4a) : dispositifs acoustiques très performants, capables de mesurer dynamiquement les profils de vitesse et la bathymétrie.



(a) Acoustic Doppler Current Profilers



(b) Moulinet hydrométrique

FIGURE 2.4 – Exemples de méthodes non-intrusives

Cependant, ces méthodes restent coûteuses, nécessitent une intervention humaine en milieux parfois dangereux, et sont peu adaptées à un suivi en continu ou à des sites difficiles d'accès. En plus de perturber la faune et la flore, leur mise en œuvre dépend fortement des conditions locales : profondeur, accessibilité, configuration du terrain, etc.

Dans le groupe GCE, une perche à charge dynamique (Figure 2.5a) ainsi que le jaugeage par dilution de sel ont été utilisées comme méthodes de référence pour l'estimation du débit.

**Perche à charge dynamique**

Cette technique, à la fois simple, peu coûteuse et adaptée aux interventions de terrain, repose sur la mesure de la différence de hauteur d'eau entre l'amont et l'aval d'une planche verticale immergée, placée perpendiculairement à l'écoulement. Ce différentiel permet d'estimer la vitesse moyenne du courant, puis le débit en fonction de la section mouillée.

Cette estimation repose sur une relation d'étalonnage semi-empirique [4], qui permet de calculer la vitesse moyenne $V$ [m/s] à partir de la différence de hauteur d'eau $\Delta h$ [m] mesurée entre l'amont et l'aval de la planche. L'équation correspondante est la suivante :

$$V = 0.641\sqrt{2g\Delta h} - 0.019 \tag{2.1}$$

où $g$ représente l'accélération gravitationnelle (9.81 m/s$^2$).

Toutefois, cette méthode présente certaines limitations, notamment liées aux caractéristiques hydrauliques du cours d'eau. Elle est uniquement applicable dans les conditions suivantes :

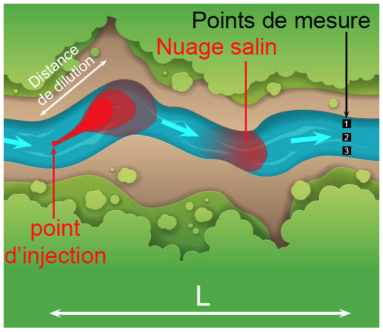
— Profondeur d'eau comprise entre 5 cm et 75 cm ;

— Charge hydraulique $\Delta h$ comprise entre 2 mm et 100 mm ;
— Plage de vitesse mesurable allant de 10 cm/s à 90 cm/s.

**Jaugeage par dilution de sel [5]**
Cette méthode consiste à injecter un volume connu de solution saline agissant comme traceur dans le cours d'eau. À l'aide d'un capteur placé en aval, on mesure ensuite la variation de la conductivité électrique de l'eau pendant une durée suffisante pour suivre le passage du traceur.
La concentration du sel permet d'établir une courbe de dilution, à partir de laquelle le volume écoulé peut être déduit (Figure 2.5b).




(a) Perche à charge dynamique [6]




(b) Fonctionnement du jaugeage par dilution au sel [7]

FIGURE 2.5 – Méthodes utilisées par le groupe GCE

**Via des méthodes non-intrusives**

Une méthode non-intrusive est une technique d'acquisition permettant de mesurer un paramètre hydraulique sans contact direct avec l'eau. Ces méthodes reposent sur des capteurs ou dispositifs placés à distance (en berge, sur un pont, un drone ou un satellite) qui observent les propriétés de l'écoulement à l'aide de signaux acoustiques, optiques, thermiques ou électromagnétiques.

Certaines de ces approches, notamment celles basées sur l'analyse de séquences vidéo, permettent d'évaluer la vitesse des particules flottant à la surface de l'eau. On peut ainsi déterminer la vitesse de surface du cours d'eau sans y pénétrer, puis estimer la vitesse moyenne grâce à l'application d'un coefficient de surface (voir Figure 2.2).

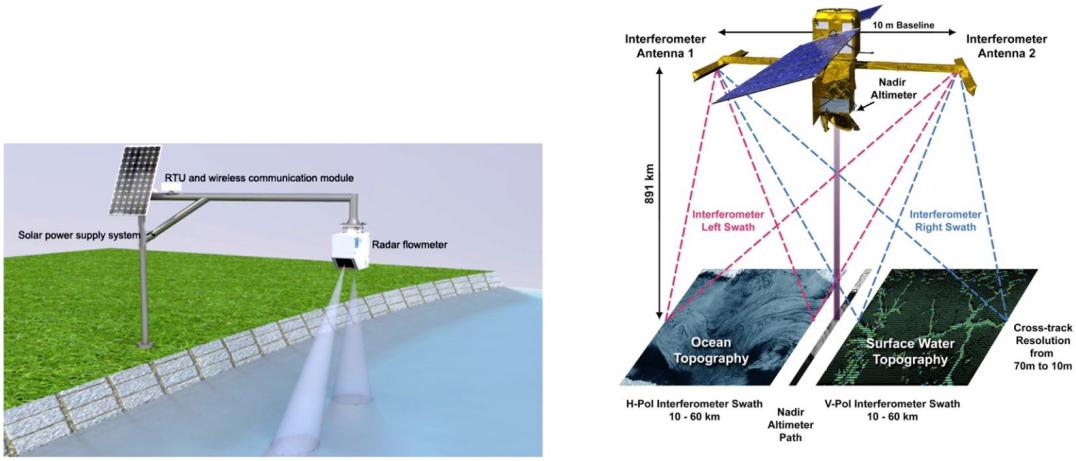
Ces techniques, indépendantes de tout contact avec l'écoulement, sont particulièrement adaptées aux environnements difficiles ou dangereux.

Voici quelques exemples de méthodes non-intrusives :
— **Radar Doppler de surface** (Figure 2.6a) : Ce dispositif exploite l'effet Doppler pour mesurer la vitesse de l'eau en surface, en analysant la fréquence des ondes

électromagnétiques réfléchies par des irrégularités naturelles ou des particules à la surface [3]. Il est souvent installé en continu depuis une berge ou un pont.

— **Télédétection satellite** (par exemple avec *Sentinel-2* ou *SWOT*) (Figure 2.6b) : Cette technique utilise des images satellites multi-spectrales ou altimétriques pour extraire des paramètres hydrologiques comme la hauteur, la surface ou parfois la vitesse de l'eau [8]. La mission *SWOT* (*Surface Water and Ocean Topography*) fournit des mesures précises d'altimétrie des surfaces d'eau, permettant d'estimer le débit via des modèles couplés.

— **Techniques de vélocimétrie optique** : telles que la méthode **PIV** (*Particle Image Velocimetry*) ou **STIV** (*Space-Time Image Velocimetry*), qui permettent d'estimer la vitesse à partir de séquences vidéo de l'écoulement.




(a) Radar Doppler [9]

(b) Swot Mission [10]

FIGURE 2.6 – Représentation du Radar Doppler et de Swot mission

Ce sont précisément ces techniques de vélocimétrie optique qui nous intéressent dans le cadre de l'application *WatCam*. Parmi elles, la méthode PIV permet de mesurer le champ de vitesses à la surface de l'eau à un instant donné [11]. En analysant le déplacement de motifs visibles à la surface — tels que des feuilles, des bulles ou des structures générées par la turbulence — sur une séquence vidéo, il devient possible d'estimer les vitesses de surface.

Ce champ de vitesses est ensuite corrigé géométriquement à l'aide de balises de calage *(Ground Control Points*, GCP), afin d'être exprimé dans un repère physique réel. Combinée à la géométrie de la rivière (via la bathymétrie) et à des hypothèses sur la distribution verticale des vitesses, cette information permet de calculer une estimation du débit instantané.

Dans *WatCam*, cette analyse est effectuée sur un court intervalle temporel supposé stationnaire, ce qui signifie que le débit est considéré constant pendant l'analyse. Il s'agit

donc d'une mesure instantanée, réalisée de manière non-intrusive et rapide.

### 2.1.4 Synthèse

Les différentes techniques de mesure du débit et de la vitesse des rivières offrent des niveaux de précision variables selon les conditions d'utilisation et les caractéristiques du site étudié. Les méthodes intrusives, telles que l'*ADCP*, présentent généralement une grande fiabilité, avec une précision de l'ordre de 1 à 2 %, mais leur efficacité dépend fortement de la profondeur et des conditions d'écoulement [12].

À l'inverse, les approches non-intrusives, bien que plus flexibles et mieux adaptées aux situations dangereuses ou difficilement inaccessibles, sont soumises à des incertitudes plus importantes liées aux conditions environnementales : éclairage, turbidité, ou encore présence de végétation [13][14].

Les techniques acoustiques et optiques sont également sensibles aux conditions météorologiques et hydrauliques. Par exemple, la présence de vagues ou de turbulences marquées peut affecter les mesures obtenues par radar Doppler ou par *LSPIV* (*Large Scale Particle Image Velocimetry*) [15]. De même, la turbidité de l'eau constitue une contrainte majeure pour les méthodes de bathymétrie optique, en limitant la pénétration du signal lumineux, ce qui réduit la précision des relevés [16].

Aujourd'hui, la combinaison d'une bonne connaissance géométrique (bathymétrie), d'une méthode robuste d'estimation de la vitesse, et d'un traitement vidéo adapté permet d'obtenir des mesures de débit fiables, même dans des contextes complexes ou dangereux [17, 18]. Ces avancées technologiques ouvrent la voie à des outils plus accessibles et à faible coût pour le suivi des cours d'eau, en temps réel ou en post-traitement.

## 2.2 Développement des approches basées sur la vision par ordinateur

L'intégration de la **vision par ordinateur** dans les sciences de l'eau a permis d'améliorer les techniques de mesure des écoulements, notamment pour l'estimation de la vitesse de surface, et dont indirectement du débit. Parmi les méthodes émergentes, la **Particle Image Velocimetry (PIV)** occupe une place centrale [19].

Initialement développée pour l'étude des écoulements internes en laboratoire, la *PIV* repose sur la corrélation d'images successives afin d'estimer le déplacement de particules dans un fluide. En suivant les variations locales de texture ou les traceurs visibles dans une séquence d'images, il est possible d'obtenir un champ de vitesses vectoriel à haute résolution.

Des améliorations récentes ont considérablement accéléré le traitement *PIV* grâce à des algorithmes multithread ou à des techniques de corrélation temporelle moyenne [20]. Le principe repose sur l'analyse de vidéos ou d'images prises à intervalles réguliers : en suivant les motifs visibles à la surface, les algorithmes de corrélation génèrent des champs de vitesse. Une *orthorectification* est généralement nécessaire pour corriger la perspective et relier ces vitesses à des dimensions physiques réelles.

Le processus d'analyse comprend en général plusieurs étapes :

1. Stabilisation de la séquence vidéo ;
2. Correction géométrique (*orthorectification*) pour ramener les coordonnées pixel à un système physique (en mètres) ;
3. Application d'un algorithme de corrélation croisée sur des sous-fenêtres d'images pour détecter les déplacements ;
4. Conversion des déplacements en vitesse (en m/s) à partir de la fréquence d'échantillonnage.

Une illustration schématique de cette analyse est présentée à la Figure 2.7.

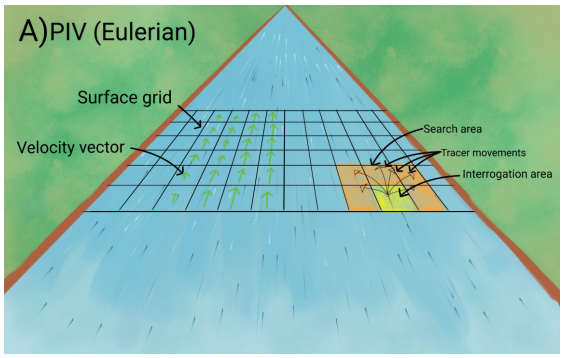



FIGURE 2.7 – Schéma d'une analyse PIV [21]

La **LSPIV** permet des mesures sans contact, en continu, à faible coût, et dans des zones potentiellement dangereuses ou isolées. Son implémentation est grandement facilitée par l'usage de drones et de caméras embarquées.

Elle applique les principes fondamentaux de la PIV à des images de grande dimension, prises à partir de caméras fixes ou embarquées (par drone), capturant l'écoulement réel d'un cours d'eau. Elle s'appuie sur les traceurs naturels présents à la surface de l'eau – bulles d'air, mousse, feuilles – et utilise un traitement d'image permettant d'estimer les vitesses de surface en chaque point d'une grille définie.

La LSPIV présente de nombreux avantages : elle est totalement non-intrusive, rapide à mettre en œuvre, et applicable même dans des environnements inaccessibles ou dangereux.

Elle s'avère particulièrement utile en période de crue, lorsque les méthodes classiques atteignent leurs limites [12], et permet de produire un champ de vitesse bidimensionnel à la surface avec une précision acceptable (erreur typique entre 5 et 20 % selon les conditions).

Cependant, la méthode reste sensible à plusieurs paramètres de calibration, notamment l'angle de la caméra, le positionnement des traceurs, ou encore les conditions de luminosité. Cela nécessite des ajustements spécifiques à chaque application [22, 23]. Même en terrain complexe, la LSPIV permet une restitution fine des structures turbulentes, avec une résolution spatio-temporelle élevée[24].

La méthode **PTV** (*Particle Tracking Velocimetry*) (Figure 2.8) permet, dans certaines conditions, de mieux restituer les vitesses de surface que la LSPIV, notamment en cas de faible visibilité ou de turbulence importante, grâce à son suivi individualisé des traceurs [25].

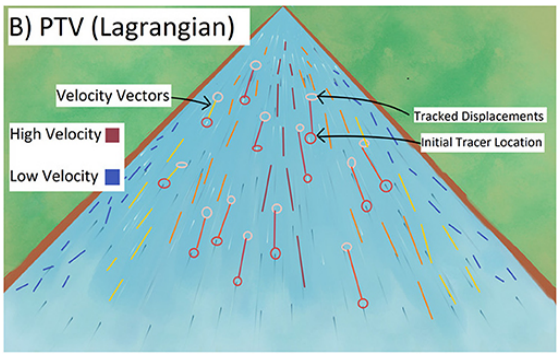



FIGURE 2.8 – Schéma de fonctionnement de la méthode PTV[21]

Toutefois, son efficacité dépend fortement de la qualité des traceurs visibles, de la stabilité de la prise de vue, ainsi que de la géométrie du site (présence de reflets, ondes, ombres).

Pour pallier certains de ces problèmes, une autre méthode a été développée : la ***Space-Time Image Velocimetry* (STIV)** (Figure 2.9). Contrairement à la LSPIV, qui analyse des zones d'images en 2D, la STIV extrait des profils linéaires dans la vidéo et construit des images espace-temps (x-t), dans lesquelles la pente des stries correspond à la vitesse des traceurs.

Cette approche est particulièrement adaptée aux séquences dans lesquelles les traceurs sont peu visibles, mais suivent des trajectoires linéaires — par exemple, de la mousse dérivant dans un chenal. Elle constitue une alternative robuste dans des contexte de faible contraste ou de faible densité de traceurs.

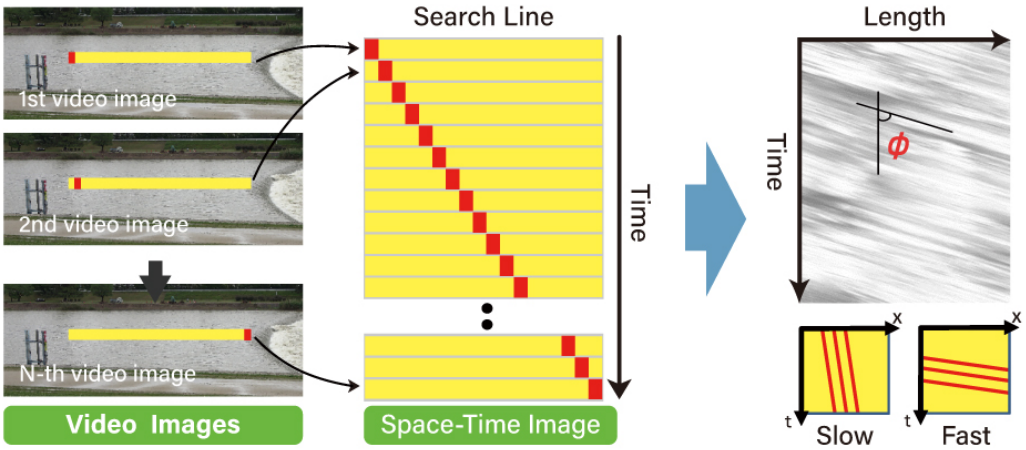



FIGURE 2.9 – Schéma de fonctionnement de la méthode STIV [26]

Le choix de la méthode dépendra de plusieurs facteurs : la fréquence d'acquisition de la vidéo, la stabilité du capteur, la présence et la nature des traceurs, les conditions d'éclairage, ainsi que la géométrie du site (voir Tableau récapitulatif 8.1 en annexe).

### 2.2.1 Facteurs influençant la qualité des mesures hydrauliques

Pour assurer une estimation fiable du débit, il est essentiel de maîtriser les nombreux paramètres qui influencent la qualité des mesures hydrauliques, en particulier lorsqu'elles reposent sur des méthodes optiques comme la PIV ou la STIV.

**Visibilité des traceurs** La détection des motifs en surface (écume, bulles, débris flottants) est une condition préalable indispensable au bon fonctionnement des algorithmes de corrélation. La *turbidité de l'eau* constitue à ce titre un facteur limitant : lorsque l'eau est trop chargée en particules, les traceurs naturels deviennent peu visibles, ce qui compromet la précision des mesures. À l'inverse, les techniques acoustiques comme l'ADCP, sont moins sensibles à la turbidité, mais peuvent être perturbées par la présence de bulles d'air ou de végétation en suspension [27].

**Qualité de l'orthorectification géométrique**

Le *positionnement de la caméra* (angle de vue, hauteur, stabilité) influe fortement sur la qualité des images. Une inclinaison importante ou une distance trop grande par rapport à la surface réduit la résolution spatiale, rendant plus difficile la détection et le suivi des traceurs. Pour garantir un référentiel métrique cohérent, une orthorectification rigoureuse s'impose.

### Conditions environnementales

Le *vent* et les conditions météorologiques influencent directement la dynamique des traceurs. Des rafales peuvent fausser les mesures en déplaçant les particules flottantes indépendamment du courant, tandis qu'une pluie fine peut dégrader la qualité visuelle des séquences. Les reflets, les ombres portées ou les scintillements sur l'eau génèrent des artefacts susceptibles de nuire à l'analyse automatique [12].

### Paramètres techniques de capture

Les réglages de la caméra doivent être définis avec soin. Une **fréquence d'acquisition** trop basse empêche de suivre les mouvements fins, tandis qu'un **temps d'exposition** trop long engendre du flou de mouvement, rendant la corrélation imprécise [19, 11]. Il faut donc trouver un compromis optimal la qualité d'image et la résolution temporelle.

**En résumé**, l'acquisition d'une vidéo destinée à une analyse PIV suppose de concilier les éléments suivants :

- la présence et la visibilité suffisante des traceurs naturels en surface ;
- une orthorectification géométrique fiable (bonne pose des balises, orientation correcte de la caméra) ;
- une caméra stable, avec des réglages adaptés (au moins 10 fps, temps d'exposition court) ;
- des conditions environnementales favorables (vent faible, bonne lumière, absence de pluie).

Une acquisition vidéo optimale exige donc une préparation rigoureuse du dispositif, tant sur le plan technique qu'environnemental, pour garantir des résultats exploitables et fiables. Un résumé des recommandation se trouve à la Table 2.1.

TABLE 2.1 – Checklist pour une acquisition vidéo optimale en analyse PIV

| **Paramètre** | **Recommandation** |
|---|---|
| **Présence de traceurs visibles** | Écume, bulles, feuilles : nécessaires à la corrélation. Éviter les plans d'eau lisses ou trop turbides. |
| **Qualité de l'orthorectification** | Utiliser des balises de calibration bien visibles et stables ; choisir un angle de vue limitant la distorsion. |
| **Stabilité de la caméra** | Support rigide (trépied ou drone stationnaire). Éviter les vibrations et mouvements parasites. |
| **Réglages de capture** | Fréquence minimale de 10 fps ; temps d'exposition court pour éviter le flou de mouvement. |
| **Conditions météorologiques** | Lumière homogène ; vent faible ; éviter la pluie, les ombres portées et les reflets excessifs. |

### 2.2.2 Applications existantes et positionnement de *WatCam*

Ces dernières années, les solutions logicielles et matérielles dédiées à la mesure non-intrusive du débit fluvial ont connu un développement rapide. Ces outils s'appuient principalement sur la vision par ordinateur (PIV, LSPIV, STIV) ou la télédétection, en réponse à une demande croissante de sécurité, de portabilité et de précision.

#### Solutions logicielles open source ou académiques

**FUDAA-LSPIV**
Parmi les pionniers de la vélocimétrie à grande échelle, FUDAA-LSPIV a été co-développé par EDF, Irstea et la société DeltaCAD. Il permet de mesurer la vitesse de surface à partir de vidéos grâce à la méthode LSPIV. Malgré une fiabilité éprouvée [27], l'outil souffre de plusieurs limites : interface vieillissante, prise en main difficile, et absence de modularité. Il s'avère peu adapté à des utilisateurs non techniques ou à des projets évolutifs comme WatCam [1].

**PIVLab**
Outil MATLAB largement adopté dans les milieux académiques, PIVLab séduit par son interface intuitive et ses capacités avancées (analyse sous-pixel, interrogation multi-passe). Toutefois, sa dépendance à MATLAB - logiciel propriétaire - limite sa diffusion dans des contextes open source ou à faible coût, comme celui de WatCam.

**OpenPIV**
Bibliothèque Python open source implémentant les fondements de la méthode PIV. Elle

est adaptée aux utilisateurs techniques souhaitant une personnalisation fine. En revanche, elle ne propose ni interface graphique native, ni structuration de projet, ce qui complique son adoption en dehors de contextes experts.

**RIVeR**
Application gratuite développée pour l'analyse de vidéos fluviales, basée sur les techniques PIV et PTV. Elle s'appuie partiellement sur les bibliothèques MATLAB (PIVLab, PTVLab), ce qui constitue un frein dans un cadre open source ou pour des utilisateurs souhaitant une adaptabilité maximale [1].

**PyORC (OpenRiverCam)**
Solution plus récente et modulaire, développée à l'échelle européenne pour proposer une alternative automatisée et open source à l'analyse de débit par imagerie. Reposant sur une architecture Python, elle dispose d'une API, d'une documentation technique claire, et peut s'intégrer à des pipelines de traitement. PyOrc a été retenu comme moteur principal de traitement pour WatCam, garantissant ainsi la compatibilité avec les méthodes de référence actuelles [28].

**Applications mobiles ou solutions propriétaires**

**Discharge (Android)**
Application Android permettant une estimation rapide du débit à partir d'une vidéo. Malgré sa facilité d'utilisation et sa rapidité, elle présente plusieurs restrictions : elle ne prend pas en charge les vidéos préenregistrées, n'est pas open source, et ne peut être adaptée à d'autres contextes de mesure [1].

**Tenevia CamFlow**
Système propriétaire de surveillance en continu basé sur l'analyse vidéo. Bien qu'il fournisse des résultats en temps réel, il nécessite une installation fixe, calibrée et constante dans le temps. Ce manque de flexibilité, combiné à son coût et à son caractère non modifiable, rend l'outil peu compatible avec les objectifs d'un projet comme WatCam [1].

