

# Perception chimique chez la larve de poisson zèbre

Benjamin Gallois

22 juillet 2020

# **Abstract**

# Dedication

# Acknowledgements

# Table des matières

<b>I</b>	<b>FastTrack</b>	<b>6</b>
<b>1</b>	<b>Introduction</b>	<b>7</b>
1.1	Le tracking vidéo . . . . .	7
1.2	Le tracking, un problème compliqué . . . . .	8
1.3	Les logiciels existants . . . . .	10
1.3.1	Les logiciels propriétaires . . . . .	10
1.3.2	Les logiciels open-sources . . . . .	11
1.4	FastTrack : une approche différente . . . . .	14
<b>2</b>	<b>Conception et réalisation</b>	<b>16</b>
2.1	Outils utilisés . . . . .	16
2.2	Implémentation . . . . .	17
2.2.1	Détection . . . . .	18
2.2.2	Association . . . . .	21
2.2.3	Post-traitement . . . . .	22
2.3	Déploiement . . . . .	22
2.3.1	CI/CD . . . . .	22
2.3.2	Documentation . . . . .	24
<b>3</b>	<b>Base de données de films</b>	<b>27</b>
<b>4</b>	<b>Résultats</b>	<b>29</b>
4.1	Analyse de la base de données . . . . .	29
4.2	Estimation de la charge de travail de correction . . . . .	30
4.3	Optimisation des paramètres . . . . .	32
<b>5</b>	<b>Perspective</b>	<b>36</b>



Première partie

**FastTrack**

# Chapitre 1

## Introduction

Talk is cheap. Show me the code.

---

Linux Torvald

### 1.1 Le tracking vidéo

Le tracking d'objets à partir d'enregistrements vidéo est un problème qui a beaucoup gagné en popularité ces dernières années. Cela est dû en grande partie à son grand potentiel, aussi bien en science que pour des applications commerciales ou de sécurité. On citera par exemple le cas des voitures autonomes [18] qui peuvent se conduire seul, ou bien le projet ATTOL [1] de la compagnie Airbus qui permet un décollage, un atterrissage et roulage au sol entièrement automatisé reposant uniquement sur de l'analyse d'images. Une grande partie des efforts de recherche est mise sur la création d'algorithmes de reconnaissance et de tracking de piétons [], dans le but d'automatiser l'analyse des données de vidéo surveillance. Le tracking est aussi très utilisé dans le domaine du cinéma et des effets spéciaux (VFX). Que ce soit pour stabiliser les prises de vues ou bien réaliser des effets spéciaux (ex. motion capture). Le tracking automatisé sur images permet de réduire les coûts et les temps de production. En science, l'utilisation du tracking automatisé, surtout en biologie et en écologie [4], est un domaine en plein essor. Il évite de devoir marquer de manière invasive les animaux et donc de les perturber. Il permet de générer un grand nombre de données fiables, de réduire les biais,

en évitant au passage une analyse manuelle longue et laborieuse.

Le tracking d'objets peut être séparé en deux catégories : le Single Objet Tracking (SOT), où le but est de détecter un seul objet dans une scène plus ou moins complexe, et le Multiple Object Tracking (MOT) [6] où il faut cette fois-ci détecter et suivre plusieurs objets. On se placera ici dans le cadre du MOT en quasi deux dimensions appliquée en grande majorité au domaine scientifique. Cela réduit grandement la difficulté du problème, en général la qualité des images est bonne, la caméra est fixe, et on peut optimiser l'éclairage pour faciliter la détection des objets. En contrepartie, la tolérance aux erreurs est faible si on veut pouvoir produire des données fiables, et donc des conclusions scientifiques robustes. Un point déterminant est la performance du l'algorithme qui doit pouvoir analyser les données dans un temps raisonnable comparé à leur débit de production, et répondre aux contraintes matérielles et technique de l'utilisateur. La facilité d'installation et d'utilisation du logiciel qui intègre l'algorithme ne doit pas être négligée. Les utilisateurs amenés à utiliser ces logiciels ne sont en général pas des experts du domaine informatique et d'analyse d'images et le logiciel doit être facilement installable et utilisable.

On verra d'abord en quoi le tracking est un problème complexe et, comment on peut réduire ou contourner cette complexité. En présentera ensuite une liste non exhaustive des logiciels existants en matière de tracking appliquée aux domaines scientifique. Enfin on verra en quoi l'approche de FastTrack, le logiciel que nous avons créé pour résoudre le problème du tracking, est différente et dans quels cas elle peut être utilisé.