**Autres dispositifs innovants récents**

Plusieurs dispositifs récents témoignent de la diversification des approches pour la mesure non-intrusive du débit :

- **RiverCAM (Royaume-Uni)** : réseau de caméras fixes déployé par l'Environment Agency britannique pour surveiller les débits pendant les crues. Le système s'appuie sur la comparaison temporelle d'images et nécessite une présence continue sur site [29].
- **FlowCapture** : solution portable intégrant LSPIV dans une interface embarquée. Elle permet un traitement sur le terrain, particulièrement adapté aux interventions rapides et mobiles.

— **HydroDrone** : drone multisensoriel conçu pour la capture simultanée de vidéos et de données bathymétriques lors d'inondations. Il vise à améliorer la cartographie des zones inondables et la modélisation dynamique des crues.

**Positionnement spécifique de WatCam**

WatCam se distingue dans le paysage des outils de mesure hydrologique par une approche intégrée, légère et polyvalente. Pensé à la fois pour l'environnement académique les campagnes de terrain, il vise à rendre la mesure du débit accessible, reproductible et scientifiquement rigoureuse.

Ses principales caractéristiques sont :

— **Portabilité** : fonctionnant sur ordinateur portable, tablette ou potentiellement smartphone, WatCam s'adapte aux contraintes de terrain et aux interventions légères, sans nécessiter d'infrastructure fixe.

— **Automatisation** : de nombreuses étapes techniques sont automatisées : calibration par balises (GCP), masquage, orthorectification, segmentation du transect et calcul du débit. L'utilisateur peut ainsi se concentrer sur la validation des résultats.

— **Personnalisation** : l'outil permet à l'utilisateur d'effectuer un ajustement fin grâce à des fichiers de configuration, des filtres vidéo paramétrables et une structuration projet claire, adaptée à différents contextes de mesure.

— **Reproductibilité** : le flux de traitement repose sur des formats ouverts et une architecture modulaire, assurant traçabilité et reproductibilité des résultats, en particulier dans un cadre scientifique.

— **Code ouvert** : la licence open source (MIT) favorise les contributions de la communauté, les audits indépendants et l'évolution rapide des fonctionnalités.

Contrairement aux plateformes de surveillance continue ou aux outils experts nécessitant des compétences avancées en traitement d'image ou en hydraulique, WatCam occupe une niche particulière : celle d'un logiciel autonome, pragmatique et léger, destiné aux chercheurs, formateurs et professionnels de terrain souhaitant un outil fiable, rapide à déployer et simple à maintenir.

Un tableau récapitulatif reprends les différentes caractéristiques de chaque solution logicielle en annexe (Table 8.2).

### 2.2.3 Conclusion

Les progrès récents en vision par ordinateur, en intelligence artificielle embarquée et en traitement d'images permettent aujourd'hui la conception d'outils de mesure du débit plus accessibles, efficaces et robustes. Toutefois, chaque solution reste conditionnée par son environnement d'usage : perturbations lumineuses, turbidité, topographie complexe ou instabilité des supports peuvent encore limiter la précision des résultats [30].

Dans ce contexte, WatCam s'inscrit comme une solution hybride et modulaire, combinant automatisation, personnalisation et facilité d'usage. Il répond aux exigences de l'expérimentation rapide sur le terrain, tout en conservant une rigueur méthodologique adaptée aux enjeux de recherche ou de formation.

# Chapitre 3

# Présentation de WatCam

## 3.1 Genèse du projet et héritage de WatCam

WatCam est né d'une volonté claire : rendre plus accessible la mesure du débit en milieu naturel en s'appuyant sur des technologies modernes, portables et open source. Ce mémoire prolonge une série de travaux réalisés sous la direction de la Prof. Sandra Soares-Frazao, et s'inscrit dans la continuité des mémoires de Debray et Hay [2], A.Bombaert [1] et A.Mignot [5].

Le premier mémoire, réalisé par Debray et Hay, avait pour objectif principal de modifier l'application WatCam pour en améliorer les performances et la précision des mesures de débit, en utilisant l'analyse par vélocimétrie d'image de particules (PIV). Une contribution majeure de ce travail est la comparaison entre WatCam et PyOrc. Cette analyse a mis en évidence les limites de WatCam, notamment sa structure logicielle, jugée moins performante.
En réponse, les auteurs ont restructuré WatCam en s'appuyant sur la structure en pipeline de PyOrc, plus performante pour le traitement des données. Une autre contribution significative a été l'intégration d'un module de détection automatique des balises de référence (GCP), supprimant ainsi l'étape fastidieuse du placement manuel complet par l'utilisateur.
Malgré ces améliorations, le mémoire soulignait que WatCam nécessitait encore un certain nombre de développements pour atteindre un niveau de robustesse suffisant dans des contextes variés. Il recommandait notamment la rédaction d'un guide développeur, essentiel pour structurer la poursuite du projet. Cette mise à jour initiale a amélioré la fonctionnalité de certaines fonctionnalités de WatCam, mais l'application avait encore besoin de développement supplémentaire pour s'assurer qu'elle peut fonctionner de manière fiable dans divers environnements.

Le deuxième mémoire, réalisé par A.Bombaert, s'est attaché à améliorer l'interface utilisateur ainsi que la gestion de la charge de calcul, dans le but de rendre WatCam plus accessible aux non-experts et plus scalable pour des usages variés.

L'une des contributions majeures de ce travail est la refonte de l'interface graphique, pensée pour être plus intuitive et ergonomique. Cette nouvelle interface facilite grandement la navigation dans l'application, rendant son utilisation plus fluide, notamment pour les utilisateurs non spécialisés.
Concernant les performances, Bombaert a proposé de déléguer la charge de calcul liée aux traitements PIV à un serveur distant, permettant un traitement plus rapide et, potentiellement, en temps réel. Cette approche représente une avancée importante vers une application légère mais performante, adaptée à un usage sur le terrain.
Enfin, une attention particulière a été portée à l'ouverture du projet : la mise en place d'une stratégie de test automatisée et la rédaction d'une documentation technique claire ont posé les bases d'un développement collaboratif pérenne. Ces apports ont renforcé la robustesse et l'accessibilité de WatCam, tout en affirmant son positionnement open source.

Le troisième mémoire, réalisé par A.Mignot, avait pour objectif d'améliorer la précision des mesures de WatCam et sa robustesse face aux variations des conditions environnementales et des paramètres vidéo. Une réalisation clé de ce travail a été le développement d'un kit d'orthorectification, destiné à faciliter un positionnement précis lors des prises de vues sur le terrain. Ce kit comprend divers matériaux et équipements pratiques, permettant d'améliorer l'expérience utilisateur en conditions réelles.
Mignot a également mené une étude approfondie de la sensibilité de WatCam, en examinant l'influence de plusieurs paramètres clés : angle de prise de vue, luminosité, résolution des vidéos ou encore état de surface de l'eau. À partir de cette analyse, des recommandations pratiques ont été formulées pour optimiser l'installation et la configuration de WatCam en fonction du contexte d'usage. Ces éléments ont permis de faire progresser WatCam vers un outil de terrain plus robuste, fiable et adapté à une diversité de situations hydrologiques.

En complément, au début de cette année académique, une collaboration avec Samuel Jean, étudiant haïtien, a permis de poursuivre cette dynamique d'amélioration. Celui-ci a mené des études de sensibilité pour évaluer la précision des pixels lors de la sélection des balises et de l'intensité lumineuse, ainsi que des tests pour optimiser l'utilisation de l'application, aboutissant à une nette accélération du traitement.

En résumé, ces contributions successives ont renforcé WatCam dans trois dimensions fondamentales :

- **Performance technique** : grâce à une restructuration du pipeline de traitement inspirée de PyORC, WatCam bénéficie d'une meilleure efficacité et d'un code plus modulaire.
- **Accessibilité et scalabilité** : via une interface repensée, une meilleure ergonomie et la possibilité de traitement déporté sur serveur, l'outil s'adresse désormais à un public plus large, y compris non-expert.
- **Précision des mesures** : à travers un kit terrain dédié et une analyse fine des

paramètres vidéo influents, WatCam gagne en fiabilité et en adaptabilité sur le terrain.

## 3.2 Architecture Générale de l'Application WatCam

L'architecture de WatCam repose sur une structure modulaire, organisée en trois couches principales :

— **Interface utilisateur (front-end)**.
— **Logique métier et traitement des données (back-end)**.
— **Utilitaires partagés**.

Ce découpage vise à assurer une meilleure lisibilité du code, une maintenance facilitée et une évolutivité du projet dans le temps.

### 3.2.1 Interface Utilisateur

Le front-end est conçu à l'aide des bibliothèques *Kivy* et *KivyMD*, qui permettent de développer des interfaces graphiques modernes, adaptatives et multiplateformes.

— **Kivy** est un framework Python open source dédié à la création d'applications à interface graphique (GUI) compatibles avec Windows, macOS, Linux, Android et iOS.
— **KivyMD** complète Kivy en y ajoutant des composants respectant les principes du Material Design de Google, apportant ainsi une esthétique cohérente et professionnelle à l'application.

L'interface repose sur des fichiers `.kv`, qui définissent la structure visuelle et les interactions de chaque écran. À chaque fichier `.kv` est associé à un fichier Python correspondant, qui en gère la logique. Par exemple :

— `piv.kv` définit l'interface utilisateur dédiée à l'analyse PIV
— `piv.py` gère la logique métier associée.

La navigation entre les différentes vues est assurée par un gestionnaire d'écrans (`MDScreenManager`), permettant des transitions fluides entre les menus principaux : liste des projets, détails d'un projet spécifique, ou étapes successives de traitement.

Les composants graphiques de l'interface — boutons, barres de progression (`MDProgressBar`), boîtes de dialogue (`MDDialog`), onglets ou menus — sont fournis par KivyMD, garantissant une expérience utilisateur fluide et ergonomique.

### 3.2.2 Traitement des Données

Le back-end assure la gestion des projets, du traitement des données et des opérations sur les fichiers. Il s'appuie sur des bibliothèques scientifiques telles que *xarray*, *NumPy*, et *OpenCV* pour effectuer des calculs et des manipulations de données.

Les fichiers notables incluent :
- `saving_project_data.py` : gère la création, la persistance et la gestion des projets. Il inclut des méthodes pour charger, sauvegarder et valider les données des projets.
- `mask.py` : applique des masques spatiaux aux données PIV pour exclure les zones non pertinentes ou bruitées.
- `riverapp_flow_calculator.py` : effectue le calcul des vitesses d'écoulement à partir des champs de vitesse obtenus via l'analyse PIV.
- `video_filters.py` : applique des filtres vidéo tels que CLAHE, gamma, et normalisation pour améliorer la qualité des données.

Les traitements longs, susceptibles de bloquer l'interface, sont systématiquement délégués à des threads distincts (`Thread`), assurant ainsi une interaction fluide et sans interruption pour l'utilisateur. Par exemple, le calcul PIV est exécuté dans un thread séparé pour éviter de bloquer l'interface utilisateur.

L'interaction entre l'interface graphique et la logique de traitement repose sur des appels de fonctions synchronisés et le passage d'objets Python ou de fichiers JSON. Les données sont souvent sauvegardées dans des fichiers `.json` ou `.nc` (NetCDF) pour garantir leur persistance et leur portabilité.

### 3.2.3 Utilitaires Partagés et Configuration

Des composants transversaux, définis dans des fichiers comme `definitions.py`, assurent la centralisation des constantes, des chemins et des configurations communes. Par exemple, `PROJECTS_DIR` définit le répertoire racine des projets, tandis que `PROJECT_DEFAULT_STRUCT` fournit une structure par défaut pour les fichiers de configuration des projets.

De plus, des modules spécifiques tels que `Pyorc` interviennent dans le traitement des vidéos et des données issues de campagnes de mesures. `Pyorc` est utilisé pour la gestion des configurations de caméra et le traitement des vidéos.

### 3.2.4 Organisation et Flux de Projet

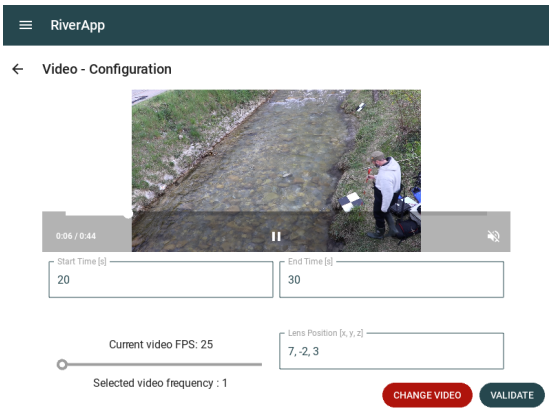
WatCam est conçu autour d'un modèle basé sur des projets. Chaque projet dispose d'un répertoire dédié, d'un fichier `"nom du projet".json` contenant les métadonnées,

et d'un ensemble d'étapes successives allant de la capture vidéo jusqu'à l'analyse par vélocimétrie d'image (PIV). Chaque étape est marquée comme terminée dans le fichier `project.json` via le champ `steps_done`, permettant à l'application de suivre la progression de l'utilisateur.

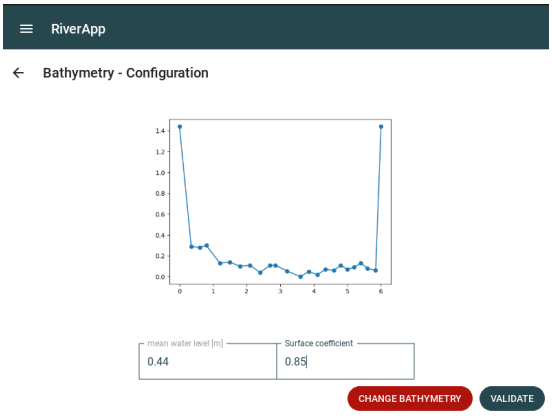
**Flux de projet** Les étapes principales incluent :

1. **Configuration vidéo** : L'utilisateur sélectionne une vidéo et configure les paramètres tels que l'intervalle de temps, le nombre de frames par seconde de vidéos ainsi que la position de la caméra (Figure 3.1a).
2. **Bathymétrie** : l'utilisateur importe ses données bathymétriques sous format `.txt`, donne la hauteur moyenne de l'eau ainsi que le coefficient de surface (Figure 3.1b).
3. **Détection des balises** : Les balises GCP (points de référence) sont configurées pour calibrer les données via le fichier `gcp_detection` et peut les calibrer lui-même si la détection éprouve des difficultés. L'utilisateur doit aussi donner les distances relatives entre chaque point (Figure 3.1c).
4. **Application de filtres vidéo** : Les filtres sont appliqués pour améliorer la qualité des données. L'utilisateur choisit lui même quel filtre appliquer pour faire ressortir les particules afin d'améliorer la précision de la PIV (Figure 3.1d).
5. **Analyse PIV** : Le calcul PIV est exécuté pour extraire les données de vélocimétrie.
6. **Post-traitement** : L'utilisateur place le transect [1]. Ensuite, les résultats sont visualisés et sauvegardés, incluant les vitesses d'écoulement et les images des transects (Figure 3.1e).

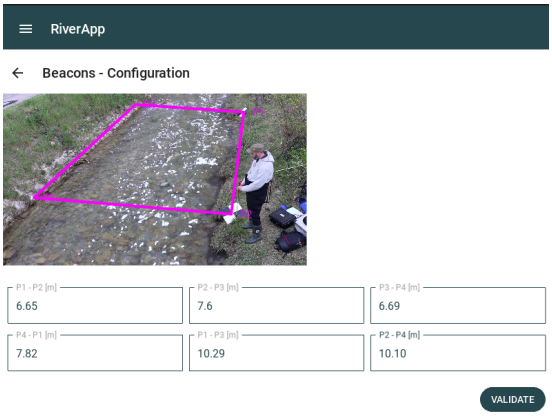
1. Désigne la traversée, selon une ligne imaginaire, d'un espace géographique afin d'en analyser les composantes paysagères. Ici, elle représente le segment sur lequel on veut calculer le débit. sur la zone dont il souhaite obtenir le débit [31]



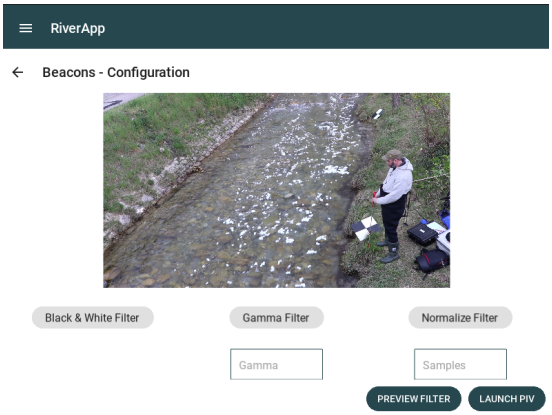

(a) Ecran de configuration de la vidéo




(b) Ecran de la bathymétrie




(c) Ecran de détection des balises




(d) Ecran des filtres




(e) Ecran de post traitement

FIGURE 3.1 – Etapes de WatCam dans sa version initiale (Septembre 2024)

A la fin du projet, le dossier possède la structure suivante :

— **Résultats PIV** : `piv_vectors.csv` qui contient les vecteurs individuels. Ainsi que `piv.nc` qui contient les données globales compressées.

— **Paramètres du projet** : stockés dans `project_info.json`.

## 3.3 Analyse PIV dans WatCam

### 3.3.1 Rappel du contexte méthodologique

Comme présenté dans l'état de l'art, la vélocimétrie par image de particules (PIV) repose sur la mesure du déplacement apparent de structures visuelles ou de traceurs à la surface de l'eau, entre deux images successives d'une séquence vidéo. Ces déplacements sont estimés par corrélation croisée dans des zones locales appelées « fenêtres d'interrogation », et permettent de reconstruire un champ vectoriel de vitesses. Ce champ, une fois correctement géo-référencé, est exploité pour estimer un débit instantané à travers une section transversale connue.

WatCam s'inscrit dans cette logique, mais adapte la méthode pour l'appliquer à des contextes non contrôlés : rivières naturelles, captations mobiles, et vidéos de qualité variable. L'analyse s'effectue de manière entièrement automatisée, tout en restant paramétrable par l'utilisateur.

### 3.3.2 Implémentation dans WatCam

Concrètement, dans WatCam, le prétraitement des vidéos et le découpage en fenêtres sont réalisés dans le module pyorc.service.velocimetry. La classe VelocityFlowProcessor est responsable de la gestion des vidéos et de l'application des méthodes de vélocimétrie. La génération des vecteurs de vitesse et leur filtrage sont réalisés dans le module `saving_project_data.py`. La méthode `generate_piv` applique les étapes nécessaires pour produire les fichiers PIV.

Le module responsable de cette analyse dans WatCam est `analysePIV.py`. Ce module suit les étapes suivantes :

1. **Prétraitement de la vidéo** : les images extraites de la vidéo sont converties en niveaux de gris, éventuellement filtrées (amélioration de contraste, lissage) pour accentuer les structures visibles à la surface.
2. **Découpage en fenêtres** : chaque image est segmentée en fenêtres carrées de taille fixe (paramétrable, généralement entre 32x32 et 64x64 pixels).
3. **Corrélation croisée** : pour chaque fenêtre, une recherche locale est effectuée dans l'image suivante afin d'identifier la zone de plus forte corrélation. La méthode Fast Fourier Transform (FFT) est utilisée pour estimer le décalage spatial $\Delta\vec{x}$ entre deux images successives via maximisation de la fonction de corrélation croisée. Un vecteur vitesse local est alors calculé de cette manière :

$$\vec{v}(x, y) = \frac{\Delta\vec{x}}{\Delta t}$$

4. **Extraction des vecteurs** : le décalage optimal est converti en vecteur vitesse en tenant compte de la fréquence vidéo et des dimensions physiques (calibration spatiale en m/pixel).

5. **Filtrage et validation** : Les vecteurs aberrants sont détectés à l'aide de filtres statistiques tels que `filter_spatial_median` ou `filter_temporal_velocity`, afin d'assurer la cohérence spatiale et temporelle des résultats.
6. **Génération de la carte de vitesse** : les vecteurs sont regroupés dans une structure tabulaire (pandas DataFrame) avec leur position et exportés au format `.csv` (fichier `piv_vectors.csv`). Un fichier `piv.nc` (NetCDF) [2] stocke les données complètes pour une visualisation ou un traitement ultérieur.

### 3.3.3 Calibration et conversion métrique

La précision de l'analyse PIV dépend fortement de la calibration spatiale. *WatCam* effectue une calibration à partir de coordonnées GPS, avec conversion via un fichier de calibration fourni par l'utilisateur. Ces données permettent de convertir les vecteurs en unités physiques (m/s) et d'alimenter les modules de calcul de débit par intégration sur la section mouillée.

## 3.4 Système de stockage des données et persistance des projets

Le système de stockage et de persistance des projets constitue un élément central de l'architecture de *WatCam*, garantissant à la fois la réplicabilité des analyses et la traçabilité des données produites. Chaque projet est organisé sous la forme d'un répertoire de travail autoportant, contenant l'ensemble des ressources nécessaires à l'exécution et à la reprise de l'analyse hydrologique.

Lorsqu'un projet est terminé, *WatCam* initialise automatiquement une structure de répertoires hiérarchiques, où sont placés les fichiers vidéo d'entrée, les images extraites, les résultats d'analyse (champs de vitesse, mesures de débit, figures), ainsi qu'un fichier de métadonnées au format JSON (`project_info.json`).
Ce fichier sert de point d'ancrage pour l'ensemble du projet : il contient les paramètres de calibration, les coordonnées géographiques, les paramètres utilisés pour les calculs, et l'historique des opérations effectuées. Dans le dossier du projet, nous retrouvons aussi le transect_plot qui représente une visualisation graphique des vitesses d'écoulement le long du transect défini.

Les résultats intermédiaires, tels que les vecteurs PIV, les segments de transects ou encore les profils de vitesse sont stockés dans des formats structurés (`.csv`, `.npy`, `.json`) facilitant à la fois l'analyse manuelle et la réutilisation automatique par d'autres

2. Network Common Data Form : format de fichier conçu pour prendre en charge la création de données scientifiques ainsi que l'accès à ces données et leur partage. Il est largement utilisé parmi les communautés océanographiques et atmosphériques pour stocker des variables, telles que la température, la pression, la vitesse du vent et la hauteur des vagues.[32]

modules. Cette approche modulaire et transparente favorise l'intégration de *WatCam* dans des chaînes de traitement plus larges (SIG, modèles hydrauliques, bases de données environnementales).

## 3.5 Gestion des performances et optimisation des calculs

La gestion des performances dans *WatCam* est un enjeu majeur, en raison du volume de données manipulé (vidéos HD, séquences d'images, champs vectoriels) et de la complexité algorithmique de certaines opérations, notamment l'analyse PIV ou le recalage géométrique. Le système a donc été conçu pour exploiter efficacement les ressources matérielles disponibles et minimiser les temps de traitement, tout en préservant la précision des calculs.

Plusieurs stratégies d'optimisation sont mises en œuvre. Tout d'abord, l'extraction et le traitement des images s'appuient sur des bibliothèques performantes telles que `OpenCV` et `NumPy`, qui exploitent des opérations vectorisées en mémoire et le parallélisme multi-cœurs. Les boucles lourdes de l'analyse PIV, notamment la corrélation croisée, sont optimisées via des transformées de Fourier rapides (FFT) et une réduction de la redondance spatiale par découpage en fenêtres adaptatives.

Ensuite, un mécanisme de mise en cache des résultats intermédiaires est implémenté. Par exemple, les vecteurs PIV déjà calculés pour une vidéo ne sont pas recalculés lors des relances successives du projet, sauf modification explicite des paramètres. Cela réduit drastiquement le coût de ré-exécution lors d'analyses itératives. Néanmoins, à chaque ré-exécution, l'application ajoute une image à coté de la précédent comme indiquer à la Figure 3.2. Nous reviendrons sur ce problème dans la partie lié au bugs.

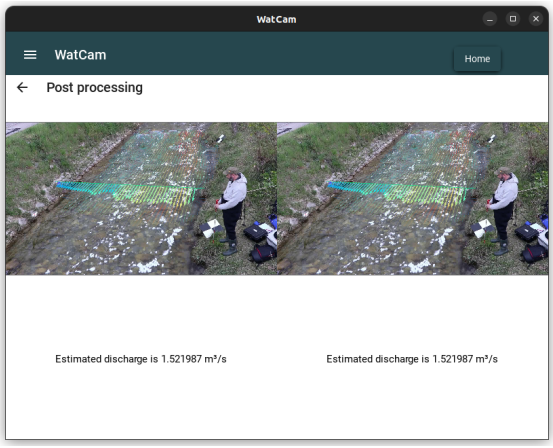



FIGURE 3.2 – Bug lors de la ré-exécution d'un projet.

Enfin, le temps de traitement est surveillé de manière systématique et consigné dans les journaux du projet. Cette information permet d'identifier les goulots d'étranglement et de guider les futures optimisations. Ainsi, *WatCam* allie une approche rigoureuse en termes d'exactitude scientifique et une attention soutenue à la performance opérationnelle, conditions nécessaires à son déploiement dans des contextes de terrain.

## 3.6 User Stories

### 3.6.1 Introduction

Dans le cadre d'un développement agile, les *user stories* — ou récits utilisateurs — permettent de capturer les besoins fonctionnels d'une application à travers des scénarios concrets, exprimés du point de vue des utilisateurs finaux. Chaque histoire suit une structure typique :

*En tant que [type d'utilisateur], je souhaite [objectif] afin de [bénéfice attendu].*

Cette méthode facilite la communication entre développeurs, concepteurs et parties prenantes, tout en guidant les priorisations, la planification des tâches, et la validation itérative des fonctionnalités. Dans le cadre de *WatCam*, ces récits traduisent les attentes d'un public diversifié en fonctionnalités concrètes, en particulier autour de la gestion de projet, du traitement vidéo, et de la restitution des résultats. Au cours de la réalisation de ce mémoire, nous avons souhaité impliquer les utilisateurs potentiels de WatCam, en particulier les membres du groupe GCE, afin de recueillir leurs retours et affiner les user stories. Dans cette optique, nous avons entrepris la création d'un fichier exécutable (`.exe`) permettant à chacun d'entre eux de tester librement l'application. Cependant, des difficultés techniques ont freiné l'aboutissement de cette version exécutable, qui reste incomplète à ce jour. Il serait pertinent, dans le cadre de futurs travaux, de finaliser ce composant pour faciliter le déploiement et la diffusion de WatCam. Les détails et perspectives liés à ce développement sont présentés dans le Chapitre 7 : Perspectives d'amélioration et développements futurs.

Ces User-stories vont structurer le développement de WatCam autour de la valeur ajoutée pour l'utilisateur, en alignant l'évolution du code avec des objectifs clairs, mesurables et adaptés aux contraintes du terrain. Elles assurent que l'application restera cohérente, évolutive et centrée sur ses finalités concrètes. La suite de cette section regroupe donc les *user stories* par fonctionnalité-clé du parcours utilisateur.

### 3.6.2 Interface générale et navigation

En tant qu'utilisateur :

— Je souhaite accéder à un bouton **Accueil** toujours visible, pour revenir rapidement à l'écran principal.

— Je souhaite disposer d'un bouton **Retour** pour naviguer facilement entre les écrans.
— Je souhaite disposer d'un bouton **Home** pour revenir à l'écran des projets avec un seul bouton.

### 3.6.3 Création d'un nouveau projet

En tant qu'utilisateur :
— Je souhaite **créer un nouveau projet** depuis l'écran d'accueil.
— Je souhaite pouvoir **saisir un nom de projet personnalisé** et qu'il soit enregistré automatiquement dans un dossier dédié.
— Je souhaite **sélectionner une vidéo d'entrée** depuis mon système de fichiers.
— Je souhaite pouvoir **définir l'intervalle temporel** à analyser.
— Je souhaite **choisir la fréquence d'analyse (nombre de FPS)**.
— Je souhaite spécifier la **position de la caméra** (angle, orientation, hauteur).
— Je souhaite être capable d'**importer un fichier de configuration** qui permet de remplir directement : la vidéo d'entrée, l'intervalle temporel, le nombre de FPS et la position de la caméra.

### 3.6.4 Bathymétrie

En tant qu'utilisateur :
— Je souhaite **importer un fichier .txt** pour avoir ma bathymétrie.
— Je souhaite pouvoir **définir ma hauteur moyenne d'eau**.
— Je souhaite pouvoir **définir mon coefficient de surface**.

### 3.6.5 Beacons

En tant qu'utilisateur :
— Je souhaite **placer manuellement les balises (GCP)** sur la vidéo.
— Je souhaite pouvoir **définir les distances réelles entre les balises**.
— Je souhaite également **importer un jeu de balises déjà existant** (en pixels ou en coordonnées GPS).
— Je souhaite pouvoir **améliorer la qualité visuelle de la vidéo** à l'aide de filtres (flou, détection de bords, etc.).
— Je souhaite **ajuster dynamiquement les paramètres des filtres** via une interface (curseurs, champs numériques).
— Je souhaite **prévisualiser le rendu des filtres** avant de lancer l'analyse.
— Je souhaite pouvoir **enregistrer les versions filtrées de mes vidéos**.
— Je souhaite **définir manuellement la section d'analyse** (zone active du transect dans la rivière).

### 3.6.6 Filtres

En tant qu'utilisateur :
- Je souhaite pouvoir **améliorer la qualité visuelle de la vidéo** à l'aide de filtres (flou, détection de bords, etc.).
- Je souhaite **activer ou désactiver des filtres** indépendamment les uns des autres.
- Je souhaite que, quand un filtre est activé, je puisse **régler son intensité**.
- Je souhaite que **l'image en preview s'actualise** à chaque fois que change l'intensité d'un filtre.
- Je souhaite avoir un bouton **"Reset Filters"** qui enlève tous les filtres de l'image en preview afin de recommencer le processus de filtre de zéro.
- Je souhaite **obtenir la vidéo avec l'intervalle de temps sélectionné avec les filtres appliqués dessus** afin de vérifier le rendu.
- Je souhaite pouvoir lancer l'analyse PIV, visualiser son état d'avancement via une indication de progression, et l'arrêter à tout moment si nécessaire, afin de garder le contrôle sur le processus et éviter les traitements inutiles.

### 3.6.7 Post-Process

En tant qu'utilisateur :
- Je souhaite **définir le transect** sur lequel je veux connaître le débit.
- Je souhaite pouvoir **lancer l'analyse PIV** à l'aide d'un bouton "Launch PIV".

### 3.6.8 Résultats PIV

En tant qu'utilisateur :
- Je souhaite **afficher le champ de vecteurs** superposé à l'image, représentant les vitesses de surface.
- Je souhaite que les vecteurs soient **colorés selon leur intensité** pour une lecture intuitive des écoulements.
- Je souhaite voir le **débit total estimé** affiché sous forme numérique.
- Je souhaite **avoir les détails du projet** sous un format JSON.
- Je souhaite que mon nouveau projet **apparaisse dans la liste des projets existants**.
- Je souhaite **avoir le "masked plot" et "transect plot"** dans mon dossier du projet afin de pouvoir observer les résultats.

### 3.6.9 Synthèse

Cette modélisation par *user stories* permet de structurer le développement de WatCam autour des attentes fonctionnelles concrètes de ses utilisateurs cibles. Elle assure également une cohérence dans la conception de l'interface graphique, le traitement vidéo, et l'exploitation des résultats, tout en préparant le terrain pour des tests utilisateurs ciblés et des itérations agiles. Il est important de souligner que ces user stories constituent

une base évolutive : elles servent à orienter le développement en gardant à l'esprit les objectifs fonctionnels de l'application, tout en restant flexibles pour s'adapter à l'évolution des besoins et des usages.

# Chapitre 4

# Développement et apports techniques

## 4.1 Rappel des objectifs

**Implémentation de filtres avancés pour l'analyse PIV**

Une attention particulière a été portée à l'application de filtres vidéo pour améliorer la qualité des résultats de l'analyse par vélocimétrie d'image de particules (PIV). Des filtres permettant améliorer la détection des particules et renforcer les champs de vecteurs calculés. Une étude comparative des performances des différentes combinaisons de filtres a été menée pour identifier les configurations optimales dans des conditions variées.

**Analyse de sensibilité des paramètres**

Une analyse systématique d'un vidéo sera réalisée afin de mieux comprendre leur influence sur la précision des mesures et leur impact sur le débit estimé. Cette analyse de sensibilité sera réalisé sur la vidéo VGC (Video Global Challenge) qui a été souvent utilisé comme référence dans les anciens mémoires.

**Refactorisation et modularisation du code**

La structure du code de *WatCam* sera revue afin de rendre l'application plus lisible, maintenable et évolutive. La mise en place d'une architecture modulaire facilitera l'intégration de nouvelles fonctionnalités et garantira la pérennité du projet.

**Développement d'une interface utilisateur conviviale**

Pour élargir l'adoption de *WatCam*, l'application sera dotée d'une interface intuitive et accessible, adaptée aux utilisateurs novices. Cette interface proposera des flux de travail

guidés pour la création et la configuration de projets, ainsi que des options avancées pour les utilisateurs experts.

#### Tests en conditions réelles

*WatCam* sera testé sur des sites de terrain variés pour évaluer ses performances dans des contextes environnementaux réels, notamment en conditions d'inondation et de sécheresse. Ces tests permettront de valider la robustesse et la fiabilité des améliorations techniques, tout en collectant des retours d'expérience de la part des utilisateurs.

### Conclusion

Ces objectifs s'inscrivent dans une démarche visant à faire de *WatCam* un outil performant, accessible et fiable, capable de répondre aux besoins des chercheurs, des gestionnaires environnementaux et des professionnels de la gestion des catastrophes. En combinant des avancées techniques, des améliorations de l'interface utilisateur et des validations sur le terrain, ce projet vise à établir *WatCam* comme une solution fiable pour l'analyse hydrologique basée sur la vélocimétrie d'image de particules.

## 4.2 Implémentation des filtres

La qualité des images vidéo constitue un paramètre fondamental pour assurer la fiabilité des mesures obtenues par vélocimétrie d'image (PIV). Les algorithmes de corrélation utilisés dans la PIV, notamment en version LSPIV, reposent sur la détection de motifs (textures, particules flottantes) se déplaçant à la surface de l'eau. Or, cette détection est fortement influencée par le contraste, la distribution lumineuse, les reflets, et les conditions météorologiques. Afin d'optimiser la lisibilité des séquences et améliorer les performances du traitement, un ensemble de filtres est souvent appliqué en amont de l'analyse [19].

Dans WatCam, les filtres vidéo sont appliqués de manière modulaire avant l'analyse PIV. Chaque frame extraite est transformée à l'aide d'une chaîne de filtres configurables (dans `video_filters.py`), puis transmise au module de vélocimétrie (pyorc.service.velocimetry). Une fonction de prévisualisation (`preview_filters` dans `filter_video.py`) permet à l'utilisateur d'observer l'impact des filtres en amont. Cette architecture améliore la lisibilité des particules et la qualité des résultats PIV en adaptant dynamiquement le prétraitement aux conditions d'éclairage ou de contraste.

#### Filtres disponibles dans WatCam

Afin de montrer correctement, les effets de chaque filtre nous allons utiliser une seule et même image (Figure 4.1) sur laquelle nous allons appliquer chaque filtre.

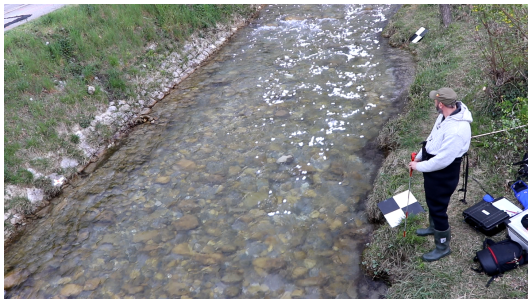


FIGURE 4.1 – Image de base sans filtres

### Normalize

Le filtre `Normalize` (Figure 4.2) ajuste la plage dynamique des niveaux de gris d'une image pour qu'elle s'étende sur toute l'amplitude possible (de 0 à 255). Il agit comme un étirement linéaire de l'histogramme, ce qui permet d'uniformiser la luminosité globale et d'améliorer la visibilité des textures et particules en surface. Cela facilite la détection de structures exploitables pour la corrélation PIV.[19]




FIGURE 4.2 – Exemples d'application du filtre de normalisation

— **Avantages :** corrige des déséquilibres d'exposition ; améliore la détection dans les zones sombres ou claires.
— **Limites :** peu efficace sur des vidéos très bruitées ; ne corrige pas les zones surexposées.

### CLAHE (Contrast Limited Adaptive Histogram Equalization)

Le CLAHE divise l'image en petites régions (tuiles) et applique une égalisation d'histogramme sur chacune d'elles, avec une limitation du contraste pour éviter l'amplification du bruit. Contrairement à l'égalisation classique, il s'adapte localement aux variations de contraste. Cela améliore la visibilité de détails fins et hétérogènes à la surface de l'eau, ce qui est crucial pour le suivi PIV. Il est notamment utilisé dans PIVLab [33] (Figure 4.3).

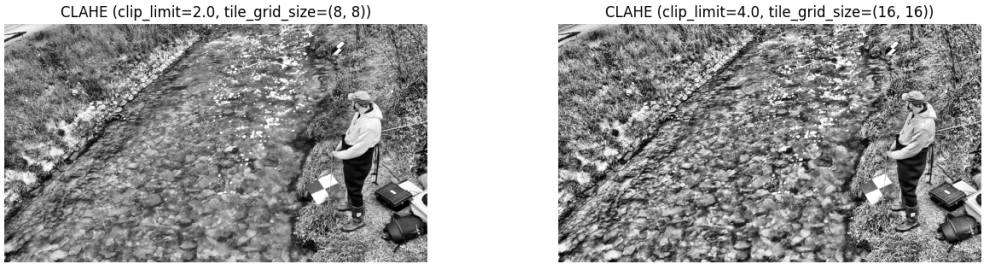



FIGURE 4.3 – Exemples d'application du filtre CLAHE

— **Avantages :** renforce efficacement les motifs de surface ; idéal pour les vidéos contenant des textures subtiles.
— **Limites :** nécessite un bon paramétrage (taille de fenêtre) ; peut introduire des artéfacts si la vidéo est instable.

### Correction Gamma

Le filtre `Gamma` (Figure 8.4) modifie la luminosité d'une image selon une fonction de puissance (exponentielle). Un gamma $< 1$ éclaircit l'image ; $> 1$ l'assombrit. Cela permet d'accentuer certaines plages de luminosité selon les conditions de captation. En PIV, cela aide à mieux faire ressortir les traceurs dans les zones sombres ou trop lumineuses et est utilisé pour corriger les biais de luminosité introduits par les conditions d'éclairage sur le terrain.

— **Avantages :** améliore les contrastes dans des scènes globalement sombres ou lumineuses.

— **Limites :** nécessite un ajustement manuel précis ; peut saturer les zones extrêmes.

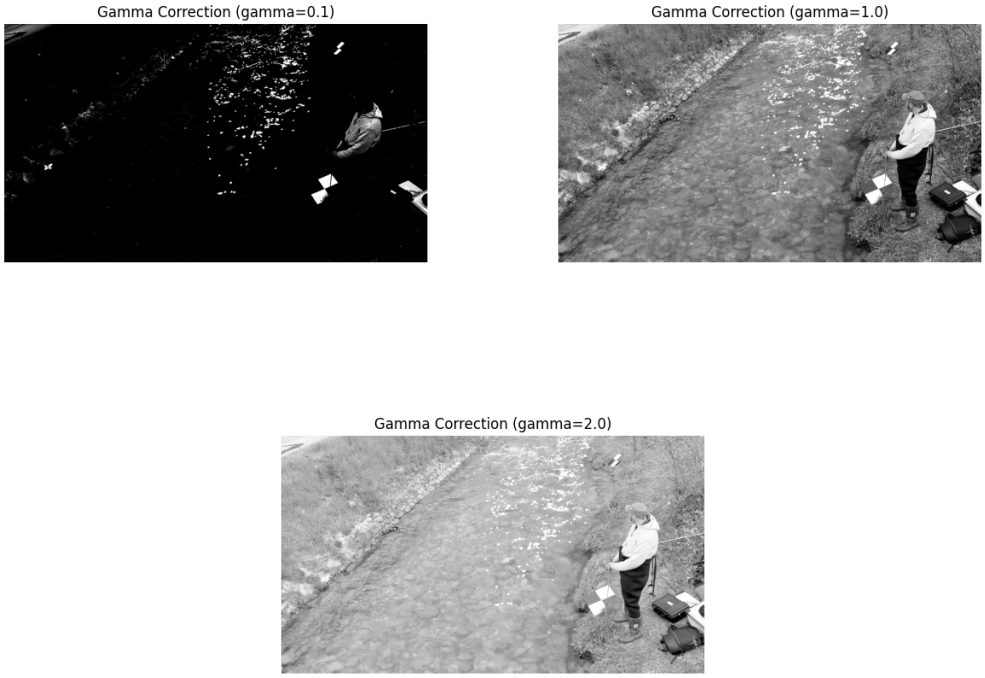



FIGURE 4.4 – Exemples d'application du filtre Gamma

### Conversion en niveaux de gris

La conversion `Grayscale` (noir et blanc) transforme l'image en intensité lumineuse pure, sans couleur. Cette étape est recommandée pour les traitements PIV, car les algorithmes se basent uniquement sur les niveaux de luminance. La conversion en noir et blanc est souvent la dernière étape du prétraitement. Elle permet de réduire la complexité des données, en ne conservant que l'information utile pour la vélocimétrie : l'intensité lumineuse. Cela améliore la rapidité de traitement et la compatibilité avec les algorithmes de corrélation.

— **Avantages :** simplifie les calculs ; évite les biais induits par la couleur.
— **Limites :** perte d'informations lorsque la couleur aide à distinguer les traceurs.

### 4.2.1 Choix d'une image optimale pour l'analyse PIV

La qualité des images utilisées dans une analyse PIV constitue un facteur déterminant de la précision des résultats. Une vidéo bien calibrée, contrastée et correctement orientée permet d'extraire des champs de vitesse fiables et exploitables. À l'inverse, des séquences mal exposées ou déformées peuvent introduire des erreurs systématiques ou rendre l'analyse inopérante [19, 12].

#### Critères d'une image optimale

Une image optimale pour la vélocimétrie optique est caractérisée par les éléments suivants :

— **Présence de traceurs visibles à la surface** : bulles d'air, mousse, végétation flottante ou tout motif naturel suffisamment texturé pour permettre la corrélation entre images successives [11] ;
— **Contraste élevé** entre les particules et le fond liquide, facilitant la détection des déplacements ;
— **Éclairage homogène** sur l'ensemble de la scène, sans surexposition ni ombre marquée, afin d'éviter les discontinuités de luminance ;
— **Caméra stabilisée**, idéalement fixée sur un trépied ou suspendue de manière rigide, pour éliminer les mouvements parasites [25] ;
— **Angle de vue perpendiculaire** à la surface de l'eau, limitant les déformations géométriques et facilitant l'orthorectification du champ de vitesses ;
— **Absence de reflets ou objets parasites** (feuilles statiques, ponts, structures fixes) qui pourraient induire des vecteurs aberrants.

#### Exemples de situations à éviter

À l'opposé, certaines conditions peuvent fortement dégrader la qualité de l'analyse :

— **Surexposition ou sous-exposition** de l'image, entraînant une perte de texture utile pour la corrélation [19] ;
— **Reflets lumineux intenses**, issus du ciel ou du soleil, qui masquent les traceurs ou créent de fausses structures optiques ;
— **Surface d'eau uniforme**, lisse ou stagnante, sans traceurs visibles, rendant la détection impossible ;
— **Caméra mobile ou instable**, introduisant un mouvement artificiel interprété comme une vitesse d'écoulement ;
— **Angle de vue trop rasante**, produisant des effets de perspective importants et rendant difficile la conversion des coordonnées pixels en unités physiques.

En résumé, la réussite d'une analyse PIV dépend avant tout de la qualité optique et géométrique de la vidéo. Une bonne pratique consiste à contrôler systématiquement la

stabilité de la caméra, la présence de traceurs, l'éclairage et l'angle de prise de vue avant toute campagne de mesure, afin de maximiser la robustesse et la fiabilité des estimations de vitesse et de débit.

## 4.3 Analyse de sensibilité

L'analyse de sensibilité constitue une étape clé dans l'évaluation des performances de *WatCam*. Elle vise à déterminer dans quelle mesure certains paramètres influencent les résultats produits par l'analyse PIV, notamment l'estimation du débit.
Voici les différents paramètres étudiés : filtre CLAHE, gamma, fréquence d'image (FPS), durée d'analyse, luminosité (valeur de gris), et normalisation.Ces paramètres ont été sélectionnés car ils influencent directement ou indirectement la qualité des images analysées par la méthode PIV, et donc, la fiabilité des estimations de débit fournies par l'application. L'ensemble des résultats, sous forme de graphes, se trouvent dans la section 8.1 : "Analyse de sensibilité des différents paramètres", se trouvant en annexe.

**Effet de la durée analysée** (Figure 8.1) : intervalle de temps analysé.

- **Observation** : Le débit diminue légèrement avec l'augmentation de la durée. Les séquences très courtes peuvent surestimer le débit.
- **Hypothèse** : Une durée trop brève ne capture pas la variabilité temporelle du courant. À l'inverse, une durée trop longue peut intégrer des phases calmes non représentatives.
- **Recommandation** : Privilégier des durées intermédiaires (5 à 10 secondes) pour un bon compromis.

**Influence de la fréquence d'image (FPS)** (Figure 8.2) : nombre d'images par seconde. Ici, la vidéo analysée dure 1s.

- **Observation** : Les estimations de débit sont plus cohérentes pour des FPS élevés (20–25). On observe aussi une diminution soudaine aux alentours de 12.5 FPS.
- **Hypothèse** : Une fréquence trop basse réduit la résolution temporelle, affectant la détection des déplacements entre images.
- **Recommandation** : Utiliser une fréquence d'acquisition d'au moins 20 images/s pour garantir une fidélité temporelle satisfaisante.

**Effet du filtre CLAHE** (Figure 8.3)

- **Observation** : Le débit estimé diminue légèrement avec l'augmentation de la valeur CLAHE jusqu'à 3 puis augmente fortement et se stabilise.
- **Hypothèse** : Bien que CLAHE améliore le contraste local, des valeurs élevées peuvent générer des artefacts visuels qui nuisent à la corrélation croisée.

— **Recommandation** : Utiliser des valeurs modérées de CLAHE (1.0 à 1.5) pour renforcer les détails sans dégrader l'image.

**Effet du filtre Gamma** (Figure 8.4)

— **Observation** : Le débit diminue progressivement avec l'augmentation du gamma. Les résultats sont les plus stables autour de 1.0.
— **Hypothèse** : Le gamma agit sur la luminosité globale. Une valeur trop faible ($<$ 0.8) ou trop élevée ($>$ 1.2) nuit à la détection des traceurs.
— **Recommandation** : Maintenir des valeurs proches de 1.0 pour un bon équilibre visuel.

**Effet de la luminosité (valeurs de gris)** (Figure 8.5)

— **Observation** : Les résultats sont relativemment stables pour des valeurs de luminosité comprises entre 80 et 200.
— **Hypothèse** : Une trop forte luminosité peut saturer l'image, tandis qu'une luminosité trop faible empêche la distinction des traceurs.
— **Recommandation** : Viser une luminance moyenne bien répartie (autour de 200) pour optimiser la détection.