## 1.2 Le tracking, un problème compliqué

Le tracking d'objets à partir d'images repose sur trois étapes : l'acquisition des images qui en fonction des paramètres d'acquisition conditionnera la difficulté du tracking et le type d'algorithme utilisable, la détection des objets qui consiste à séparer les objets du fond, et enfin l'assignation des objets d'une image sur l'autre permettant de conserver leurs identités. La tracking d'objet est en général un problème de traitement d'image complexe [4]. D'une part en fonction des objets étudiés, par exemple en biologie les animaux sont des objets grandement déformables ce qui complique la détection. D'autre

part la scène peut être complexe, avec des objets disparaissant derrière des éléments du décors, se superposant (phénomène dit d'occlusion) ou entrant et sortant du champ de vue en permanence, complique l'étape d'association.

Les problème de détection d'objets peuvent en général être contourner par la conception de l'expérience quand cela est possible. Un point de vue fixe et une optimisation de l'éclairage permettent en général une simple détection par soustraction d'une image de fond (sans objets) et application d'un seuil. Pour les cas plus compliqués, une grande variété algorithmes est disponible [20] et applicable suivant la qualité des images. Les plus courants sont la détection de points d'intérêt de l'objet, invariant par rapport au point de vue et de l'illumination, cette technique nécessite une bonne qualité d'image. La segmentation permet de séparer l'image par zone de similitudes et donc de détecter les objets d'intérêt, de nombreux algorithmes et approches existent pour segmenter une image. Le machine learning peut aussi être appliqué pour la détection des objets [21].

Deux grandes classes d'algorithmes se distinguent pour mitiger les problèmes d'association. La première classe d'algorithme utilise les quantités cinématiques de l'objet, comme sa vitesse ou sa position [11, 12], pour prédire ou retrouver à postériori la position de l'objet ainsi garder son identité. Le taux d'erreur de cette méthode reste constant quand on augmente le nombre d'individus, elle est en générale rapide et cela en fait donc un bon candidat pour les application de tracking en temps réel. L'inconvénient majeur de cette approche vient du phénomène de propagation des erreurs. Si l'algorithme fait une erreur dans l'assignation, alors il n'a aucun moyen de corriger l'erreur à l'étape suivante et elle se propage jusqu'à la fin de l'analyse. Une deuxième classe d'algorithme est elle basée sur la reconnaissance de features de l'objet, permettant ainsi de créer une "empreinte digital" unique à chaque objet, en utilisant soit une méthode classique [2, 10], soit du machine learning [7, 14]. Cela permet de solutionner le problème de la propagation des erreurs d'assignation et également de suivre des objets dans le temps, par exemple un animal d'un jour d'expérience sur l'autre, ce qui peut être très utile notamment pour certaines applications d'étude du comportement. Cette méthode nécessite une image de qualité suffisante pour détecter des features représentative de l'objet. En général, elle demande aussi plus de ressources matériels, donc une analyse qui ne peut pas être faite en temps réel. Mais la principale limitation est le nombre d'objets qu'elle peut suivre, il est actuellement limité à

une dizaine d'objets par image avant que les performances des algorithmes se dégradent significativement avec les méthodes classiques. L'approche par machine learning permet d'augmenter le nombre d'objets au prix d'un temps de calcul long et la nécessité d'avoir recourt à des machines très performantes.

## 1.3 Les logiciels existants

De nombreux logiciels de tracking existent déjà, on fera ici une liste non exhaustive des plus populaires en les séparant en deux catégories : les logiciels open-sources et les logiciels propriétaires.

### 1.3.1 Les logiciels propriétaires

Les logiciels propriétaires présentés ici sont closed-sources, l'utilisateur ne peut pas modifier le code pour adapter le logiciel à son projet. En contrepartie ils ne nécessitent aucune connaissance en informatique et bénéficie d'un service d'assistance. Ils sont une bonne alternative aux plugins parfois difficiles à installer mais leurs prix élevés peuvent être un frein pour certain laboratoire.

**EthoVision XT** EthoVision XT est un logiciel développé par la société Noldus. Il accompagne l'utilisateur de l'acquisition des images grâce à un système de templates d'expériences, jusqu'à l'analyse des données avec un module permettant de visualiser les quantités intéressantes. Le logiciel est très complet et très largement utilisé. Il est plutôt spécialisé dans le domaine des neurosciences comportementales. Il inclut des modules pour les expériences typiques comme un "module d'interaction social pour rats", "water maze" etc... Il permet aussi d'effectuer le tracking en direct, ce qui permet de ne pas à avoir à enregistrer les images.