**Effet de la normalisation** (Figure 4.2)

— **Observation** : Le débit reste stable entre 50 et 200. Des valeurs extrêmes ($>$ 240) entraînent des écarts importants.
— **Hypothèse** : Une normalisation trop poussée peut accentuer des bruits ou des artefacts, faussant les corrélations.
— **Recommandation** : Travailler dans une plage modérée (100 à 200) pour améliorer le contraste sans dégrader la structure de l'image.

### 4.3.1 Conclusion

Cette analyse de sensibilité a mis en lumière l'impact significatif des paramètres de traitement vidéo sur la fiabilité des estimations de débit fournies par WatCam. Si certains réglages, comme la fréquence d'acquisition (FPS) ou la durée de séquence, influencent directement la précision temporelle et statistique de l'analyse, d'autres — tels que les filtres CLAHE, gamma ou la normalisation — modifient subtilement la perception visuelle des traceurs et leur détectabilité par les algorithmes PIV.

Ces résultats confirment l'importance d'un calibrage rigoureux des paramètres en fonction des conditions spécifiques d'observation (lumière, contraste, nature des traceurs). Ils montrent également qu'il n'existe pas de configuration unique optimale, mais plutôt une combinaison de plages de valeurs à privilégier selon les cas.

## 4.4 Refonte logicielle et architecture MVC

Une part importante du travail a porté sur la réorganisation interne du code, en particulier autour de l'écran principal de gestion des projets. Cette section décrit pourquoi une refonte était nécessaire, comment elle a été conduite, et les bénéfices apportés par l'adoption d'une architecture logicielle plus modulaire.

### 4.4.1 Pourquoi une refonte ? Les limites du code initial

À l'arrivée sur le projet, l'application fonctionnait correctement, mais son architecture logicielle s'avérait peu structurée. De nombreux fichiers concentraient à la fois des responsabilités d'affichage (interface utilisateur), de logique métier (traitements de projets) et de manipulation de fichiers (accès disque, lecture/écriture). Cette absence de séparation nuisait à la lisibilité, à la maintenance et à l'évolutivité du code.

Par exemple, le fichier `projects.py` cumulait la gestion de l'affichage, la création et la suppression de dossiers, la gestion d'événements utilisateur, ainsi que des opérations asynchrones. Ce type de module monolithique compliquait toute tentative de modification ou de test unitaire : chaque ajustement avait un impact potentiel sur l'ensemble du système.

Cette complexité justifiait une refonte en profondeur, orientée vers une architecture plus claire et plus modulaire.

### 4.4.2 Principe de l'architecture MVC

Pour répondre à ces limitations, nous avons opté pour une refonte partielle basée sur l'architecture **MVC** (Modèle–Vue–Contrôleur). Le principe central est de séparer clairement les rôles de chaque composant, ce qui rend le code plus lisible, plus modulaire et plus facile à faire évoluer.

- **Modèle (Model)** : Gère les données de l'application et la logique métier (ex : les opérations sur les projets).
- **Vue (View)** : Affiche les éléments à l'écran et capture les interactions utilisateur.
- **Contrôleur (Controller)** : Sert d'interface entre les deux. Il interprète les actions utilisateur, applique la logique correspondante et notifie la Vue des résultats.

### 4.4.3 Étude de cas : refonte du module `projects.py`

#### Avant refonte : un module monolithique

Le fichier `projects.py` assurait à lui seul les fonctions suivantes :

- Affichage de la liste des projets et des boutons d'action.
- Ouverture des fenêtres de dialogue (erreurs, confirmations).

— Création/suppression de répertoires sur le disque.
— Traitement de certains événements dans des threads.

Ce type d'architecture est qualifié de *monolithique* : un modèle de développement logiciel traditionnel où tous les composants d'une application sont regroupés en une seule unité cohérente, formant un bloc unique, sans séparation claire des rôles. Ce modèle entraîne des dépendances croisées, une forte complexité, et une fragilité du code. Il était difficile de faire évoluer une fonction sans risquer d'en altérer d'autres.

**Après refonte : séparation en composants MVC**

Le module a été entièrement repensé pour suivre l'approche MVC. Il a été divisé en trois composants principaux :

— `ProjectsController` : reçoit les événements de l'interface (clics), exécute les opérations métier dans des threads si nécessaire, et notifie la vue du résultat via des callbacks.
— `Projects.py` : vue Kivy associée à un fichier `.kv`, elle affiche les projets et envoie les événements utilisateur au contrôleur.
— `SavingProjectData.py` : modèle responsable des interactions avec le système de fichiers (création, suppression, chargement).

Deux modules complémentaires sont utilisés pour améliorer la modularité :

— `DialogFactory.py` : centralise la création des fenêtres de dialogue (erreurs, confirmations, formulaires).
— `ProjectCard / NoResults` : composants graphiques pour représenter un projet ou l'absence de projet.

**Résultats de cette refonte**

La nouvelle structure de ce module est illustré à la Figure 4.5. Cette nouvelle organisation a rendu le module :
— Plus robuste : les erreurs sont isolées, les responsabilités mieux définies.
— Plus simple à tester : on peut tester la logique sans lancer l'interface.
— Plus évolutif : ajouter une fonctionnalité (comme trier ou renommer un projet) se fait sans impacter l'affichage.
— **Un exemple marquant** : l'ajout de la fonctionnalité *renommer un projet* s'est fait uniquement via le contrôleur et le modèle — sans toucher à la vue ni aux autres fonctions existantes.

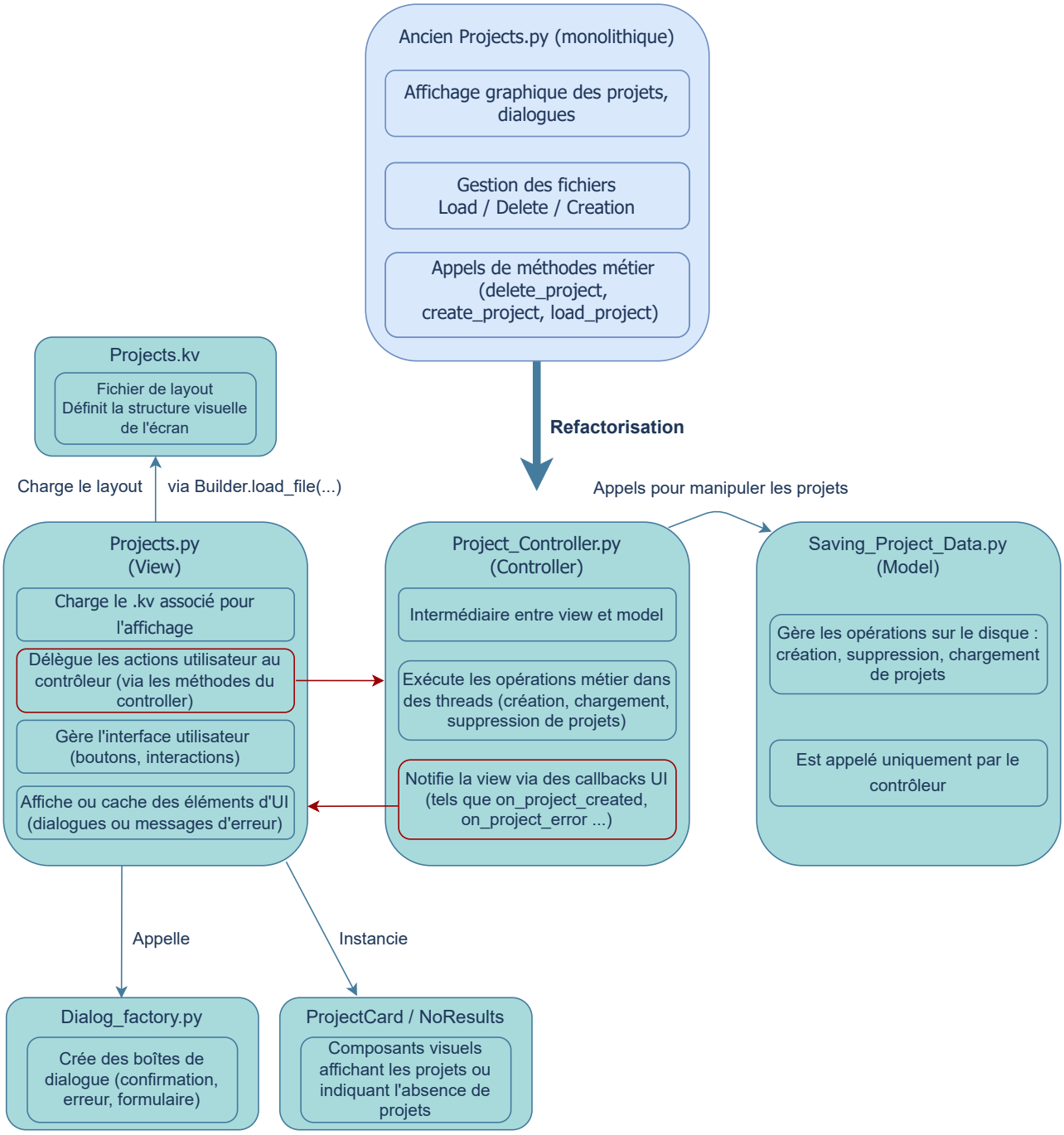



FIGURE 4.5 – Schéma du découpage MVC de l'écran Projects

### 4.4.4 Pourquoi cette architecture n'a pas été appliquée partout

Bien que la refonte MVC ait apporté de réels bénéfices, elle n'a pas été appliquée à l'ensemble de l'application pour les raisons suivantes :

— Certains écrans sont très simples (affichage statique, peu d'interactions) et ne justifient pas une séparation MVC.
— D'autres modules sont stabilisés et rarement modifiés, donc peu prioritaires pour une refonte.
— Le découpage MVC demande un effort de conception initial : mieux vaut donc le réserver aux modules critiques ou évolutifs.

La migration vers une architecture plus modulaire s'est donc faite de manière progressive, en commençant par les composants les plus sollicités par l'utilisateur.

### 4.4.5 Pourquoi MVC plutôt qu'une autre architecture ?

Parmi les architectures possibles (MVP, MVVM, etc.), le choix du MVC s'est imposé comme le plus adapté à *WatCam* pour plusieurs raisons :

— **Simplicité** de mise en place avec Kivy.
— **Clarté** de séparation des rôles.
— **Facilité** de compréhension pour des profils variés (ingénieurs, développeurs, chercheurs).

Voici une brève comparaison des architectures envisageables (Figure 4.1) :

| Architecture | Typiquement utilisée dans... | Avantages et inconvénients |
|---|---|---|
| **MVC** (Model — View — Controller) | Interfaces graphiques classiques (desktop, web) | Séparation claire des responsabilités. Facile à mettre en œuvre. Compatible avec Kivy. Bien adapté aux applications avec logique utilisateur modérée. |
| **MVP** (Model — View — Presenter) | Applications Android, Java Swing | Variante de MVC où le Presenter gère activement les interactions et la logique. Plus structurant mais plus verbeux. Moins naturel avec Kivy. |
| **MVVM** (Model — View — ViewModel) | Frameworks modernes basés sur la réactivité (ex. : React, Vue, Flutter) | Architecture très puissante pour l'UI dynamique, avec mise à jour automatique entre l'interface et les données ("data-binding"). Complexe à mettre en œuvre ici, peu pertinent dans un contexte non réactif. |

Table 4.1 – Comparaison de différentes architectures logicielles orientées interface

Le MVC (Model-View-Controller) s'est révélé être un bon compromis pour WatCam, car il allie simplicité de mise en œuvre, clarté architecturale et robustesse. Sa séparation claire entre les données (Model), l'interface utilisateur (View) et la logique de contrôle (Controller) facilite la maintenance et l'évolution du code, ce qui est essentiel dans un projet en développement comme WatCam.

De plus, cette architecture est bien adaptée aux frameworks comme Kivy, qui encouragent une structuration modulaire sans imposer une complexité excessive. Contrairement à des architectures plus modernes mais plus complexes telles que MVVM, le MVC permet à l'équipe de développement de rester agile tout en assurant un découplage fonctionnel suffisant pour gérer les différentes composantes de l'application.

Enfin, dans le contexte spécifique de WatCam, qui nécessite à la fois une interface graphique réactive et une gestion claire des flux de données issus de capteurs, MVC offre un cadre éprouvé qui répond efficacement aux besoins sans surcharger le développement avec des patterns plus lourds.

### 4.4.6 Un retour d'expérience à la fois technique et pédagogique

Même sans mesure chiffrée ou test automatisé, la refonte du module `projects.py` a eu des effets concrets et bénéfiques.

— Le code est devenu plus lisible, avec des responsabilités clairement séparées.

— La distinction entre interface graphique, logique métier et accès aux fichiers rend les modifications plus sûres et localisées.
— Il devient possible (et plus simple) de tester certaines parties sans passer par l'interface — ce que nous avons commencé à faire pour le contrôleur.

Au-delà des aspects techniques, cette refonte a été l'occasion de mettre en œuvre des principes abordés en théorie : modularité, découplage, gestion des dépendances. Travailler à partir d'un module existant, plutôt que tout réécrire, nous a permis de mieux comprendre comment améliorer un projet sans risque de le détériorer.

Elle nous a aussi amenés à repenser notre manière de développer : voir le code non plus comme une suite de correctifs ou d'ajouts ponctuels, mais comme une structure pensée pour durer et évoluer dans le temps.

**En résumé**, cette refonte partielle basée sur le MVC a permis de renforcer la stabilité du code, d'en améliorer la structure, et de créer une base plus saine pour les évolutions futures de WatCam.

## 4.5 Améliorations interface utilisateur

La prise en main initiale de *WatCam* s'est révélée peu intuitive, notamment pour les utilisateurs non familiers avec l'environnement de développement ou les formats de données attendus. Afin d'améliorer l'expérience utilisateur et de renforcer la robustesse de l'application, plusieurs ajustements ont été apportés à l'interface et à la gestion des erreurs.

### 4.5.1 Problèmes initiaux

L'une des principales faiblesses de la version initiale de *WatCam* résidait dans sa sensibilité aux erreurs de manipulation. Par exemple :

— Chargement d'un fichier incorrect — comme une vidéo dans un format non supporté.
— Bathymétrie au mauvais format — entraînait un plantage complet de l'application. Ce manque de résilience compromettait l'usage fluide du logiciel, notamment en contexte de terrain.
— Apparition d'un point rouge à chaque clique droit.
— Lorsque la lens position était laissé vide après avoir cliqué sur le champ pour écrire et puis autre part

**Fix**

Pour résoudre ces problèmes, plusieurs modifications ont été implémentées :

— Mise en place de blocs `try...except` autour des opérations critiques, permettant de capturer les erreurs et d'afficher un message explicite à l'utilisateur sans faire planter l'application.
— Ajout de filtres intelligents dans les boîtes de dialogue d'importation : seuls les formats compatibles sont désormais visibles. Par exemple, lors du chargement d'une vidéo, seuls les fichiers `.avi`, `.mp4`, et `.MP4` sont affichés. De même, pour la bathymétrie, seuls les fichiers `.txt` sont proposés.
— Redirection automatique du navigateur de fichiers vers le répertoire racine de *WatCam*, afin de faciliter la navigation. Auparavant, l'utilisateur était redirigé vers le répertoire système, ce qui entraînait confusion et perte de temps.

### 4.5.2 Ecran des projets

Au cours de l'utilisation de *WatCam*, nous avons été confrontés à une limitation importante : l'impossibilité de renommer un projet après sa création. Cette contrainte devenait problématique en cas d'erreur de nommage ou de changement de contexte d'analyse. Pour y remédier, nous avons intégré une nouvelle fonctionnalité à l'interface : un bouton *"Rename"* accessible directement sur chaque projet (voir Figure 4.6). Cette option permet à l'utilisateur de modifier rapidement le nom d'un projet sans passer par les dossiers système ou modifier les fichiers manuellement.

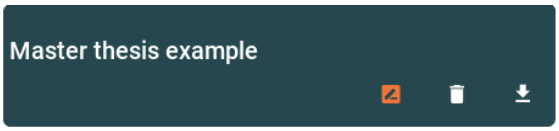



FIGURE 4.6 – Un projet dans WatCam avec les options *"Rename"*, *"Delete"* et *"Upload"*

Par ailleurs, avec l'augmentation progressive du nombre de projets générés, la navigation devenait de plus en plus difficile. Pour améliorer l'organisation, nous avons ajouté une fonctionnalité de tri personnalisable. Un bouton dédié permet désormais de trier les projets selon trois critères : ordre alphabétique, date de création, ou date de dernière modification. Le tri s'effectue en saisissant le paramètre souhaité dans le champ prévu à cet effet (voir Figure 4.7).

Ces ajouts contribuent à rendre l'interface plus ergonomique et adaptée à un usage opérationnel, notamment lors de campagnes de terrain ou de traitements sur de nombreux jeux de données.

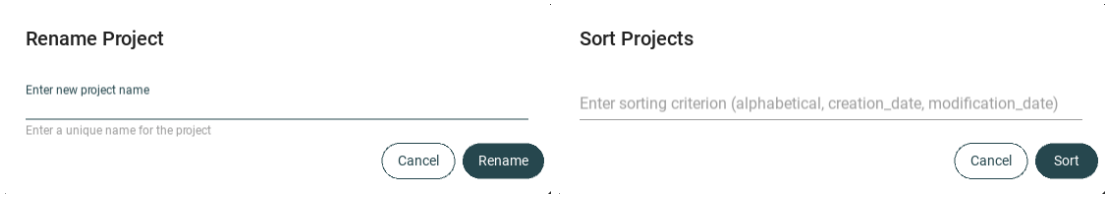



FIGURE 4.7 – Pop-up des nouveaux boutons "Rename" et "Sort"

Au cours du développement, nous avons constaté qu'il n'était pas possible de revenir directement à l'écran d'accueil listant les projets une fois qu'un projet était en cours d'édition. L'utilisateur ne pouvait revenir à l'écran des projets une fois lancé dans un projet, ce qui rendait l'expérience peu ergonomique. Pour pallier cette limitation, nous avons ajouté un bouton *"Home"* visible en permanence, permettant de retourner instantanément à l'écran des projets, quelle que soit l'étape atteinte dans le flux de travail (voir Figure 4.8). Cette amélioration simplifie considérablement la navigation au sein de l'application et renforce son intuitivité.




FIGURE 4.8 – La nouvelle `Top bar` de WatCam

### 4.5.3 Ecran des filtres

Initialement, l'écran dédié aux filtres dans WatCam présentait plusieurs incohérences d'interface : bien que certains filtres comme *Gamma*, *Normalize*, *Edge Detection*, *Noir et Blanc* et *Time Difference* étaient visibles, ceux-ci n'étaient pas fonctionnels. L'utilisateur devait saisir manuellement des valeurs dans des champs de saisie, ce qui compliquait l'expérience, en particulier pour un public non technique. De plus ces filtres ne s'appliquaient pas.

Pour améliorer l'ergonomie et la compréhension de cette section, nous avons introduit des boutons *switch* associés à chaque filtre. Lorsqu'un filtre est activé, une jauge de réglage s'affiche dynamiquement, permettant à l'utilisateur d'ajuster les paramètres de manière visuelle et intuitive (voir Figure 4.9). Ce système de jauges s'est imposé naturellement, car les paramètres optimaux des filtres dépendent fortement des caractéristiques spécifiques de chaque vidéo. Cette approche permet donc à l'utilisateur de procéder par tâtonnement tout en conservant un contrôle simple et direct. De plus, nous avons ajouté une barre de défilement sur le côté afin de bien montrer à l'utilisateur que d'autres filtres sont disponibles.

Par ailleurs, l'ancien bouton *Preview Filters*, qui ne prévisualisait pas réellement les effets des filtres appliqués, a été repensé. Il a été renommé *"Reset Filters"* et réassigné à une fonctionnalité plus claire : la désactivation immédiate de tous les filtres pour repartir

d'une configuration neutre (voir Figure 4.10).

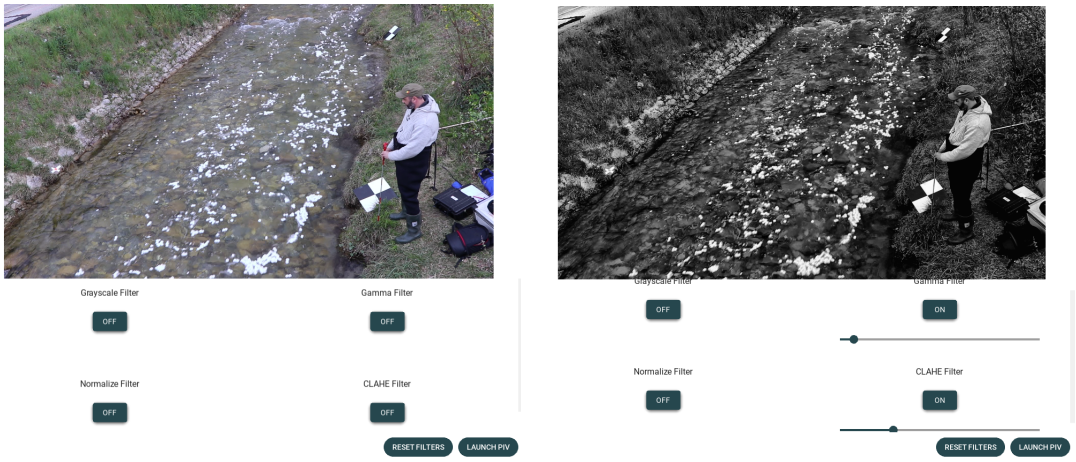


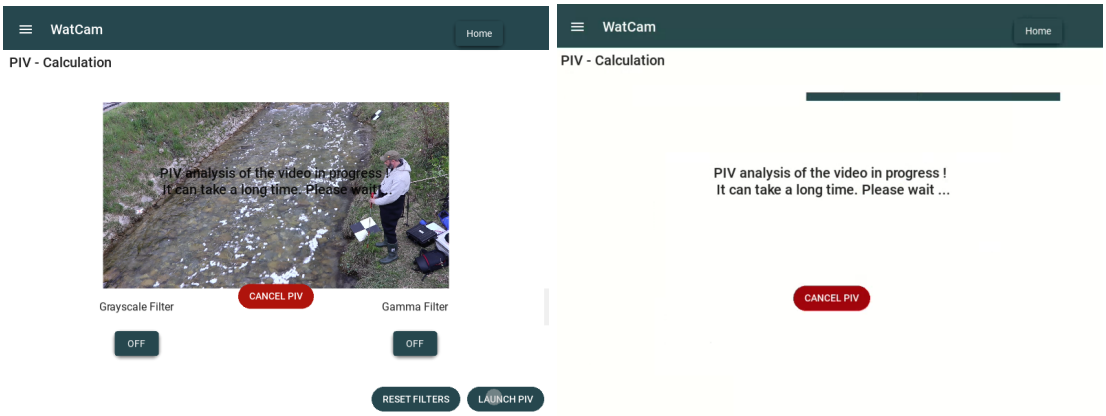

FIGURE 4.9 – Le nouvel écran des filtres avant et après avoir activé des filtres

Juste à côté se trouve le bouton "Launch PIV", destiné à lancer l'analyse PIV. Dans la version initiale, un message s'affichait directement sur l'image de prévisualisation pour indiquer que l'analyse avait débuté. Cependant, selon le contraste ou la couleur de l'image, ce texte était parfois difficilement lisible, voire totalement invisible, ce qui rendait l'interface peu informative. De plus, aucun indicateur de progression n'était présent, et le bouton "Cancel" restait inactif pendant l'exécution, ce qui empêchait l'utilisateur de savoir si le traitement était réellement en cours ou bloqué.

Pour remédier à ces problèmes, nous avons utilisé l'écran PIV qui était mal implémenté afin de le séparer de l'écran des filtres et donc d'éviter la confusion comme à la Figure 4.10a.