EthoVision XT est un logiciel mature, un grand nombre de modules est disponible ainsi qu'un système permettant de créer soit-même des templates d'expérience. Le plus gros inconvénient est que l'utilisateur ne peut pas modifier le logiciel ou contrôler comment le tracking y est effectué. Le prix peut-être une barrière pour certains utilisateurs, le logiciel coûtant au minimum 5 850 USD sans module. Il est compatible avec Windows uniquement.

**Any-maze** Any-maze est un logiciel développé par la société Stoelting Co. Any-maze est spécialisé dans l’analyse des expériences de neuroscience comportementales. Il intègre directement des outils pour les tests les plus connus tel que le ”forced swim test” ou le ”fear conditionning test” permettant une analyse entièrement automatique de ces expériences. Il peut faire du tracking en temps réel ou depuis des vidéo déjà enregistrés.

Any-maze est très complet dans l’analyse des expériences typiques de comportement et peut être acheté avec le dispositif expérimentale déjà optimisé et calibré pour le logiciel. La suite Any-maze se décompose en trois logiciels, la partie tracking est disponible au prix de 6 495 USD ou 1 500 USD l’année. Le logiciel est disponible pour Windows uniquement.

### 1.3.2 Les logiciels open-sources

Les logiciels open-sources permettent à l’utilisateur de modifier et distribuer le logiciel. C’est la meilleure alternative aux logiciels commerciaux. En contrepartie aucune assistance n’est fournie, le développement collaboratif de certains logiciels (par exemple sur Github) permet de signaler des bugs et de participer à leur correction et donc d’améliorer le logiciel.

**idTracker** idTracker [10] est une bibliothèque MATLAB qui permet de tracker plusieurs objets. Il est basé sur l’extraction d’une ”empreinte digital” pour chaque objet, et donc une détection fiable sans propagation d’erreurs. L’avantage d’idTracker est qu’il peut reconnaître un objet sur plusieurs vidéo et après une relativement longue période de temps, ce qui peut être utile pour suivre le comportement d’individus sur plusieurs séries d’expériences s’étalant dans le temps. IdTracker est cependant limité par le nombre d’objets qu’il peut tracker, actuellement une vingtaine, dû à la longueur des vidéos nécessaire à l’extraction optimale de ”l’empreinte digitale” de chaque objet, elle peut aller jusqu’à 30 minutes minimum pour une grande densité d’objet. La qualité requise des images est un facteur important et doit au moins être de 150 pixels par animaux. Le temps de calcul est relativement long, de l’ordre de 0.5 à 2 secondes par image et nécessite une grande quantité de RAM. L’installation d’idTracker peut se faire sans avoir besoin d’acheter MATLAB grâce au Matlab Run Time Compiler mais uniquement sous Windows. Il est par conséquent nécessaire d’acheter une licence MATLAB pour les autres plateformes, et d’avoir une connaissance minimale du langage pour mettre en place idTracker.

**ToxTrack** ToxTrack [13] est un logiciel qui implémente dans une interface graphique l'algorithme ToxId [12]. L'algorithme extrait les objets du fond en appliquant un seuil. Les morceaux de trajectoire entre chaque occlusion sont ensuite divisés en trajectoires courtes et trajectoire longues basé sur une durée seuil définis par l'utilisateur. Un groupe de trajectoires longues où tous les individus sont observés en même temps est alors extraite, dans ce cas là l'assignation est fait grâce à l'algorithme Hongrois. Les trajectoires courtes sont ensuite assignées à l'objet correspondant par corrélation entre une matrice d'identification des trajectoires. Les auteurs rapportent que ToxId est aussi performant que les logiciels déjà existants, très rapide, il est possible de tracker les objets en temps réel. Un désavantage que l'on peut constater dans cette algorithme est qu'il ne marche seulement pour un nombre constant d'animales. L'initialisation de l'algorithme nécessite d'avoir à un instant  $t$  l'ensemble des objets à tracker simultanément détectable pendant un temps  $t + dt$  défini par l'utilisateur. Le gui inclus des outils permettant de définir des zones D'intérêt ainsi qu'une analyse statistique des données recueilli. On constate que la matrice de corrélation de trajectoire ne permet pas une reconnaissance de l'objet et que les erreurs dans cette méthode sont propagations. Le logiciel est disponible pour Windows et compilable pour les autres systèmes.

**DeepLabCut** DeepLabCut [7] est un framework qui permet de résoudre le problème dit de "pose estimation" qui consiste à retrouver un objet et sa position, ou une partie d'objet, dans une image. Il peut directement s'apparenter au problème du SOT si les objets à tracker sont différents, exemple une oreille droite et un nez de souris, qui pourront ensuite être retrouvé sur chaque image puis associé dans le cas où il n'y a qu'une seul souris. Dans le cas de plusieurs objets similaire à retrouver et associer d'une image sur l'autre (MOT), il faudra combiner cette détection avec une association pour obtenir le tracking. Même si DeepLabCut répond à un problème légèrement différent, il peut, par sa conception être couplé à un algorithme d'association externe pour en faire un logiciel de tracking.

DeepLabCut est directement basé sur l'algorithme de détection de feature du framework DeeperCut [5], spécialisé dans la détection des mouvements du corps humain. Les auteurs de DeepLabCut on étudié les performances de cet algorithme appliqué au domaine des neurosciences comportementales, comme la détection du museau d'une souris, ou des pattes d'une drosophile,

et on ajouté des outils permettant de facilement entraîner l'algorithme et de tester sa robustesse. Il tire partie du deep learning pour résoudre le problème de "pose estimation". Pour rappel, le deep learning est un type d'algorithme de machine learning qui consiste à entraîner un réseau de neurones possédant plusieurs couches. Ce réseau est constitué de plusieurs ResNets (residual neural network) qui ont été pré-entraînés sur la base de données ImageNet. Des couches de déconvolution sont placées en sortie des ResNets et permettent d'obtenir la probabilité de présence de l'objet dans l'image. Le réseau est ensuite finement ajusté en l'entraînant sur des images où sont annotées les parties qui doivent être détectées. Les auteurs ont ainsi montré qu'on peut obtenir une performance au moins aussi bonne que celle d'une détection par un être humain avec très peu de donnée d'entraînement, de l'ordre de 200 images.

DeepLabCut, comme indiqué précédemment, est une framework, et malgré une très bonne documentation [9], il peut être difficile à mettre en place pour un utilisateur possédant peu de compétence en informatique. Le processus d'installation durant de 10 à 60 minutes et nécessitant l'installation d'un GPU et des drivers associés pour tirer au mieux partie du logiciel. De plus, l'algorithme demande beaucoup de puissance de calcul et un GPU (NVIDIA GForce 1080 8GB au moins) est fortement conseillé. Pour donner un ordre d'idée, des images de 682x540 pixels, analysées avec un GPU de dernière génération, permet d'attendre une vitesse d'analyse de 30 images par seconde, sans GPU ce temps peut être multiplié par un facteur 10 ou 100 [8]. On voit donc que DeepLabCut présente un grand intérêt pour retrouver dans une image des objets avec une grande précision. Il s'adresse plus particulièrement aux neurosciences comportementales où il permet de suivre des mouvements complexes (ex. doigt de la main chez une souris). Il ne conviendra pas aux utilisateurs avec peu de connaissance en informatique s'intéressant à des problèmes plus larges et ayant une grande quantité de données à traiter.

**idTracker.ai** IdTracker.ai [14] est un framework qui permet de tracker des animaux (de l'ordre d'une centaine) avec une précision quasi parfaite. Id-Tracker.ai tire partie du deep learning pour réaliser le tracking. Dans une première étape, chaque objet est segmenté en appliquant un seuil. Un réseau convolutionnel classe ensuite chaque blob détecté comme contenant un seul objet, ou contenant plusieurs objets. Un autre réseau convolutionnel permet

ensuite de trouver l'identité de chaque individu tout au long du film. Pour fonctionner, ce système nécessite assez de donnée pour pouvoir entraîner le réseau à reconnaître chaque individu. Les auteurs ont trouvé qu'un tracking robuste peut être obtenu avec seulement 30 images isolées de chaque individu comme base, il faut donc prévoir au minimum des vidéos de 500 images pour une dizaine d'individus avec un minimum de 25 fps. Une résolution de 300 pixels par animale est recommandée pour avoir une bonne précision de tracking. Un facteur limitant d'idTracker.ai est qu'il demande beaucoup de temps de calcul et beaucoup de RAM, les auteurs rapportent une durée du vingtaine de minutes pour 8 zebrafish et de l'ordre de 6 heures pour 100 zebrafish sur environ 2000 images hautes résolutions. Même si un gui est disponible pour aider l'utilisateur, des connaissances de base en informatique et en programmation sont nécessaires, tout comme un matériel adapté, l'utilisation d'un GPU est fortement recommandé. Il est recommandé à des utilisateurs souhaitant un tracking parfait, entièrement automatisé depuis des images de grandes qualité et des films assez long possédant une machine puissante et sans contrainte de temps.