De plus, une barre de progression visuelle a été implémentée. Elle informe désormais l'utilisateur de l'état du calcul PIV, offrant ainsi une meilleure transparence sur le traitement et une expérience plus fluide et rassurante.

Le bouton "Cancel" est désormais fonctionnel. Lorsqu'il est activé, il interrompt immédiatement l'analyse PIV en cours. Par défaut, l'utilisateur est ensuite redirigé vers l'écran de `Post Process`, même si le traitement n'a pas été mené à terme. Dans ce cas, si les étapes restantes sont complétées, le système affichera une estimation de débit égale à 0, puisque l'analyse PIV a été annulée. Toutefois, l'utilisateur a la possibilité d'appuyer sur le bouton "Home" pour revenir à l'écran des projets et en démarrer un nouveau, sans avoir à passer par les étapes incomplètes. Cette approche permet de conserver la fluidité du flux de travail tout en offrant une sortie propre et maîtrisée à l'utilisateur.




(a) Ancien (b) Nouveau

FIGURE 4.10 – Ancien et nouvel écran de l'analyse PIV

### 4.5.4 Écran des résultats

Une légère modification a été apportée à l'écran final de l'application, celui qui affiche le résultat du projet. Dans la version initiale que nous avons reçue, le texte affiché était : "*Calculated river flow...*". Cette formulation pouvait prêter à confusion, car elle laissait entendre que le débit était mesuré de manière directe et déterministe.

Nous avons donc choisi de la remplacer par "*Estimated Discharge...*" (voir Figure 4.11), afin de mieux refléter la nature indirecte et probabiliste de la méthode employée. En effet, les résultats produits par WatCam reposent sur des algorithmes de vélocimétrie d'image (*PIV*), qui déduisent le débit à partir de l'analyse de séquences vidéo. Cette estimation dépend de nombreux facteurs environnementaux (lumière, turbulence, visibilité des traceurs, etc.) et repose sur des hypothèses spécifiques. Ainsi, le terme "*estimated discharge*" exprime de manière plus rigoureuse le caractère approché, incertain et conditionnel de la valeur affichée.

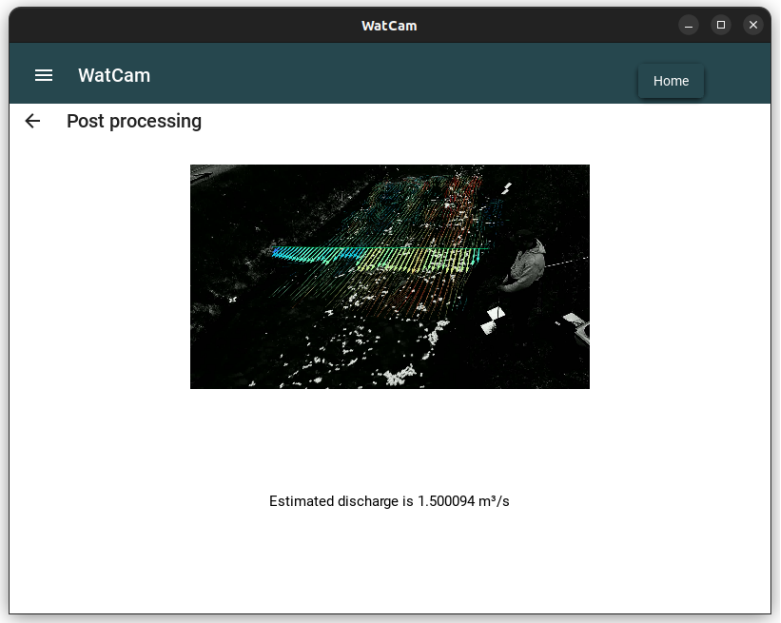



FIGURE 4.11 – Nouvel écran des résultats finaux

### 4.5.5 Vérification des fichiers bathymétriques

Dans les premières versions de l'application, l'import de fichiers bathymétriques ne faisait l'objet d'aucune vérification de validité. Un fichier au format incorrect pouvait provoquer des erreurs silencieuses ou des dysfonctionnements difficiles à diagnostiquer, en particulier pour les utilisateurs non familiers des formats attendus.

Bien qu'un message d'erreur puisse apparaître dans certains cas, il restait trop vague pour permettre à l'utilisateur de comprendre la nature du problème ou la manière de le résoudre.

Une vérification automatique de la structure du fichier a donc été introduite dès le chargement. Si le fichier ne respecte pas le format attendu, une boîte de dialogue explicite informe l'utilisateur (voir Figure 4.12). Elle précise les exigences minimales et fournit un exemple de structure valide.

Les contraintes attendues sont rappelées de façon claire :

— Fichier au format `.txt` ou `.csv`, encodé en UTF-8 ;
— Une ligne par point, avec deux valeurs numériques séparées par une virgule (coordonnées $x$ et $y$) ;
— En-tête facultative.

Ce mécanisme améliore la robustesse de l'application et facilite l'utilisation autonome du logiciel, même en cas de fichier mal formaté.

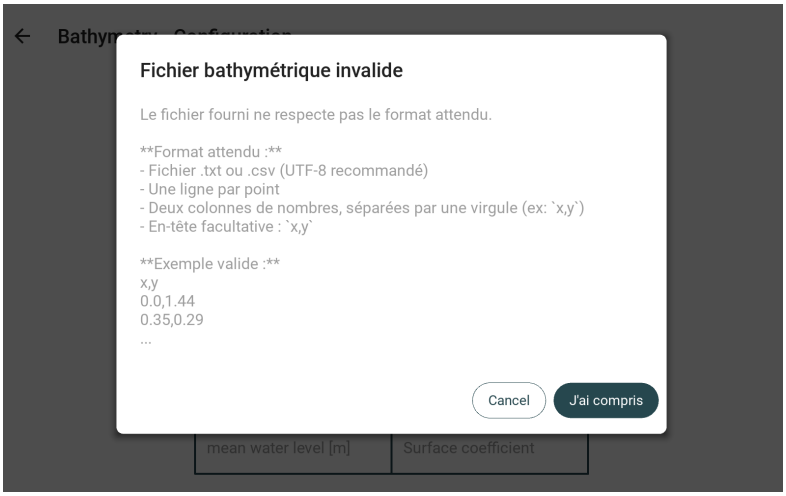



FIGURE 4.12 – Message d'erreur lors de l'import d'un fichier bathymétrique invalide

## 4.6 Application du principe DRY dans le code

Le principe **DRY** (Don't Repeat Yourself) vise à réduire la duplication dans le code. Il facilite la maintenance, limite les erreurs potentielles lors de modifications, et améliore la lisibilité générale du programme.

Dans ce projet, ce principe a été appliqué à plusieurs niveaux. Quelques exemples représentatifs sont détaillés ci-dessous.

### Utilisation d'un dictionnaire pour la gestion des états

La logique conditionnelle liée aux différents états du lecteur vidéo a été remplacée par un dictionnaire associant chaque état à l'action correspondante. Cette refactorisation réduit la complexité du code et simplifie son évolution.

### Formatage du temps centralisé

Plusieurs méthodes nécessitent l'affichage d'une durée au format minutes :secondes. Une méthode dédiée `format_time` a été introduite, garantissant une présentation cohérente et évitant la redondance de code.

#### Centralisation des messages d'erreur

Les messages d'erreur utilisés lors de la validation des données sont désormais regroupés dans une structure unique. Cela améliore leur cohérence, facilite leur modification ou leur traduction, et réduit le code dupliqué.

#### Bénéfices observés

— **Tests unitaires simplifiés :** la factorisation réduit le nombre de cas spécifiques à couvrir ;
— **Réutilisation facilitée :** les méthodes communes peuvent être appelées depuis plusieurs modules ;
— **Meilleure lisibilité :** un code moins redondant est plus simple à comprendre et maintenir.

Les extraits de code correspondants sont disponibles en **Annexe** 8.5.

## 4.7 Création d'un exécutable (.exe) et limites rencontrées avec Kivy

Dans le but de faciliter la diffusion de l'application, une tentative de génération d'un exécutable autonome (.exe) a été réalisée. L'objectif était de permettre son utilisation sans installation préalable de Python.

#### Procédure suivie

L'outil `PyInstaller` a été utilisé via la commande :

```
pyinstaller --name RiverApp --onefile --windowed main.py
```

Cette commande produit un fichier `.spec` personnalisable. Plusieurs ajustements ont été nécessaires pour adapter l'application au format exécutable :

— Ajout manuel des fichiers `.kv` (interface Kivy) ;
— Inclusion explicite de ressources externes (e.g. `proj.db` pour `pyproj`) ;
— Génération manuelle et inclusion explicite d'un fichier `md_icons.py` ;
— Déclaration de modules implicites utilisés par certaines bibliothèques (e.g. `rasterio.sample`) ;
— Suppression des logs Kivy via `KIVY_NO_CONSOLELOG`.

Le fichier `main.py` a également été adapté pour prendre en charge le répertoire temporaire utilisé par PyInstaller (`sys._MEIPASS`).
Un exemple de cette adaptation est fourni en **Annexe** 8.6.1.

### Limites rencontrées

Malgré ces ajustements, l'exécutable généré rencontrait des erreurs au moment d'accéder à certaines ressources internes. Ces dysfonctionnements semblent liés à la manière dont Kivy gère les chemins et fichiers en contexte figé, notamment via `sys._MEIPASS`.

### Retour d'expérience

Bien que l'exécutable n'ait pas pu être finalisé, cette tentative a permis de documenter les principales difficultés de packaging avec Kivy. Ces informations pourront être réutilisées ou complétées pour de futures tentatives.

### Limites structurelles de Kivy dans ce contexte

Kivy, bien que très souple pour créer des interfaces Python modernes, présente plusieurs difficultés majeures :

- Difficultés de packaging, en particulier pour les fichiers de ressources et les fichiers .kv,
- Documentation incomplète pour des cas complexes (utilisation de bibliothèques comme rasterio, pyproj),
- Faible communauté active et outils annexes moins robustes (peu de support officiel pour PyInstaller par exemple),
- Moteur graphique personnalisé qui ne s'intègre pas bien avec les standards du système d'exploitation (accès fichiers, styles natifs, etc.),
- Sensibilité à la gestion des threads, pouvant entraîner des blocages de l'interface lors de traitements longs — un point déjà soulevé dans le mémoire de A.Bombaert[1] comme facteur limitant pour le déploiement mobile.

### Éléments de comparaison pour une migration éventuelle

Une migration vers un autre framework pourrait être envisagée si la stabilité ou la portabilité devenaient des enjeux prioritaires. Le tableau ci-dessous résume quelques options possibles :

| Framework | Avantages | Inconvénients | Idéal pour |
|---|---|---|---|
| **PySide6 (Qt)** | Interface native, excellent support pour les exécutables, documentation riche | Syntaxe plus verbeuse, dépendances plus lourdes à gérer | Application desktop pérenne avec besoins avancés |
| **Tkinter** | Intégré par défaut à Python, très simple à prendre en main | Interface datée, peu flexible graphiquement | Prototypes rapides, petits outils internes |
| **PyWebView + Flask** | Interface web moderne intégrée dans une fenêtre desktop, bonne portabilité | Déploiement web local plus complexe, séparation front/back à maintenir | Applications hybrides ou interfaçables avec du web |
| **DearPyGui** | Performant pour le rendu, approche orientée scientifique, très réactif | Moins standard, plus jeune, moins de support communautaire | Outils d'analyse, data visualisation, interfaces scientifiques |
| **Kivy (actuel)** | Multiplateforme, interface fluide, support tactile, design moderne | Packaging difficile (notamment pour .exe), erreurs peu explicites, support parfois limité | Applications tactiles ou animées, outils portables |

TABLE 4.2 – Comparaison de différents frameworks d'interface graphique en Python

**Perspectives de migration**

Les choix architecturaux récents (architecture MVC, modularité accrue) faciliteraient une éventuelle transition vers un autre framework. La logique métier est suffisamment découplée de l'interface pour permettre une migration progressive, en particulier vers PySide6 ou un autre outil plus robuste pour les déploiements desktop.

## 4.8 Conclusion

Les améliorations apportées à l'interface utilisateur de *WatCam* visent à rendre l'application plus accessible, plus robuste et mieux adaptée aux besoins concrets des utilisateurs, qu'ils soient chercheurs, étudiants ou techniciens de terrain. En corrigeant les principales sources de friction dans l'utilisation du logiciel — gestion des erreurs, navigation dans les projets, paramétrage des filtres — nous avons transformé l'interface initialement rigide en un environnement de travail plus fluide, ergonomique et tolérant aux erreurs.

L'ajout de nouvelles fonctionnalités comme le renommage de projet, le tri dynamique, le bouton *Home*, les jauges de réglage pour les filtres ou encore la clarification de la terminologie finale (*"Estimated Discharge"*) témoigne d'une volonté constante de mettre

l'utilisateur au centre de la conception. Ces évolutions ne sont pas uniquement esthétiques : elles participent directement à la fiabilité du traitement, à la reproductibilité des analyses, et à l'adoption du logiciel par des profils non techniques.

La refonte logicielle basée sur l'architecture MVC a également constitué un tournant important. En introduisant une séparation claire entre l'interface graphique, la logique métier et la gestion des fichiers, elle a permis de rendre le code plus modulaire, plus lisible et plus facile à maintenir. Cette restructuration a favorisé une approche plus rigoureuse du développement, facilitant l'ajout de nouvelles fonctionnalités tout en réduisant les risques de régression. Elle a aussi été l'occasion d'une montée en compétence concrète sur des principes d'architecture logicielle réutilisables dans d'autres projets.

Enfin, un effort particulier a été réalisé pour améliorer la qualité du code en appliquant le principe *Don't Repeat Yourself* (DRY) dans plusieurs modules. Cette démarche a permis de factoriser les traitements récurrents, d'uniformiser les messages d'erreur et de clarifier certaines logiques internes. Outre la lisibilité accrue du code, cela a facilité la mise en place de tests unitaires, encouragé la réutilisation de fonctions génériques et amélioré la collaboration au sein de l'équipe. Ces bonnes pratiques, bien que parfois invisibles pour l'utilisateur final, sont essentielles pour assurer la pérennité et l'évolutivité du projet.

Concernant la création du fichier `.exe`, bien que la compilation ait rencontré des obstacles techniques, notamment liés au packaging des ressources spécifiques et à la gestion des dépendances *Kivy*, cette expérience a permis de documenter des solutions et erreurs fréquentes. Ce retour d'expérience est précieux pour faciliter le travail des futurs contributeurs souhaitant distribuer l'application plus facilement.

Par ailleurs, la phase de développement a largement bénéficié des ressources disponibles sur des plateformes communautaires comme *Stack Overflow* et *GitHub*, qui se sont révélées indispensables pour résoudre de nombreuses erreurs techniques rencontrées tout au long du projet. Ces échanges en ligne, complétés par la consultation régulière de la documentation officielle, ont permis d'adopter rapidement des solutions éprouvées et d'améliorer la qualité du code.

En parallèle, le respect des bonnes pratiques de programmation telles que le principe DRY, le *refactoring* continu et l'écriture de tests unitaires a non seulement amélioré la maintenabilité et la lisibilité du code, mais aussi facilité la collaboration entre nous. Ces efforts méthodiques assurent une base solide pour l'évolution future de l'application.

Ce travail sur l'interface constitue ainsi un levier essentiel pour garantir l'usage de *WatCam* dans des contextes variés, allant de la recherche académique aux campagnes de terrain. Il ouvre également la voie à de futures évolutions, notamment vers des interfaces plus adaptatives, multilingues ou encore accessibles sur d'autres plateformes (tablettes, smartphones).

# Chapitre 5

# Expériences et validations

Dans le cadre de ce mémoire, une série d'expériences a été menée dans des environnements contrastés afin d'évaluer les performances globales de l'application WatCam. L'objectif était de tester la chaîne complète de traitement — de la capture vidéo à l'estimation des débits — dans des contextes expérimentaux aux contraintes variées. Les essais ont été réalisés à la fois en laboratoire, dans des conditions maîtrisées, et en extérieur, en situation réelle. Cette diversité d'environnements permet une évaluation complète de la robustesse, de la précision et des limites de l'outil. L'ensemble des campagnes a été conduit avec l'appui du groupe GCE.

Il est important de rappeler que les résultats qui suivent relèvent davantage du qualitatif que du quantitatif, le débit réel n'étant pas toujours connu pour permettre une comparaison directe. Pour chaque expérience, nous afficherons le `masked_plot` et le `transect_plot`, afin d'observer ce que l'application retourne.

— Le masked_plot est une visualisation des données de vélocimétrie après l'application de masques pour identifier et supprimer les points de données erronés ou non pertinents.
— Le transect_plot est une visualisation qui représente les variations des données de vélocimétrie le long d'une ligne ou d'un transect défini dans l'image.

## 5.1 Campagnes en laboratoire

### 5.1.1 Expérience du *Dike Breaching*

La première expérience, intitulée *Dike Breaching*, a été conduite avec Masoumeh, Arthur et Cyril. Elle visait à observer l'évolution dynamique d'une brèche dans une digue en sable, soumise à un écoulement contrôlé. Le but de notre analyse était d'estimer le débit transitant à travers la brèche à différents instants.

Deux caméras GoPro ont été utilisées : l'une placée en amont, l'autre en aval du dispositif, afin de capturer les phénomènes d'écoulement. Des GCP ont été disposés sur

les parois du canal, orientés vers le haut, comme illustré à la Figure 5.1. Cependant, en raison de la présence du dispositif au centre de l'expérience, certains GCP ne sont parfois pas visibles.

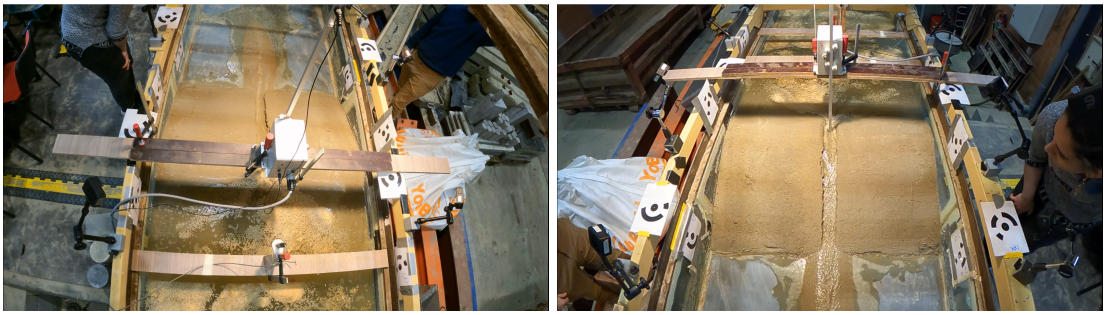


FIGURE 5.1 – Prise de vue en amont et en aval du Dike Breaching

Cette expérience présentait une configuration complexe en raison de la bathymétrie changeante. Nous avons donc supposé que, sur l'intervalle d'analyse d'une seconde, la bathymétrie restait constante. Bien que l'objectif ait été d'estimer le débit traversant la brèche, l'analyse PIV réalisée avec WatCam a rencontré certaines limitations.

Voici le résultat de cette expérience :

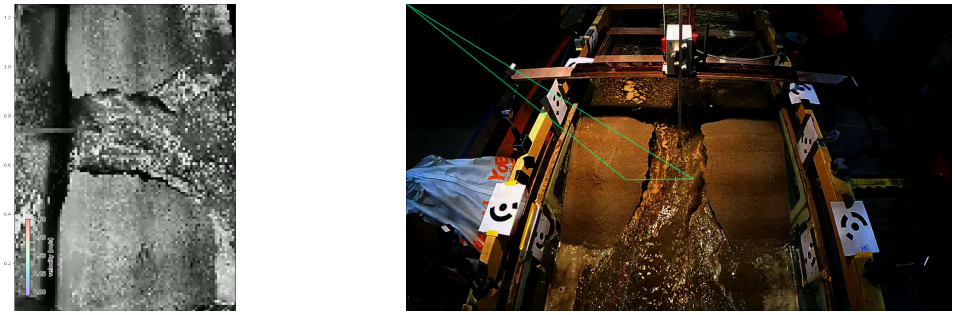


FIGURE 5.2 – Résultat en aval du Dike Breaching

L'application semble avoir du mal avec les parois du canal, ce qui empêche le programme de fonctionner correctement en renvoyant l'erreur suivante. Nous avons plusieurs hypothèses concernant cette erreur, étant donné qu'elle ne survient que lors des expériences en laboratoire. De plus, nous constatons que le transect final prend une forme étrange. Néanmoins, l'application détecte des particules.

— La première hypothèse serait que les parois se trouvent trop haut par rapport au niveau de l'eau, ce qui mènerait à un problème d'orthorectification.
— Une deuxième hypothèse serait une limitation de la bathymétrie : les particules sont détectées, mais WatCam n'arrive pas à effectuer les calculs.

— Enfin, WatCam présente peut-être une limitation pour les expériences de petit gabarit, en raison des approximations sur les petits volumes d'eau, qui seraient arrondis à zéro, ce qui conduirait à cette erreur.

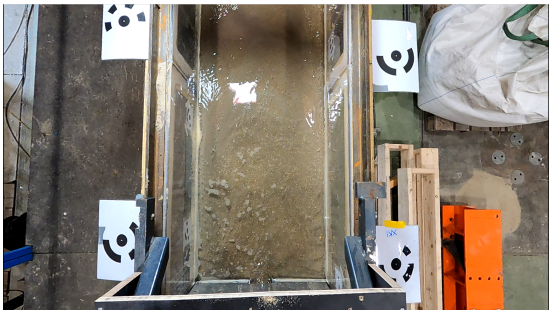
### 5.1.2 Expérience de rupture de barrage

La seconde expérience menée en laboratoire a été réalisée en collaboration avec Erwan et Florian. L'objectif était d'étudier l'impact morphologique sur le lit d'une rivière lors de chasses de barrage.

Deux séries de mesures ont été effectuées pour cette expérience : l'une avec de la frigolite 5.3a et l'autre avec des copeaux de bois (Figure 5.3b). Lors de ces deux sessions, les GCP ont été placés de manière similaire à ceux de l'expérience Dike Breaching (Figure 5.1), le long des parois du canal. À chaque prise de mesure, un spot lumineux a été installé afin d'évaluer l'effet d'un éclairage renforcé sur les résultats de l'expérience.



(a) Expérience avec la frigolite



(b) Expérience avec les copeaux de bois

FIGURE 5.3 – Expérience Erwan et Florian avec la frigolite et les copeaux de bois

#### Expérience avec la frigolite

Lors de la première prise de mesures, nous n'avions pas accès à une GoPro. Nous avons donc dû filmer avec un téléphone. Cette situation s'est révélée intéressante, dans la mesure où l'un des objectifs de WatCam est d'être accessible et facile d'utilisation. Cela représentait donc une bonne opportunité pour évaluer les limites de la qualité d'image.

Cependant, cette prise de vue a rencontré plusieurs difficultés, notamment l'agglomération des traceurs en frigolite, qui a perturbé la dynamique du flux et introduit un biais dans la détection par WatCam. En effet, lorsque nous avons disposé la frigolite en amont,

afin de rendre les particules visibles dans l'application, elle s'est rapidement agglomérée, formant des amas qui ont gêné le mouvement global de l'écoulement. Nous avons donc dû en ajouter en aval, juste après le barrage. Malheureusement, cela a de nouveau entraîné la formation d'agglomérats de frigolite (Figure 5.3a), qui stagnent et induisent en erreur l'application.

Malgré ces difficultés, cette tentative a permis d'obtenir un résultat de 0,00467 $m^3/s$, une valeur qui semble plausible. À l'avenir, il serait pertinent que ce type d'expérience en laboratoire soit pleinement compatible avec WatCam.