## 1.4 FastTrack : une approche différente

On a vu précédemment les logiciels existants ainsi que les différents paradigmes du tracking appliqués aux sciences. On voit qu'une approche rapide, nécessitant peu de puissance de calculs, et généraliste (pouvant s'appliquer à différents systèmes, nombre d'objets variable etc...) est manquante. Pour combler ce vide, nous avons conçu un logiciel nommé FastTrack. Ce logiciel contient deux parties distinctes :

- Une interface où sont implémentées les procédures standards d'analyse d'images permettant la détection des objets, et un algorithme de tracking qui permet de conserver l'identité des objets d'une image sur l'autre, rapide et avec un faible taux d'erreur.
- Une interface ergonomique permettant la vérification et éventuellement la correction manuelle de toutes erreurs ayant pu être commises par l'algorithme de tracking.

On remarquera ici la différence d'approche entre FastTrack et les logiciels existants. Au lieu de développer un système nécessitant une grande puissance de calcul, longue mais apportant des résultats entièrement automatisés et d'une grande fiabilité, FastTrack implémente une méthode rapide et fa-

cile à régler, très généraliste. La correction des erreurs restantes est laissé à la charge de l'utilisateur mais peut-être effectué nativement dans le logiciel, l'interface ergonomique permettant une correction rapide et efficace. Cette solution présente plusieurs avantages, le premier étant qu'elle ne nécessite aucune connaissance en programmation, n'importe quel utilisateur peut réaliser une analyse parfaite en très peu de temps. De plus, on a pu montrer que le travail de post-traitement peut être estimé a priori par une analyse des paramètres géométriques et dynamiques du système étudié ce qui permet à l'utilisateur de savoir si le logiciel est adapté à son besoin. Pour beaucoup de systèmes étudiés, le post-traitement ce résume à une rapide vérification. Dans certains cas, si le nombre d'occlusions est trop importantes et qu'une précision de tracking parfaite est nécessaire sans vouloir avoir recours à une correction manuelle, une autre solution doit être envisagée.

FastTrack est distribué sous une licence open-source et implémenté de manière modulaire et entièrement documenté. Chaque utilisateur peut ainsi modifier le logiciel à sa convenance ou y contribuer. L'algorithme de tracking est découplé de l'interface de détection et de correction se qui rend extrêmement facile l'intégration de FastTrack dans un projet déjà existant. Le logiciel est facilement installable en moins de 5 minutes et est compatible Linux, MacOs et Windows, il peut fonctionner sur des configurations modestes et des Single Board Computer (SBC) tel que le Raspberry Pi.

# Chapitre 2

## Conception et réalisation

Testing can only prove the presence of bugs, not their absence.

---

Edsger W. Dijkstra

### 2.1 Outils utilisés

Le choix des outils et bibliothèques utilisés dans la conception d'un logiciel est primordial et plusieurs facteurs de sélection doivent être pris en compte. Le premier critère à considéré est celui de la licence, j'ai fait le choix de mettre le logiciel FastTrack sous licence libre (GPL3) ce qui implique que le langage utilisé ainsi que les bibliothèques soient elles aussi sous des licences compatibles (MIT, GPL, etc...). Le choix d'une licence open-source est évident dans le cas d'un logiciel scientifique, l'utilisateur pouvant alors vérifier comment le logiciel fonctionne, le modifier pour l'adapter à ses besoins ainsi que le partager. Le deuxième critère est de bien choisir les bibliothèques utilisées en envisageant le futur du logiciel de manière à ne pas avoir à changer de bibliothèques si ses capacités s'avèrent insuffisante à mesure que le logiciel évolue. On privilégiera les bibliothèques matures offrant un support sur le long terme.

Dans cette perspective, FastTrack a été implémenter en C++ [16] en utilisant les bibliothèques Qt [17] et OpenCV [3] respectivement pour l'interface

graphique et l'analyse d'image. Les tests unitaires sont réalisés à partir de la bibliothèque Google Test.

Le C++ est un langage informatique créé par Bjarne Stroustrup en 1985. Offrant de grandes performances, il est standardisé par l'International Organization for Standardization (ISO) et est le langage de choix pour les applications d'analyse d'images ainsi que la création d'interfaces graphiques complexes.

Qt est une bibliothèque d'interface graphique open-source créée par Haavard Nord et Eirik Chambe-Eng, tous deux physiciens, en 1991 alors qu'il réalisaient un logiciel d'analyse d'images ultrasoniques. Très mature, possédant une vaste documentation, une très grande communauté, elle permet de réaliser des interfaces graphiques pour Linux, Mac et Windows avec le même code source.

OpenCV est une bibliothèque d'analyse d'images open-source créée par Intel en 2020. Très complète et efficace, elle est devenue la référence en matière d'analyse d'images aussi bien en recherche que pour la réalisation d'applications commerciales.

Google test est une suite permettant d'automatiser les tests unitaires en C++, elle est notamment utilisée par OpenCV. Le but des tests unitaires est de vérifier que chaque partie du programme fonctionne comme attendu. Cette pratique présente plusieurs avantages, les principaux étant d'identifier plus facilement d'éventuelle erreurs lors de l'implémentation de nouvelles fonctionnalités, et de faciliter le développement du logiciel quand celui-ci grandit en taille pour éviter toutes inclusions d'erreurs. Cette série de tests est automatiquement effectuée à chaque nouveau commit, voir section ??.

## 2.2 Implémentation

Le fonctionnement de FastTrack peut être séparé en trois parties que sont : la détection des objets, l'association des objets d'une image sur l'autre, et enfin une étape de correction.

Chaque analyse commence par l'ouverture d'une séquence d'images, ou

d'un film qui peut être automatiquement converti en séquence d'images. L'utilisateur a le choix entre deux types d'interfaces, une interface interactive où il ne peut ouvrir qu'un seul film à la fois. Elle permet de voir en temps réel l'impact des paramètres sur les images ce qui facilite la détermination des paramètres optimaux d'analyse. Une deuxième interface permet d'ouvrir simultanément une grande quantité de films, soit en donnant un fichier de paramètres ou en sélectionnant les paramètres dans l'interface. Elle est utile quand l'utilisateur veut analyser un grand nombre de films dont il connaît déjà les paramètres optimaux d'analyse.

Les deux interfaces peuvent être utilisé de manière complémentaire. L'utilisateur peut trouver les paramètres optimaux avec l'interface interactive et ensuite automatiser l'analyse d'un grand nombre de films en les ajoutant par lots dans le logiciel.

### 2.2.1 Détection

L'étape de détection a pour but d'extraire les paramètres cinématiques de chaque objet qui seront ensuite utilisé dans l'étape d'association. FastTrack inclut une collection de filtres d'analyse d'images permettant à l'utilisateur d'optimiser la détection des objets sans avoir à recourir à un logiciel externe.

**Calcul du fond** Chaque analyse commence par calculer une image de fond. Si l'utilisateur possède déjà une image de fond préalablement enregistré, il peut directement l'ouvrir dans le logiciel. Sinon trois méthodes de calcul sont possibles :

- Projection du maximum d'intensité.
- Projection du minimum d'intensité.
- Projection de la moyenne d'intensité.

Les trois méthodes reposent sur le même principe. L'utilisateur choisit  $n$  images dans la séquence, le logiciel va projeter dans la direction perpendiculaire à l'image soit le maximum, soit le minimum ou bien la moyenne de la séquence. En pratique, on projettera le maximum (resp. minimum) si les objets sont plus foncés (resp. clairs) que le fond de manière à faire disparaître les objects et ainsi obtenir le fond. L'utilisateur peut effectuer une registration de chaque images avant projection de manière à corriger un éventuel mouvement de la caméra.

**Registration** L'utilisateur peut choisir d'effectuer une registration des images, trois méthodes sont proposées dans le logiciel. Chaque méthode est implémenté de manière pyramidale, c'est à dire que la registration est d'abord effectuée sur une image dégradée pour corriger de manière grossière le déplacement, puis la correction est affinée en augmentant la qualité de l'image jusqu'à arriver à l'image originel. Cela permet d'accélérer le processus, la registration étant souvent un procédé relativement coûteux en temps de calculs.

La première méthode proposée est la corrélation de phase. Elle permet de corriger les mouvements de translation entre deux images en utilisant le théorème de Fourier dans le domaine fréquentiel. Cette méthode est très rapide mais reste limité à de petits mouvements de translation uniquement.

La deuxième méthode proposées est la méthode Enhanced Correlation Coefficient (ECC). Dans FastTrack, elle est restreinte à corriger les mouvements de translation et de rotation uniquement. Elle consiste à utiliser le coefficient de corrélation comme mesure pour trouver la meilleure transformation entre deux images. Cette méthode a pour avantage d'être relativement rapide, le problème d'optimisation non linéaire pouvant être résolue de manière linéaire. Elle est performante pour des images bruités et ayant des distorsions photométrique (contraste, luminosité).

La troisième méthode est une méthode basée sur le repérage de points clefs. Elle permet de corriger les mouvements et les déformations (homographie). Les points clefs (environ 500) sont automatiquement déterminé sur deux images grâce à l'algorithme ORB. Ces points sont ensuite associés deux à deux en utilisant l'algorithme RANSAC permettant de trouver la meilleure transformation entre les deux images. Cette méthode, plus précise nécessite une qualité d'image suffisante pour pouvoir discerner des points clefs.