FIGURE 5.4 – Résultats de l'expérience avec la frigolite

Au cours de cet essai, nous avons également remarqué que l'application pouvait détecter les mouvements de particules à partir des reflets sur les parois, comme illustré à la Figure 5.5. Il convient donc de veiller à ce que l'application ne prenne pas ces reflets en compte dans l'analyse.



FIGURE 5.5 – Illustration de particules détectées dans le reflet de la paroi

### Expérience avec les copeaux de bois

La seconde prise de mesures a été réalisée à l'aide d'une GoPro fixée à 90° par rapport au canal, positionnée au-dessus du barrage. Les particules utilisées pour cette session étaient des copeaux de bois (Figure 5.3b). L'expérience s'est déroulée sans encombre, et les copeaux n'ont pas perturbé l'écoulement.

Pour cette prise de mesure, WatCam a estimé le débit à 0,006574 $m^3/s$. Ce résultat semble également plausible, bien qu'il représente une augmentation de 33,86 % par rapport au débit estimé avec la frigolite, alors que les conditions expérimentales étaient similaires. Cette différence peut s'expliquer par la présence d'agglomérats de frigolite en suspension, qui ont pu fausser la première estimation, et/ou par une meilleure détection des copeaux de bois par l'application.

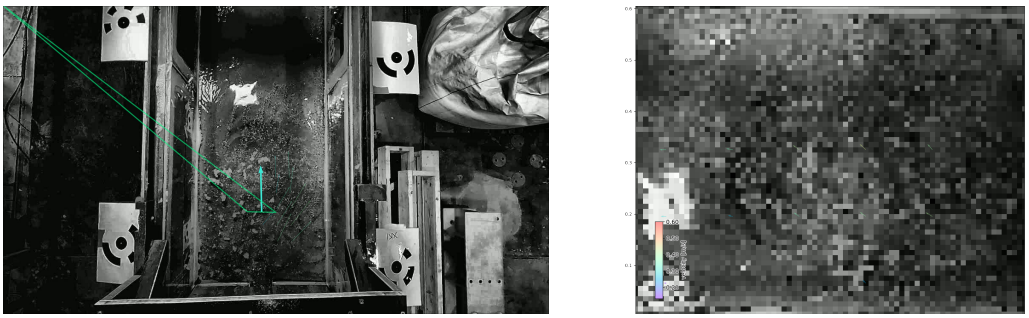


FIGURE 5.6 – Résultats de l'expérience avec les copeaux de bois

## 5.2 Expérience en extérieur

Deux campagnes de terrain ont été réalisées sur le cours d'eau de l'Orneau, à Gembloux, afin d'évaluer le comportement de WatCam dans des conditions environnementales réelles. Le site a été choisi pour sa morphologie représentative et sa facilité d'accès, permettant de varier les angles de vue et les configurations de capture.

Les deux campagnes ont été menées dans des conditions météorologiques similaires, marquées par une alternance entre zones ombragées et zones ensoleillées. Cette configuration a mis en évidence les défis posés par l'hétérogénéité de la luminosité, un facteur ayant une influence notable sur la détection des traceurs en surface.

Pour l'analyse des séquences, une même bathymétrie de référence et une même hauteur d'eau moyenne — fournies par Zoé à partir des mesures collectées lors de la première sortie — ont été utilisées.

**Orneau - Sortie du 17 mars 2025**

La première sortie a eu lieu le 17 mars 2025, en présence de plusieurs membres du GCE. Des GCP en papier biodégradable ont été installés sur les berges, dans un souci de limiter l'impact environnemental au cas où ils tomberaient dans la rivière à cause du vent. Cependant, en raison d'une mauvaise préparation, leur positionnement s'est révélé inadapté : ils étaient placés trop en hauteur par rapport au niveau de l'eau. Par conséquent, l'application a rencontré de grandes difficultés à fournir des résultats exploitables.

Quatre angles de prise de vue ont été exploités : une vue frontale depuis l'intérieur du lit du cours d'eau (Figure 5.7d), une vue latérale perpendiculaire au courant depuis la rive droite (Figure 5.7b), et une vue oblique à 45° dans le même sens (Figure 5.7a) et le sens opposé du courant (Figure 5.7c). Cette diversité de perspectives visait à évaluer l'influence de l'angle de captation sur la précision des mesures, en particulier en termes de distorsion géométrique et de visibilité des traceurs. De plus, Zoé nous a fournis un débit estimé aux alentours de 0,38 $m^3/s$.



(a) 45°



(b) 90°



(c) 45° opposé au courant



(d) Dans le sens du cours d'eau

FIGURE 5.7 – Les différentes prises de vue lors du 17/03

Les résultats pour cette prise de mesure a donné des résultats étonnants. En effet, due aus GCP placsé sur un plan trop haut par rapport au cours d'eau et les prises de vue pas assez en hauteur l'analyse PIV présente des dysfonctionnements. Ces dysfonctionnements s'expriment par une inversion du sens du courant jamais rencontré dans les autres cas.

Voici le résultat obtenu lors de la prise perpendiculaire au courant. On remarque que les traceurs vont bien dans le sens du courant, néanmoins, le résultat du transect indique le débit de 0.1989 $m^3/s$ dans le mauvais sens comme indiqué à la Figure 5.8. La différence de résultat doit probablement venir de la hauteur erronée des GCP.

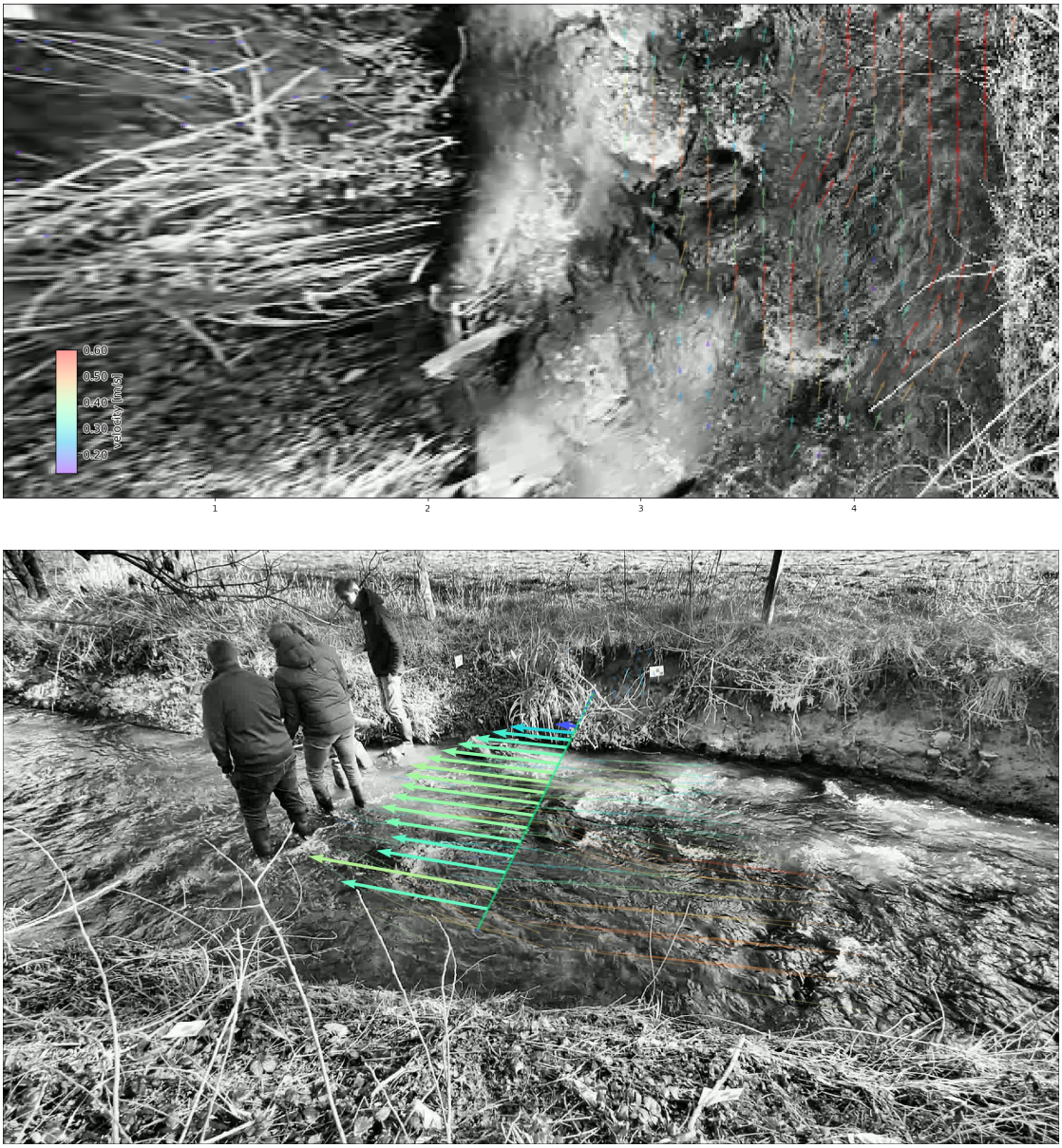



FIGURE 5.8 – Résultats avec les filtres de la prise à 90°

### Orneau - Sortie du 7 avril 2025

La seconde sortie, organisée le 7 avril 2025, a été réalisée avec l'appui d'un drone piloté par Rotchild. Deux vidéos aériennes ont été capturées : l'une selon un angle de plus ou moins 45° (Figure 5.9a), l'autre à la verticale (90°) par rapport au cours d'eau (Figure 5.9b). Cette configuration a permis d'étudier l'impact de la stabilité sur la qualité

des résultats issus de l'analyse PIV.



(a) 45° (b) 90°

FIGURE 5.9 – Les différentes prises de vue par drone le 7/04

Avec un angle de 45° sans filtres, l'application arrive à bien détecter les particules dans le cours d'eau grâce aux différents reliefs présents dans l'eau (pierres,...). L'application calcul un débit estimé à 0.4530 $m^3/s$ (Figure 5.10). Avec la présence des filtres, il semblerait que plus de particules sont détectées et nous obtenons un débit de 0.5336 $m^3/s$ (Figure 5.11).




FIGURE 5.10 – Résultats sans filtres de la prise de vue à 45° avec le drone

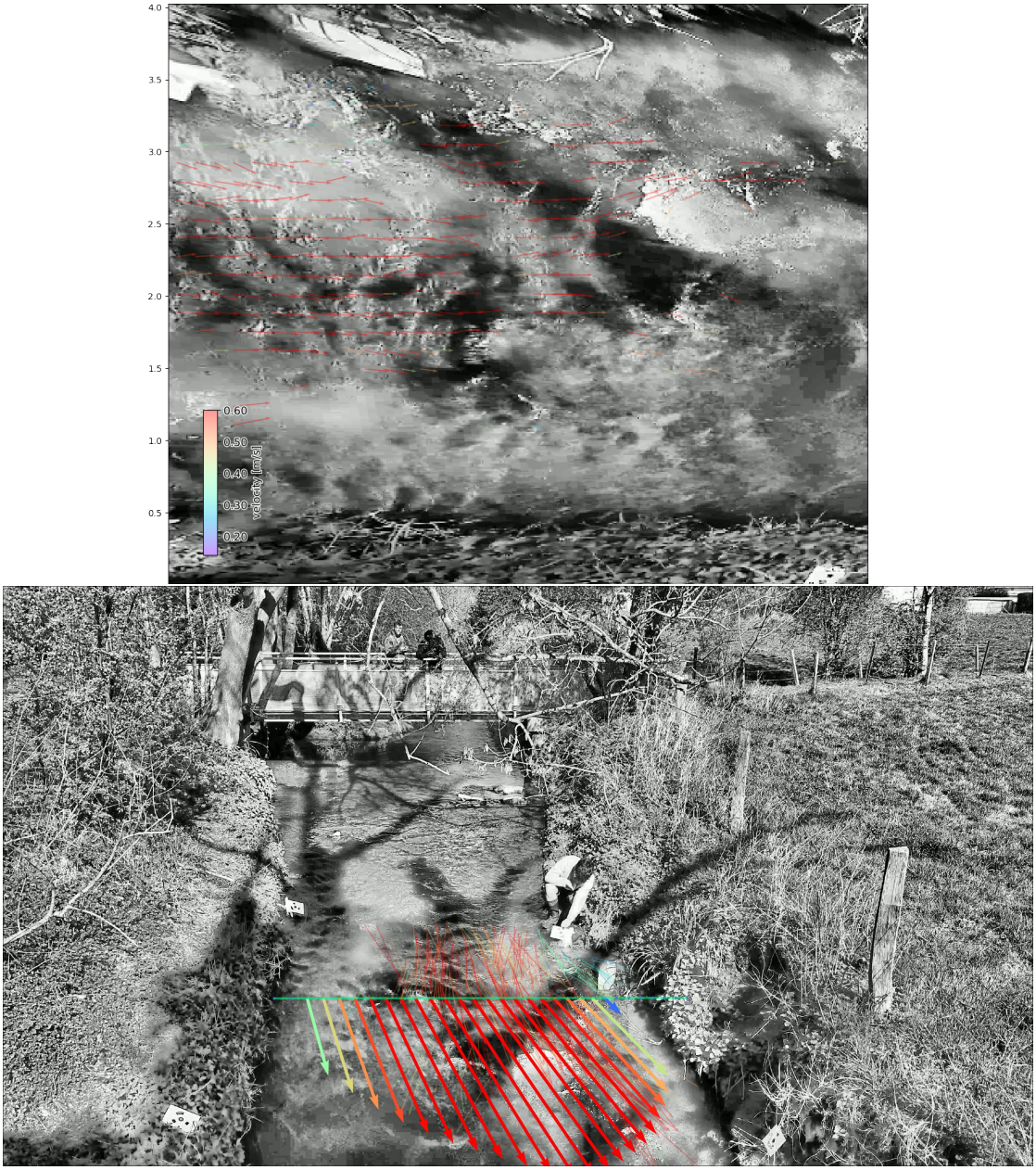



FIGURE 5.11 – Résultats avec filtres la prise de vue à 45° avec le drone

Pour les prises de vue à 90°, la différence se fait moins ressentir. Nous observons un débit de 0.5205 $m^3/s$ sans la présence de filtres (Figure 5.12). Avec la présence des filtres, nous obtenons un débit de 0.4938 $m^3/s$ (Figure 5.13).

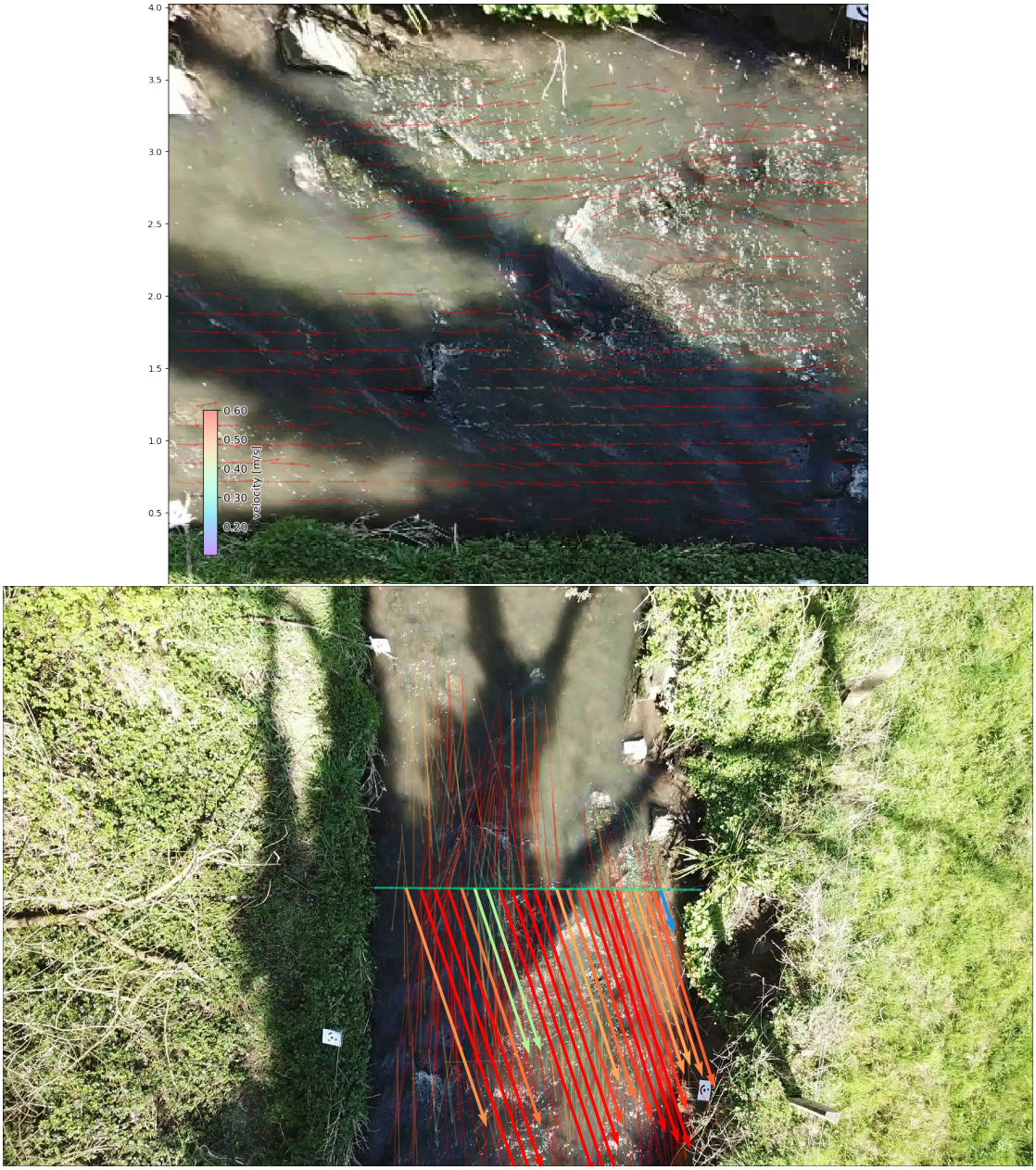



FIGURE 5.12 – Résultats sans filtres en prise de vue à 90° avec le drone

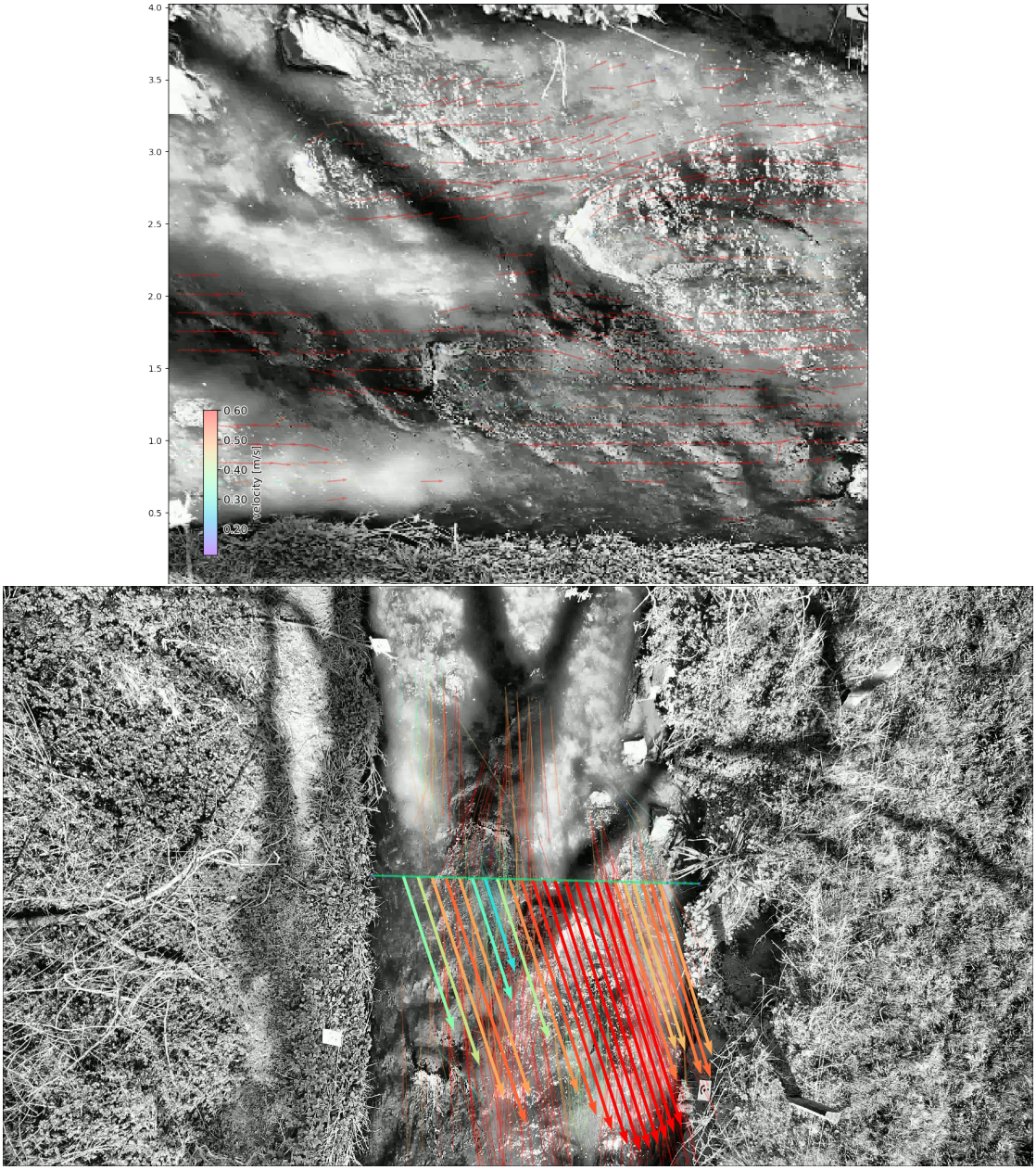



FIGURE 5.13 – Résultats avec filtres en prise de vue à 90° avec le drone

Les résultats des transects apparaissent globalement cohérents. Toutefois, on observe un léger décalage du sens général de l'écoulement vers la droite. Cette anomalie pourrait s'expliquer par l'incertitude liée à la position exacte du drone, celle-ci ayant dû être estimée a posteriori en l'absence de géoréférencement précis.

## 5.3 Bilan

Ces expérimentations, réalisées à la fois en laboratoire et sur le terrain, ont constitué un banc d'essai essentiel pour évaluer la chaîne complète de traitement de WatCam dans des conditions proches des usages réels. Elles ont permis de prendre en compte la variabilité des conditions lumineuses, l'instabilité des supports de captation, ainsi que la présence d'artefacts naturels tels que les reflets, la végétation ou les turbulences.

Nous avons constaté que les expériences en laboratoire restent encore peu exploitées par WatCam. Cette limitation pourrait s'expliquer par la présence des parois du canal. Par exemple, lors de la sortie du 17 mars, les GCP avaient été placés trop haut par rapport au niveau du courant, ce qui a rendu difficile l'obtention d'une estimation fiable du débit. Une autre hypothèse est que les débits mesurés en laboratoire soient trop faibles pour être correctement calculés par l'application.

Pour pallier ce problème, une solution envisageable serait d'utiliser un système de ventouses permettant de fixer de petits GCP directement sur les parois lors des expériences.

Concernant les sorties en extérieur, le site de l'Orneau présentait un avantage notable : un lit très irrégulier, générant des particules plus facilement détectables par WatCam. Dans ce contexte, les filtres appliqués ont perdu une partie de leur utilité. Toutefois, nous avons pu observer qu'ils restent pertinents pour traiter des images où les particules sont peu visibles à l'ombre, voire pour révéler des particules initialement masquées. Enfin, la seconde sortie a permis d'approcher le débit attendu, estimé à 0,38 $m^3/s$.