**Binarisation** Chaque image est ensuite binarisée en soustrayant l'image de fond puis en définissant une valeur seuil. Dans le mode interactif, l'utilisateur peut voir l'impact des paramètres sur l'image binarisée ce qui permet d'ajuster facilement le seuil de binarisation. Le logiciel détecte également si le fond est plus foncé (resp. clair) que les objets permettant d'avoir à la fin de cette opération une image binaire où les pixels appartenant à l'objet sont égaux à 1 et les pixels appartenant au fond sont égaux à 0.

**Opération morphologique** Un ensemble d'opérations morphologiques (di-latation, érosion, ouverture etc...) peut-être effectué sur l'image binaire pour améliorer la détection et éliminer d'éventuels artefacts. Différentes formes et tailles de kernels sont disponibles.

**ROI** L'utilisateur peut sélectionner une région D'intérêt et exclure le reste de l'image de l'analyse. Cela permet d'accélérer le processus d'analyse et d'éviter la détection d'objets parasites. En mode interactif, cette ROI peut être dessiné directement sur l'image.

**Tri** Pour exclure les objets trop petits correspondant à du bruit ou trop gros comme par exemple deux objets superposés, l'utilisateur doit sélectionner deux tailles caractéristiques. En mode interactif, les objets sont coloriés soit en rouge, soit en vert suivant si leur taille appartient à la gamme de tailles sélectionnée.

**Extraction des paramètres cinématique** Sur la base des images binaires, le logiciel va détecter le contour de chaque objet. Une étape importante de toutes procédure de tracking est l'extraction des paramètres qui serviront dans l'étape d'association. C'est en générale dans le choix de ses quantités que les algorithme de tracking diffère pour se spécialiser à un type d'objet donné. Dans FastTrack, les paramètres extraient sont la positions du centre de masse ainsi que l'orientation de l'objet, dans quantité rapidement calculées et assez généralistes pour s'adapter à une grande diversité de type d'objets.

Pour cela, FastTrack calcule l'ellipse équivalente de l'objet à partir des moments d'ordre deux de l'image binaire. Cette procédure est accélérée en utilisant directement le contour grâce à la formule de Green. L'orientation de l'objet est donnée par l'axe majeur de l'ellipse équivalente. Sa direction est déterminée en projetant chaque pixel de l'objet sur l'axe majeure de l'ellipse. On calcule ensuite la skewness de la distribution des distances des points projetés par rapport au centre de masse. Le signe de la skewness est un indicateur robuste de l'asymétrie de l'objet par rapport à son axe majeure à partir de laquelle on peut déterminer la direction de l'objet.

Pour les objets déformables, la direction calculer précédemment peut être différente de la direction de déplacement de l'objet. Par exemple dans le cas du poisson zèbre, celui-ci déforme son corps de manière périodique pour se déplacer, seul la tête est dirigée dans la même direction que le mouvement. C'est pourquoi on décompose l'objet en deux ellipses équivalentes, l'utilisateur peut alors choisir quelle ellipse représente le mieux la direction de déplacement.

### 2.2.2 Association

L'étape d'association a pour but de conserver l'identité des objets d'une image sur l'autre. Pour se faire, FastTrack utilise une méthode dérivée de ?, qui tire partie du fait que la position et la direction de chaque objet change très peu d'une image sur l'autre. Pour chaque paire d'objet (i,j) appartenant à deux images successives, deux coûts sont calculés. Le coût "dur" calculé comme suit :

$$\begin{cases} h_{i,j} = 1 & \text{si } r_{i,j} < h_d \\ h_{i,j} = \inf & \text{sinon} \end{cases}$$

avec  $r_{i,d}$  la distance entre les objets i et j,  $h_d$  un seuil qui représente la distance maximale de déplacement autorisée entre deux images successives. Le coût "mou" calculé comme suit :

$$c_{i,j} = \frac{r_{i,j}}{s_d} + \frac{\delta\alpha_{i,j}}{s_\alpha}$$

où  $\delta\alpha_{i,j}$  est la valeur absolue de la différence angulaire entre les directions de i et j,  $s_d$  et  $s_\alpha$  sont des coefficients de normalisation qui représente un déplacement et une réorientation typique du système étudié entre deux images successives. Ces deux coûts sont ensuite multipliés élément par élément pour donner la matrice suivant :

$$C_{i,j} = \begin{cases} c_{i,j} & \text{si } r_{i,j} < h_d \\ \inf & \text{sinon} \end{cases}$$

La matrice de coup peut être rectangulaire si le nombre d'objets n'est pas constant comme lors d'occlusion ou d'ajout d'objets. Un paramètre de mémoire peut être sélectionné de manière à ne plus pouvoir assigner les objets si ceux-ci ont disparu depuis plus de  $n$  images. La meilleure association est celle dont la somme des coûts est minimale. Ce problème est appelé "the rectangular assignment problem" et peut être résolu de manière exacte en utilisant

l'algorithme hongrois, FastTrack utilise l'implémentation Kuhn-Munkres en C++ pour résoudre rapidement ce problème.