# Chapitre 6

# Perspectives d'amélioration et développements futurs

## 6.1 Filtres pour l'analyse PIV

L'application de filtres d'image constitue une étape essentielle dans le prétraitement des vidéos utilisées en Vélocimétrie d'Image de Particules (PIV). Ces filtres permettent de réduire le bruit, d'améliorer le contraste et de mettre en évidence les particules à suivre, éléments cruciaux pour garantir une mesure de vitesse précise. Dans le cadre de WatCam, où les conditions de captation peuvent être très variables (lumière, turbidité, présence de reflets), le choix et la combinaison de filtres adaptés influencent directement la qualité de l'estimation du débit. C'est pourquoi il est important pour le futur de WatCam d'envisager d'autres filtres à implémenter.

### 6.1.1 Vue d'ensemble des filtres

Voici les principaux filtres envisagés pour l'analyse PIV, ainsi que leurs effets et applications :

#### Flou gaussien

- **Effet** : convolution avec une fonction gaussienne, ce qui a pour effet de lisser l'image. Il réduit les variations à haute fréquence (bruit), tout en conservant la structure globale des motifs. Cela améliore la stabilité de la corrélation dans les algorithmes PIV, en supprimant les artefacts numériques ou optiques parasites. [19].
- **Application en PIV** :
  - Idéal pour éliminer le bruit de petite échelle causé par la turbulence de l'eau ou les artefacts vidéo.
  - Préserve les formes des particules pour une meilleure détection lors des étapes suivantes.

**Flou médian**

- **Effet** : remplace chaque pixel par la médiane de ses voisins. Il est extrêmement efficace contre le bruit impulsionnel (« sel et poivre ») et préserve les bords, ce qui est crucial en PIV où la forme des particules compte.
- **Application en PIV** :
  - Utile dans les scénarios contenant du bruit de type sel et poivre ou des artefacts pixelaires.
  - Préserve les bords plus efficacement que le flou gaussien, assurant ainsi la netteté des contours des particules.

**Filtre de Sobel**

- **Effet** : opérateur de détection de bord qui calcule l'intensité du gradient horizontal ou vertical d'une image. Il met en évidence les zones de transition rapide en luminosité, ce qui correspond généralement aux contours des objets. En PIV, il aide à faire ressortir les bords des traceurs (bulles, feuilles, motifs turbulents), facilitant la corrélation locale dans les fenêtres d'interrogation. [19, 11].
- **Application en PIV** :
  - Améliore la visibilité des contours des particules, facilitant le processus de segmentation.
  - Utile pour détecter les motifs directionnels des écoulements dans le fluide.

**Filtre laplacien**

- **Effet** :filtre Laplacien applique un opérateur du second ordre (dérivée seconde) qui détecte les zones de variation rapide de luminance, y compris dans plusieurs directions. Il est souvent utilisé pour rehausser les détails fins ou les petits motifs dans l'image. En PIV, il peut être utilisé pour accentuer les petites particules ou fluctuations texturales en surface — mais doit souvent être précédé d'un flou pour limiter le bruit. De plus, ce filtre est utilisé dans MatLab [34].
- **Application en PIV** :
  - Utile pour améliorer les petites particules ou les détails fins du champ d'écoulement.
  - Souvent combiné à un pré-lissage pour détecter des structures fines sans bruit amplifié

**Détection des contours de Canny**

- **Effet** : méthode sophistiquée de détection de contours basée sur plusieurs étapes : lissage (souvent gaussien), calcul du gradient, suppression des non-maxima et seuillage double. Il fournit des contours fins et nets. En PIV, cette méthode est

utile pour identifier précisément les bords des motifs de surface dans les cas de faible contraste ou d'arrière-plans complexes [35].

— **Application en PIV** :
  — Fournit des contours clairs et précis pour la segmentation des particules.
  — Particulièrement efficace dans des environnements bruyants ou lorsque les contours des particules sont peu visibles.

La Figure 6.1 représente les différents filtres proposés ci-dessus appliqués individuellement sur une image de base tirée de l'exemple VGC.

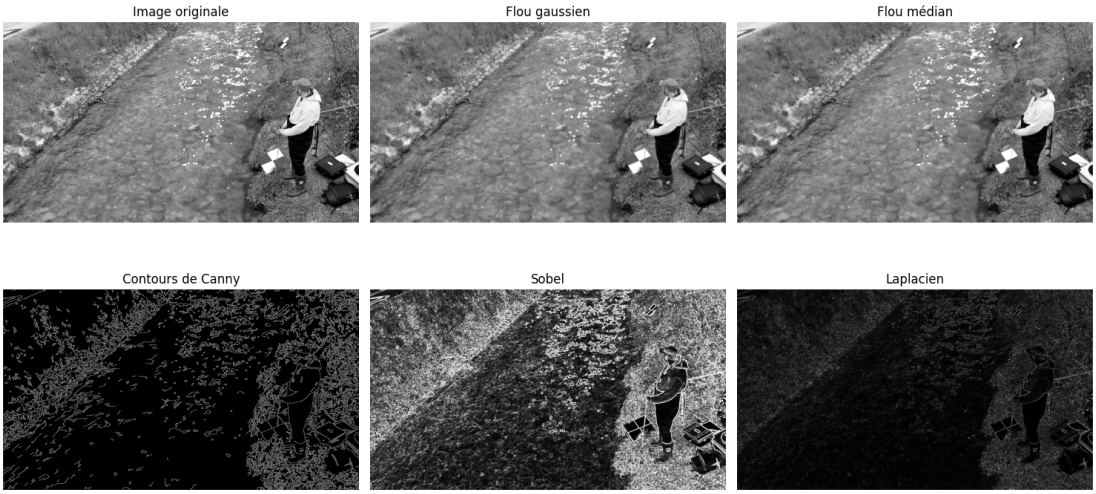



FIGURE 6.1 – Illustration des filtres sur une image de base (VGC)

### 6.1.2 Optimisation des combinaisons de filtres

Bien que chaque filtre apporte des avantages spécifiques, leur application combinée permet souvent d'obtenir des résultats supérieurs. Par exemple :

— Un **flou gaussien** suivi d'une **détection des contours de Canny** peut réduire le bruit tout en extrayant simultanément des contours de particules précis.
— La séquence **flou médian** et **filtre de Sobel** garantit que les bords sont préservés tout en éliminant les artefacts de bruit isolés.

La détermination de la séquence optimale de filtres dépend de facteurs tels que la clarté de l'eau, la résolution vidéo et les conditions d'éclairage. Des tests et des analyses de sensibilité sont nécessaires pour identifier les combinaisons maximisant la précision de l'analyse PIV dans des scénarios spécifiques.

### 6.1.3 Défis et considérations

Bien que les filtres améliorent la qualité des images, leur application nécessite des ajustements précis :

— **Sur-lissage** : Un flou excessif peut entraîner une perte de détails des particules, réduisant la précision de la détection [19].
— **Amplification du bruit** : Les filtres de détection des bords comme le Laplacien peuvent amplifier le bruit haute fréquence si l'image n'est pas pré-lissée.
— **Compromis de performance** : Le coût computationnel de l'application de plusieurs filtres doit être équilibré avec les exigences de traitement en temps réel de l'application WatCam.

## 6.2 Autres travaux futurs

Plusieurs pistes concrètes peuvent encore être explorées pour faire évoluer WatCam.

D'abord, sur le plan technique, l'interface graphique souffre encore de limitations, notamment lorsqu'on redimensionne la fenêtre ou qu'on change d'écran. Ces défauts d'affichage nuisent à l'ergonomie, en particulier sur les écrans de grande taille ou à faible résolution. Une refonte de la gestion des layouts dans Kivy, voire une prise en charge adaptative des différentes tailles d'écran (ordinateur, tablette, mobile), s'impose. Par ailleurs, il serait pertinent d'envisager, à moyen terme, l'adoption d'un autre framework graphique plus robuste et mieux supporté que Kivy, afin d'améliorer la stabilité et la portabilité de l'interface. Cela permettrait aussi, à plus long terme, d'envisager un portage vers le mobile — un atout considérable pour les campagnes de terrain.

Ensuite, du côté de l'exécution et des performances, il reste important de mieux gérer les opérations longues comme le calcul PIV ou l'analyse des transects. Actuellement, ces traitements bloquent parfois l'interface ou donnent l'impression d'un crash. Une refonte de la logique de gestion des threads ou l'intégration d'une file d'événements permettrait de conserver une interface réactive en toute circonstance.

Du côté du fichier exécutable, bien que sa génération ait été amorcée, elle reste inaboutie. Finaliser cette étape permettrait de tester le logiciel dans un contexte "utilisateur final", sans environnement de développement, et ainsi de recueillir des retours concrets pour améliorer encore sa robustesse.

Enfin, une perspective plus ambitieuse serait d'intégrer un système de filtrage adaptatif reposant sur l'analyse en temps réel des caractéristiques visuelles des vidéos, telles que la luminosité, le contraste ou le niveau de bruit. En combinant ce filtrage avec des algorithmes de deep learning — comme ceux proposés par Hu et al. Hu et al. [36] — il serait possible d'automatiser la sélection et le réglage des filtres de prétraitement. Cette

approche permettrait d'ajuster dynamiquement les traitements appliqués pour optimiser la qualité des images analysées, sans intervention manuelle, et ainsi renforcer la robustesse et la fiabilité de l'analyse dans des conditions variées.

En résumé, WatCam dispose désormais d'une base fonctionnelle plus solide, mais plusieurs verrous techniques et ergonomiques doivent encore être levés.

# Chapitre 7

# Conclusion

Ce mémoire a présenté le développement, l'évaluation et la mise en œuvre de *WatCam*, une application modulaire, libre et accessible destinée à l'estimation non-intrusive du débit fluvial à partir de séquences vidéo.

Parmi les contributions majeures de ce travail, l'implémentation d'une interface de prévisualisation des filtres constitue une avancée importante. Cette fonctionnalité offre à l'utilisateur la possibilité d'adapter dynamiquement le traitement d'image en fonction des conditions de captation (turbidité, contraste, reflets), améliorant potentiellement la précision des analyses PIV.

Par ailleurs, une attention particulière a été portée à la refactorisation du code, indispensable pour garantir la robustesse, la maintenabilité et l'évolutivité du projet. La réorganisation des fichiers, la clarification des responsabilités entre front-end et back-end, ainsi que l'ajout de protections (assertions, exceptions) améliorent significativement la structure du programme. Ces ajustements facilitent le développement de nouvelles fonctionnalités et le déploiement futur de l'application, notamment sur des plateformes mobiles.

Les expérimentations, conduites en laboratoire comme en conditions réelles, ont montré la robustesse de *WatCam* dans des contextes variés, tout en mettant en évidence certaines limites inhérentes aux méthodes optiques : sensibilité aux reflets, à la turbulence, à la qualité de l'image ou à l'absence de traceurs visibles. Malgré ces défis, les résultats obtenus sont encourageants, notamment dans les configurations bien calibrées (vue stable, bathymétrie fiable, éclairage homogène), et les perspectives d'amélioration sont nombreuses.

Parmi ces perspectives, on peut citer l'intégration de méthodes hybrides combinant PIV et STIV, le couplage avec des données bathymétriques acquises automatiquement (par drone ou lidar), ou encore le recours à l'apprentissage automatique pour améliorer la détection des particules et la segmentation des zones d'analyse. De plus, l'ouverture

du code en open-source permet de favoriser une adoption plus large dans la communauté scientifique et opérationnelle.

Au-delà de l'aspect purement technique, ce mémoire s'inscrit également dans une démarche de science ouverte : reproductibilité des résultats, accessibilité de l'outil, et respect des enjeux environnementaux dans les campagnes expérimentales (utilisation de traceurs biodégradables, captation par drone, etc.). *WatCam* se positionne ainsi comme une solution innovante au croisement de l'ingénierie hydraulique, de la vision par ordinateur et de l'informatique scientifique, avec un fort potentiel d'impact pour la recherche appliquée et les acteurs du territoire.

# Chapitre 8

# Annexe

## 8.1 Techniques de vélocimétrie optique

TABLE 8.1 – Comparaison des principales techniques de vélocimétrie optique

| Méthode | Erreur typique | Avantages | Inconvénients |
|---|---|---|---|
| **PIV** | 2–5% | — Haute précision en laboratoire<br>— Permet l'analyse fine des écoulements turbulents | — Nécessite un environnement très contrôlé |
| **LSPIV** | 5–20% | — Adapté aux conditions réelles (crues, terrain difficile)<br>— Utilisable avec drones ou caméras fixes<br>— Adapté à la vision de surface | — Dépend fortement de la présence de traceurs en surface<br>— Sensible aux reflets, ombres, variations de lumière |
| **PTV** | 5–15% | — Suit précisément les trajectoires individuelles<br>— Bon pour les flux à faible densité de particules | — Moins robuste si trop de traceurs (chevauchement de trajectoires)<br>— Nécessite une bonne résolution temporelle |
| **STIV** | 10–15% | — Efficace avec peu de traceurs visibles<br>— Moins dépendant des artefacts visuels<br>— Robuste en rivière canalisée | — Limité aux profils rectilignes<br>— Moins performant pour les écoulements complexes ou multidirectionnels |

## 8.2 Solutions existantes

TABLE 8.2 – Comparaison synthétique de WatCam avec des solutions existantes

| **Critères** | **WatCam** | **Fudaa-Mascaret** | **PIVLab** | **OpenPIV** | **RIVeR** |
|---|---|---|---|---|---|
| Interface utilisateur | Graphique (Kivy) | Graphique (Java) | MATLAB GUI | Ligne de commande / script | Basique |
| Licence | Open source (MIT) | Open source (GPL) | Propriétaire (MATLAB) | Open source (MIT) | Libre académique |
| Analyse PIV | Oui | Non (modélisation hydraulique) | Oui (avancée) | Oui | Oui |
| Gestion projet | Structurée et persistée | Basique | Projet unique | Aucune | Moyenne |
| Temps de calcul | Modéré (threading) | Élevé (calculs 2D) | Rapide (optimisé MATLAB) | Variable | Variable |
| Déploiement | Python standalone | Java | MATLAB requis | Python pur | Windows / académique |
| Visualisation | Matplotlib intégrée | Graphique vectoriel | MATLAB plots | Custom Python | Variable |

## 8.3 Effet des filtres

TABLE 8.3 – Comparaison fonctionnelle des filtres vidéo dans WatCam

| **Filtre** | **Utilité principale** | **À éviter si...** |
|---|---|---|
| Normalize | Homogénéisation des niveaux de gris | L'image est déjà correctement exposée |
| CLAHE | Rehaussement local du contraste | La vidéo contient du bruit visuel important |
| Gamma | Correction d'exposition globale | Le facteur gamma est mal ajusté |
| Noir et blanc | Standardisation pour algorithme PIV | Les couleurs contiennent de l'information utile |

## 8.4 Analyse de sensibilité des différents paramètres

### 8.4.1 Durée de la vidéo

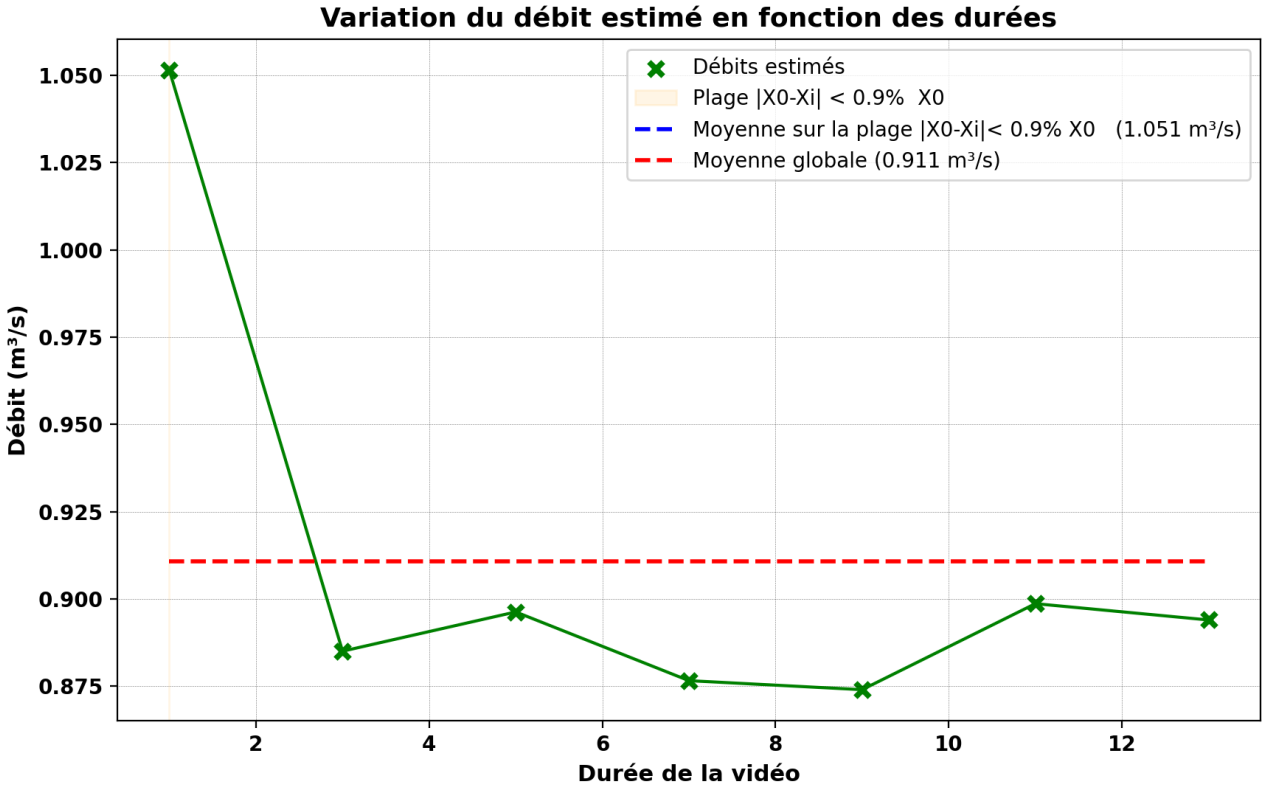



FIGURE 8.1 – Analyse de la durée de la vidéo

### 8.4.2 FPS

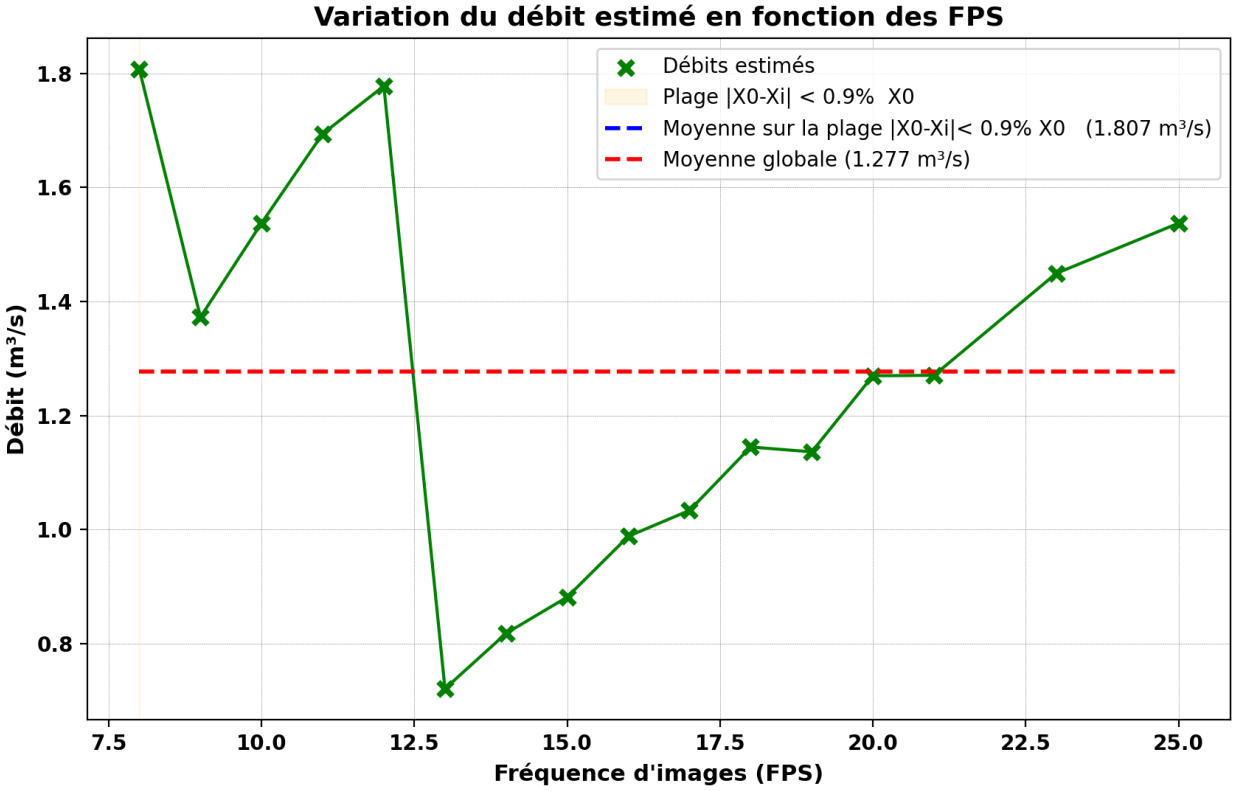



FIGURE 8.2 – Analyse des FPS

### 8.4.3 Filtres

#### CLAHE

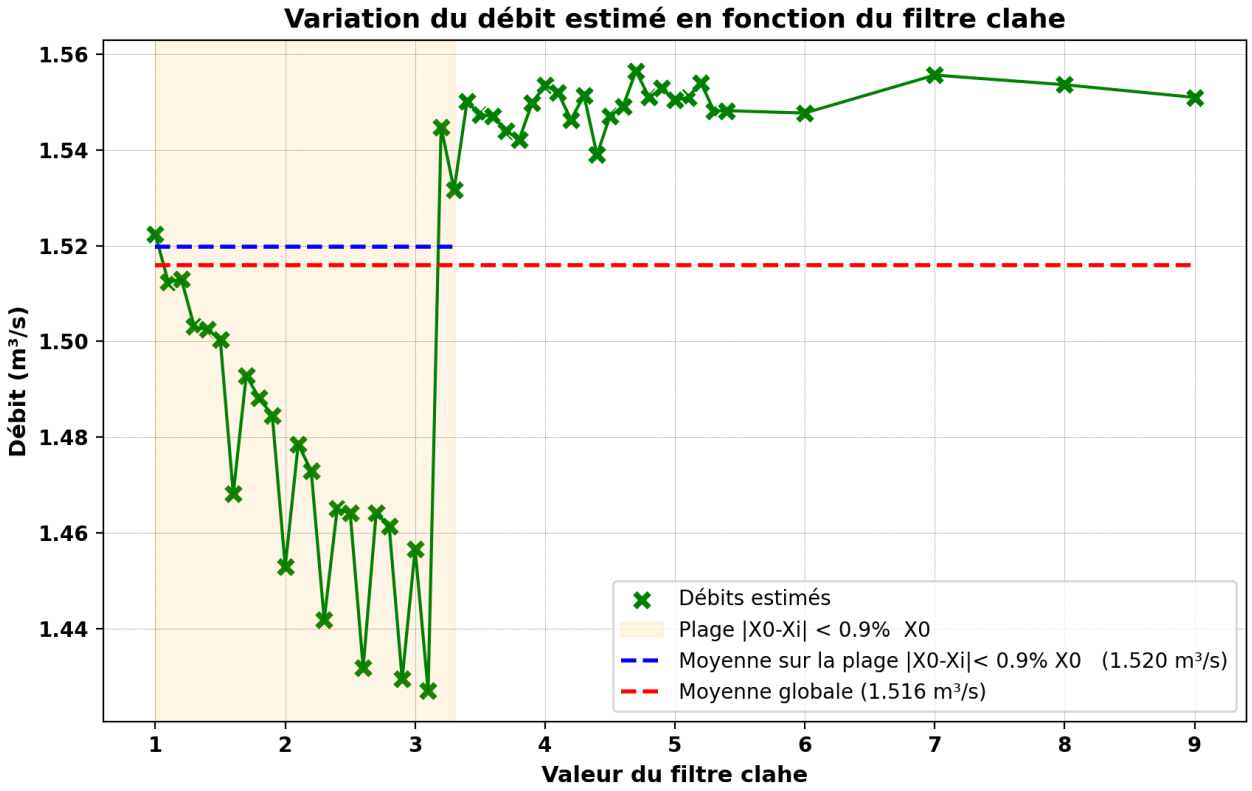



FIGURE 8.3 – Analyse du filtre CLAHE

**Gamma**

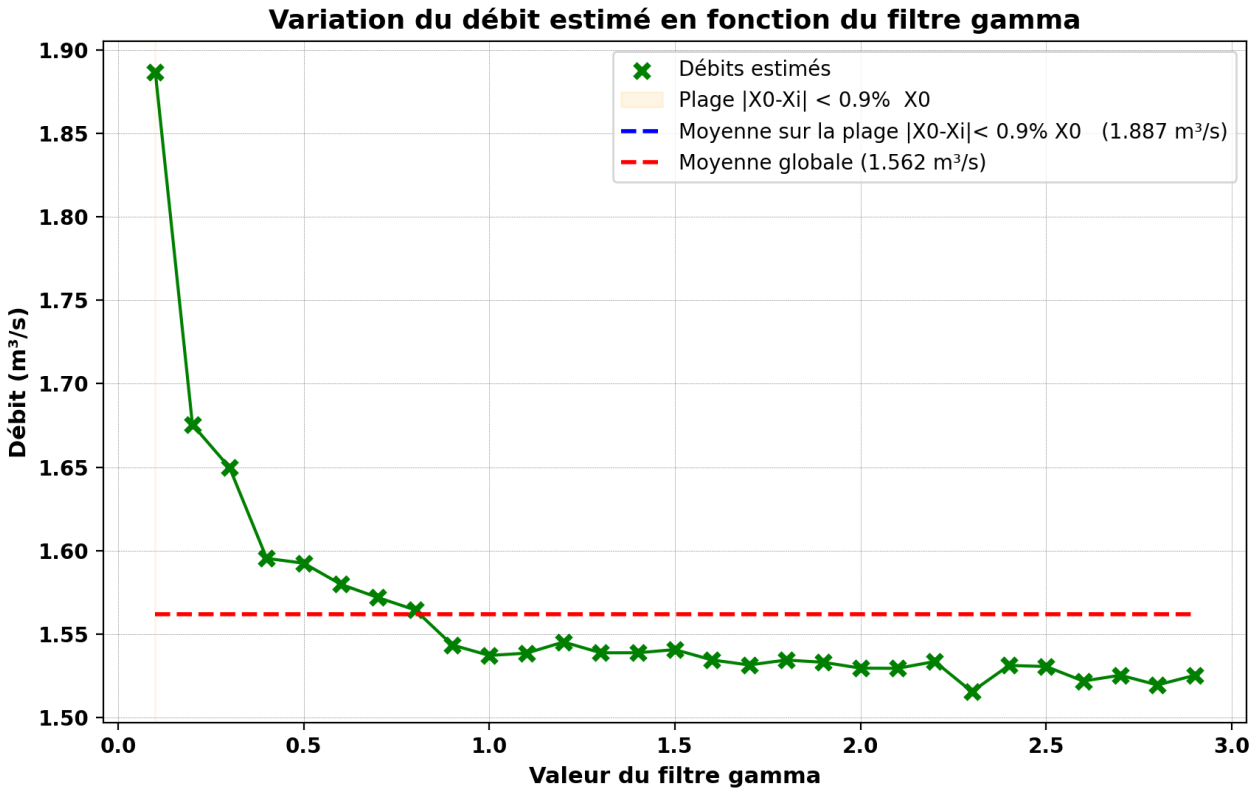



FIGURE 8.4 – Analyse du filtre gamma

### Grayscale

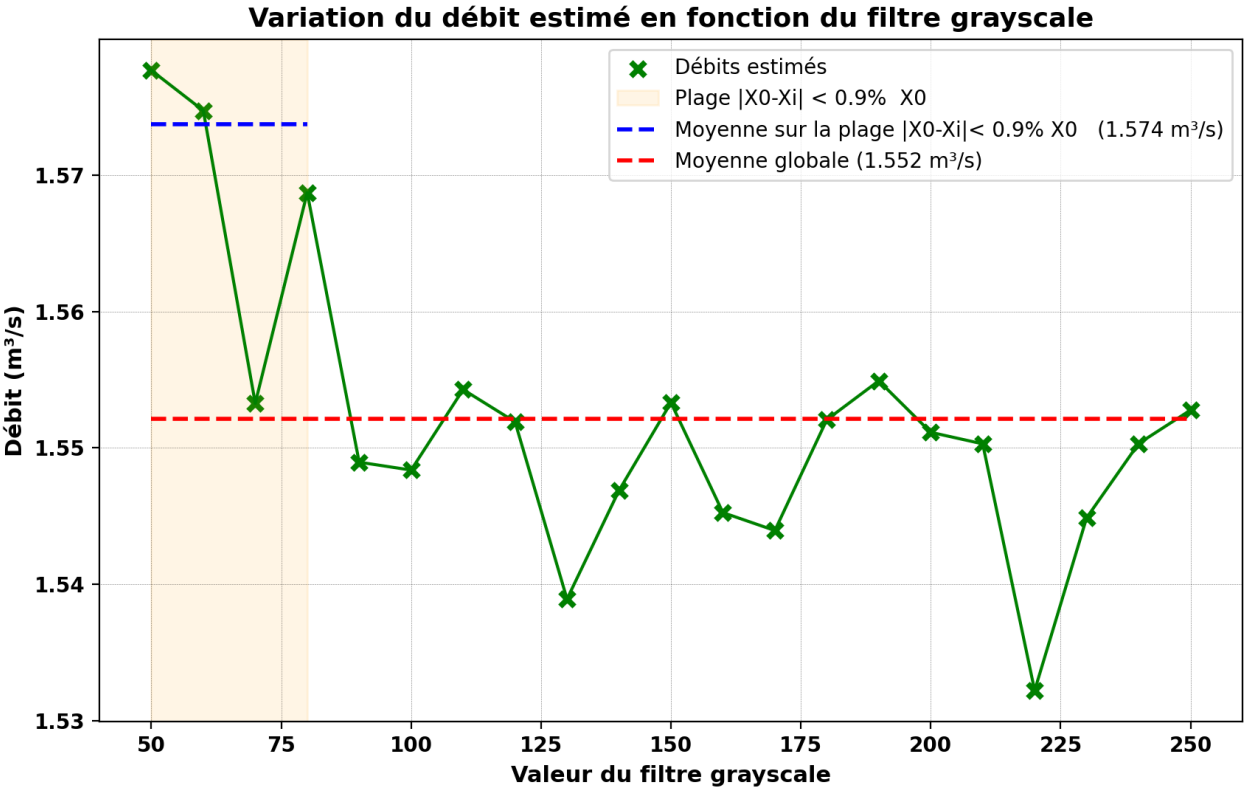



FIGURE 8.5 – Analyse du filtre grayscale

**Normalize**

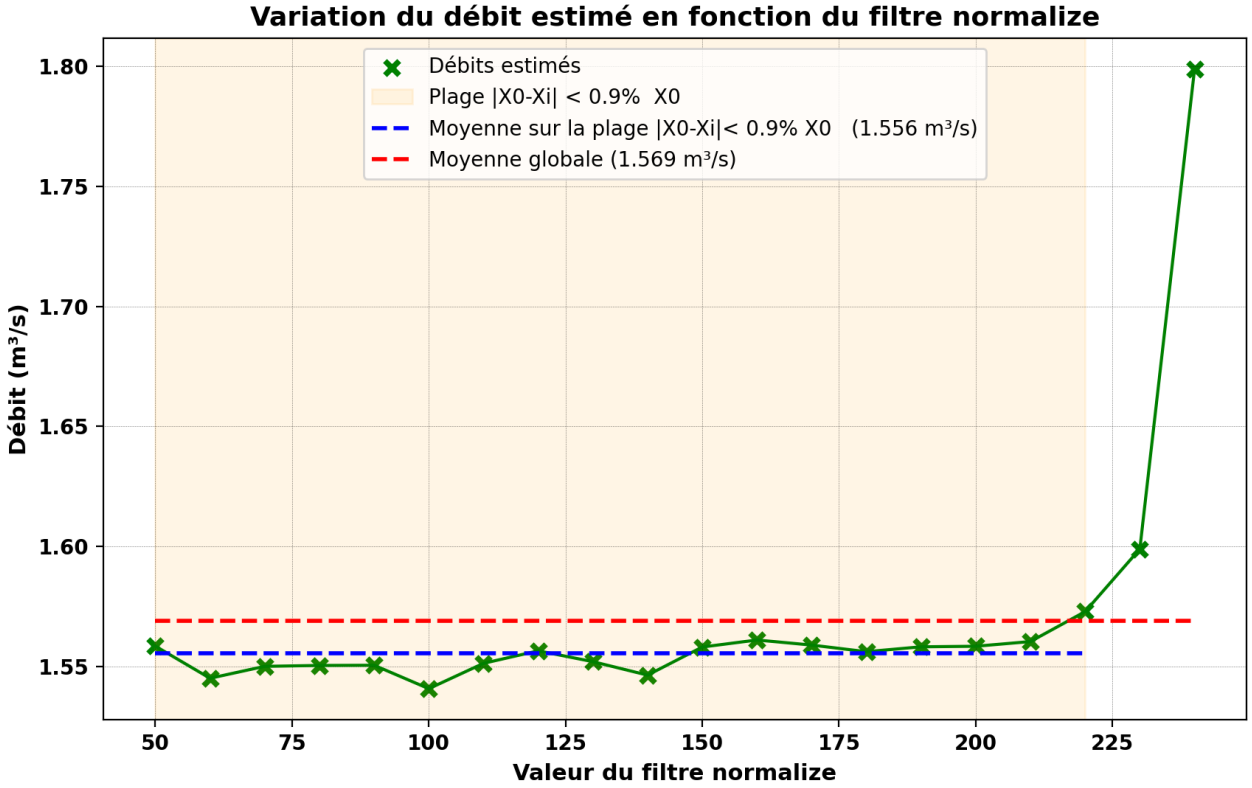



FIGURE 8.6 – Analyse du filtre normalize

## 8.5 Extraits de code

### 8.5.1 Gestion simplifiée des états dans le lecteur vidéo

```
def on_state(self, instance, value: str) -> None:
    state_map = {"play": "play", "pause": "pause", "stop": "stop"}
    if value not in state_map:
        raise ValueError(f"Unknown state: {value}")
    self.ids.video.state = state_map[value]
```

### 8.5.2 Factorisation de la mise en forme du temps

```
def format_time(self, seconds: float) -> str:
    minutes = int(seconds // 60)
    seconds = int(seconds % 60)
    return f"{minutes}:{str(seconds).zfill(2)}"

def set__time_duration(self) -> None:
    self._time_duration = self.format_time(self._duration)

def set__current_time(self, value: float) -> None:
    self._current_time = self.format_time(value)
```

### 8.5.3 Centralisation des messages d'erreur

```
ERROR_MESSAGES = {
    "start_time": {
        "required": "Start Time is required",
        "invalid": "Start Time should be a number",
        "negative": "Start Time cannot be less than 0",
        "greater_than_end": "Start Time cannot be greater than End
            Time",
    },
    "end_time": {
        "required": "End Time is required",
        "invalid": "End Time should be a number",
        "negative": "End Time cannot be less than 0",
        "smaller_than_start": "End Time cannot be smaller than
           Start Time",
    },
    # autres messages...
}
```

## 8.6 Annexes techniques

### 8.6.1 Code pour la génération de l'exécutable Windows

#### Fichier `.spec` modifié

Le fichier `.spec` généré par `pyinstaller` doit être adapté pour inclure certaines ressources utilisées par l'application (notamment Kivy, PyProj, KivyMD, etc.). Ci-dessous, un extrait du fichier modifié :

```
# -*- mode: python ; coding: utf-8 -*-
import os

# Chemins generiques vers les ressources necessaires
proj_path = r"C:\Path\To\Project\venv\lib\site-packages\pyproj\
    proj_dir\share\proj"
md_icons_path = r"C:\Path\To\Project\venv\lib\site-packages\kivymd
    \icon_definitions\md_icons.py"
base_path = r"C:\Path\To\Project"

datas = [
    (proj_path, "proj"),
    (md_icons_path, "kivymd/icon_definitions"),
    (os.path.join(base_path, "src/front/riverapp_controller/
        riverapp_controller.kv"), "riverapp_controller.kv")
]

a = Analysis(
    ['main.py'],
    pathex=[base_path],
    binaries=[],
    datas=datas,
    hiddenimports=[
        'rasterio.sample', 'rasterio._io', 'rasterio._base',
        'rasterio._features', 'rasterio.vrt'
    ],
    ...
)
```

Listing 8.1 – Extrait du fichier .spec modifié

**Remarque :** Pour relancer PyInstaller en utilisant la configuration personnalisée, il suffit d'exécuter la commande `pyinstaller RiverApp.spec` dans le terminal. Le fichier complet `.spec` modifié est disponible dans le dépôt GitHub du projet pour référence.

### Modification du fichier `main.py`

Des modifications doivent être apportées au début du fichier `main.py` afin de gérer correctement les chemins vers les ressources externes lors de l'exécution dans un exécutable compilé :

```
import os
import sys
from kivy.resources import resource_add_path

# Supprimer les logs console Kivy
os.environ["KIVY_NO_CONSOLELOG"] = "1"

# Corriger le chemin vers les ressources PROJ si l'ex cutable est "gel "
if getattr(sys, 'frozen', False):
    os.environ["PROJ_LIB"] = os.path.join(sys._MEIPASS, "proj")
    resource_add_path(sys._MEIPASS)
```

Listing 8.2 – Préparation de main.py pour l'exécutable

**Note importante :** L'environnement `sys._MEIPASS` est une zone temporaire dans laquelle PyInstaller extrait les fichiers nécessaires lors de l'exécution. Il peut poser problème si des ressources sont mal référencées ou si l'accès y est mal géré, en particulier avec Kivy.

# Bibliographie


[1] A. Bombaert. “RiverApp : Application de mesure non-intrusive du débit des rivières basée sur l’analyse PIV d’une vidéo de l’écoulement”. Mémoire de master. Louvain-la-Neuve, Belgique : Université catholique de Louvain, 2024. URL : https://thesis.dial.uclouvain.be/entities/masterthesis/0097c3fd-8bf6-4413-b001-bf1aa11a5bb1.

[2] Grégoire Debray et D. Hay. “Mesure de débit en rivière : analyse et optimisation du projet RiverApp”. Mémoire de master. Louvain-la-Neuve, Belgique : Université catholique de Louvain, 2023. URL : https://thesis.dial.uclouvain.be/entities/masterthesis/e90736b9-5bc6-4387-9ba4-2df68ad64862.

[3] Teledyne RD Instruments. “Acoustic Doppler Current Profilers (ADCPs) : Principles of Operation”. In : *Teledyne Technical Note* (2007). URL : https://www.teledynemarine.com/en-us/support/SiteAssets/RDI/Manuals%20and%20Guides/General%20Interest/BBPRIME.pdf.

[4] Robin Pike, Todd Redding et Carl Schwarz. “Development and Testing of a Modified Transparent Velocity-Head Rod for Stream Discharge Measurements”. In : *Canadian Water Resources Journal* (fév. 2016). DOI : 10.1080/07011784.2015.1127776.

[5] Alexandre Mignot. “Riverapp : application de mesure non-intrusive du débit des rivières basée sur l’analyse PIV d’une vidéo de l’écoulement”. Mém. de mast. Université catholique de Louvain, 2024. URL : https://hdl.handle.net/2078.2/37456.

[6] J. Le Coz. *Règles à jauger - River Hydraulics - INRAE RiverLY*. URL : https://riverhydraulics.riverly.inrae.fr/outils/instrumentation/regles-a-jauger.

[7] Fmourau. *Jaugeage du débit d’un cours d’eau par dilution de chlorure de sodium*. URL : https://www.pedagogie.ac-nice.fr/svt/?p=1744.

[8] Audrey. *Mission SWOT de la NASA | Centre de géomatique du Québec — cgq.qc.ca*. https://www.cgq.qc.ca/mission-swot-nasa/. [Accessed 22-04-2025].

[9] Cdsentec. *Leading provider of open channel flow meters - SENTEC*. Août 2023. URL : https://cdsentec.com/open-channel-flow-meter/.

[10] *SWOT : un satellite pour cartographier le cycle de l'eau sur Terre | CNRS Terre Univers*. URL : https://www.insu.cnrs.fr/fr/cnrsinfo/swot-un-satellite-pour-cartographier-le-cycle-de-leau-sur-terre.

[11] R. J. Adrian. "Twenty years of particle image velocimetry". In : *Experiments in Fluids* 39.2 (2005), p. 159-169. DOI : 10.1007/s00348-005-0991-7.

[12] M. Muste, I. Fujita et A. Hauet. "Large-scale particle image velocimetry for measurements in riverine environments". In : *Water Resources Research* 44.4 (avr. 2008). DOI : 10.1029/2008wr006950. URL : https://doi.org/10.1029/2008wr006950.

[13] Matthew Perks, Andrew Russell et Andy Large. "Advances in remote sensing of river hydro-morphology and hydraulic processes". In : *Earth Surface Processes and Landforms* 41.14 (2016), p. 1967-1978. DOI : 10.1002/esp.3964.

[14] F. Tauro, R. Piscopia et S. Grimaldi. "Streamflow observations from cameras : Large-Scale particle image velocimetry or particle tracking velocimetry ?" In : *Water Resources Research* 53.12 (nov. 2017), p. 10374-10394. DOI : 10.1002/2017wr020848. URL : https://doi.org/10.1002/2017wr020848.

[15] Pariva Dobriyal et al. "A review of methods for monitoring streamflow for sustainable water resource management". In : *Applied Water Science* 7.6 (oct. 2016), p. 2617-2628. DOI : 10.1007/s13201-016-0488-y. URL : https://doi.org/10.1007/s13201-016-0488-y.

[16] Carl J. Legleiter, Paul J. Kinzel et Jonathan M. Nelson. "Remote measurement of river discharge using thermal particle image velocimetry (PIV) and various sources of bathymetric information". In : *Journal of Hydrology* 554 (sept. 2017), p. 490-506. DOI : 10.1016/j.jhydrol.2017.09.004. URL : https://pubs.usgs.gov/publication/70192233.

[17] J. Le Coz et al. "Performance of image-based velocimetry (LSPIV) applied to flash-flood discharge measurements in Mediterranean rivers". In : *Journal of Hydrology* 394.1-2 (juin 2010), p. 42-52. DOI : 10.1016/j.jhydrol.2010.05.049. URL : https://doi.org/10.1016/j.jhydrol.2010.05.049.

[18] Pgp. *View of Optical sensing for stream flow observations : A review*. URL : https://www.agroengineering.org/jae/article/view/836/719.

[19] Markus Raffel et al. *Particle Image Velocimetry : A Practical Guide*. 2nd. Springer, 2007. DOI : 10.1007/978-3-540-72308-0.

[20] David W. Dobson, Kieran Holland et Joseph Calantoni. "Fast, large-scale, particle image velocimetry-based estimations of river surface velocity". In : *Computers & Geosciences* (2014). DOI : 10.1016/j.cageo.2014.05.007.

[21] Martin J. Jolley et al. "Considerations when applying Large-Scale PIV and PTV for determining river flow velocity". In : *Frontiers in Water* 3 (déc. 2021). DOI : 10.3389/frwa.2021.709269. URL : https://doi.org/10.3389/frwa.2021.709269.

[22] Ichiro Fujita, Marian Muste et Anton Kruger. “Large-scale particle image velocimetry for flow analysis in hydraulic engineering applications”. In : *Journal of Hydraulic Research* 36.3 (mai 1998), p. 397-414. DOI : 10.1080/00221689809498626. URL : https://doi.org/10.1080/00221689809498626.

[23] Alexandre Hauet, Jean-Dominique Creutin et Philippe Belleudy. “Sensitivity study of large-scale particle image velocimetry measurement of river discharge using numerical simulation”. In : *Journal of Hydrology* (2008). DOI : 10.1016/j.jhydrol.2007.10.062. URL : https://www.sciencedirect.com/science/article/abs/pii/S0022169407006518.

[24] Quinn W. Lewis et Bruce L. Rhoads. “Resolving two-dimensional flow structure in rivers using large-scale particle image velocimetry : An example from a stream confluence”. In : *Water Resources Research* (2015). DOI : 10.1002/2015wr017783.

[25] Flavia Tauro, A. Petroselli et Salvatore Grimaldi. “Optical sensing for stream flow observations : A review”. In : *Journal of Agricultural Engineering* 49 (avr. 2018). DOI : 10.4081/jae.2018.836.

[26] *Hydro-STIV portable*. URL : https://www.hydrosoken.co.jp/en/hydrostiv-portable/.

[27] *Laboratory and field LSPIV measurements of flow velocities using FuDaA-LSPIV, a free User-Friendly software*. URL : https://www.iahr.org/library/infor?pid=19562.

[28] Niels van Steenbergen. *pyOpenRiverCam documentation*. https://github.com/nlesc-jcer/pyOpenRiverCam. 2023.

[29] *Information for boaters on the River Great Ouse*. Avr. 2022. URL : https://www.gov.uk/guidance/river-great-ouse-bridge-heights-locks-and-facilities.

[30] Marian Muste, Ichiro Fujita et Alexandre Hauet. “Large-scale particle image velocimetry (LSPIV) for stream flow measurements”. In : *Flow Measurement and Instrumentation* 19.3-4 (2008), p. 137-147. DOI : 10.1016/j.flowmeasinst.2008.03.001.

[31] École normale supérieure de Lyon. *Transect — géoconfluences*. URL : https://geoconfluences.ens-lyon.fr/glossaire/transect.

[32] *Présentation des données netCDF—ArcGIS Pro | Documentation*. URL : https://pro.arcgis.com/fr/pro-app/latest/help/data/multidimensional/a-quick-tour-of-netcdf-data.htm#:~:text=NetCDF%20(Network%20Common%20Data%20Form,ces%20donn%C3%A9es%20et%20leur%20partage..

[33] William Thielicke. *PIVlab tutorial*. Déc. 2024. URL : https://pivlab.blogspot.com/p/blog-page_19.html.

[34] GeeksforGeeks. *Laplacian Filter using Matlab*. URL : https://www.geeksforgeeks.org/laplacian-filter-using-matlab/.

[35] Deepak Gaur, Deepti Mehrotra et Karan Singh. “A new approach to simulate the dynamic behavior of particulate matter using a canny edge detector embedded PIV algorithm”. In : *The Visual Computer* 39 (juill. 2022). doi : 10.1007/s00371-022-02591-2.

[36] Qiming Hu et al. “Space-Time Image Velocimetry Based on Improved MobileNetV2”. In : *Electronics* 12.2 (2023), p. 399. doi : 10.3390/electronics12020399.

**UNIVERSITÉ CATHOLIQUE DE LOUVAIN**
**École polytechnique de Louvain**
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**