### 2.2.3 Post-traitement

**Correction manuelle** FastTrack intègre un outil de correction manuel du tracking. Une fois une analyse terminée, le résultat peut être affiché dans une interface ergonomique créée à cet effet. L'utilisateur peut rejouer le film en y superposant les résultats de l'analyse, il peut sélectionner un objet pour en consulter les paramètres (aire, contour, identité, etc...). L'utilisateur peut aussi directement corriger les erreurs en supprimant des objets ou en échangeant l'identité des objets. Cette interface est concue dans un soucis d'ergonomie et de performance. Des raccourcis claviers ainsi qu'une sélection à la volée des objets directement en cliquant sur la vidéo permettent à l'utilisateur de rapidement vérifier et corriger les analyses. Il est aussi possible d'enregistrer, en plus des données brutes d'analyse, un film avec les résultats du tracking superposés. Cette interface de correction manuel permet de déplacer la charge de travail qui est traditionnellement placé sur le pré-traitement des données, vers le post-traitement. Le logiciel reste alors général et peut s'adapter à une grande diversité d'objets et performant grâce à une interface spécialement pensée pour réduire le temps de post-traitement.

**Analyse** L'analyse statistique des données n'est pas implémenté dans Fast-Track. Après chaque analyse, le logiciel génère un dossier contenant les résultats. Le fichier principal est nommé tracking.txt et il contient les données brutes de l'analyse avec une image et un objet par ligne. Ce format est compatible avec tous les logiciels d'analyse les plus utilisés (R, Python, MATLAB), des exemples sont disponibles dans la documentation.

## 2.3 Déploiement

### 2.3.1 CI/CD

Le déploiement est une partie à ne pas négliger dans la conception d'un logiciel et deux aspects sont tout particulièrement importants à considérer. Du point de vue de l'utilisateur, le logiciel doit pouvoir être installé facilement sur les plateformes supportées. Du point de vue du mainteneur,

la partie de déploiement doit être facilement réalisable et reproductible de manière à pouvoir intégrer rapidement les correctifs et nouvelles fonctionnalités développées. C'est dans cette optique que FastTrack suit la philosophie CI/CD [15] [19] en tirant partie du nouveau système GitHub Actions.

L'intégration continue (CI) est un ensemble de pratiques qui a pour but d'intégrer rapidement les changements au projet de manière automatisée. Elle est couplée avec une automatisation des tests unitaires. FastTrack tire partie du système CI/CD de GitHub nommé Action, à chaque nouveau changement (commit<sup>1</sup>) ou nouvelle collaboration (pull-request<sup>2</sup>), une série de tests est automatiquement déclenchée. Ces tests vont vérifier le bon fonctionnement de l'algorithme de tracking ainsi que le formatage du code source. Seul les changements qui passent les tests peuvent être intégrés au projet ce qui garantie la reproductibilité des analyses ainsi que la cohérence du code source et de la documentation.

La livraison continue (CD) quant à elle automatise la livraison du logiciel dans sa forme finale. Elle permet d'intégrer rapidement les changements au logiciel sans avoir à le faire manuellement pour chaque plateforme supportée. Dans le cas de FastTrack, le CD est implémenter grâce à GitHub Action et une nouvelle version du logiciel est compilée pour Linux, MacOs et Windows à chaque nouveau commit qui est intégré à la branche principale. Des version stables du logiciel sont quand à elle compilée à chaque palier de développement. Ce système est un gain de temps majeur pour un logiciel multi-plateformes comme FastTrack, et il permet à l'utilisateur de toujours disposer des derniers correctifs et des dernières fonctionnalités.

FastTrack supporte nativement les trois plateformes majoritairement utilisées : les systèmes Linux avec une AppImage qui supportent toutes les distributions et un PPA pour Ubuntu uniquement, Windows avec un installateur, MacOs avec une App. La dernière version stable peut-être téléchargé sur le site web <http://www.fasttrack.sh>, la dernière version CD sur <https://github.com/bgallois/FastTrack/releases>. La procédure pour compiler soit même le logiciel est disponible dans la documentation du développeur

- 
1. Action d'envoyer la liste de modifications effectuées dans le système de gestion de version
  2. Action de demander l'ajout de modifications effectuer dans une branche au projet

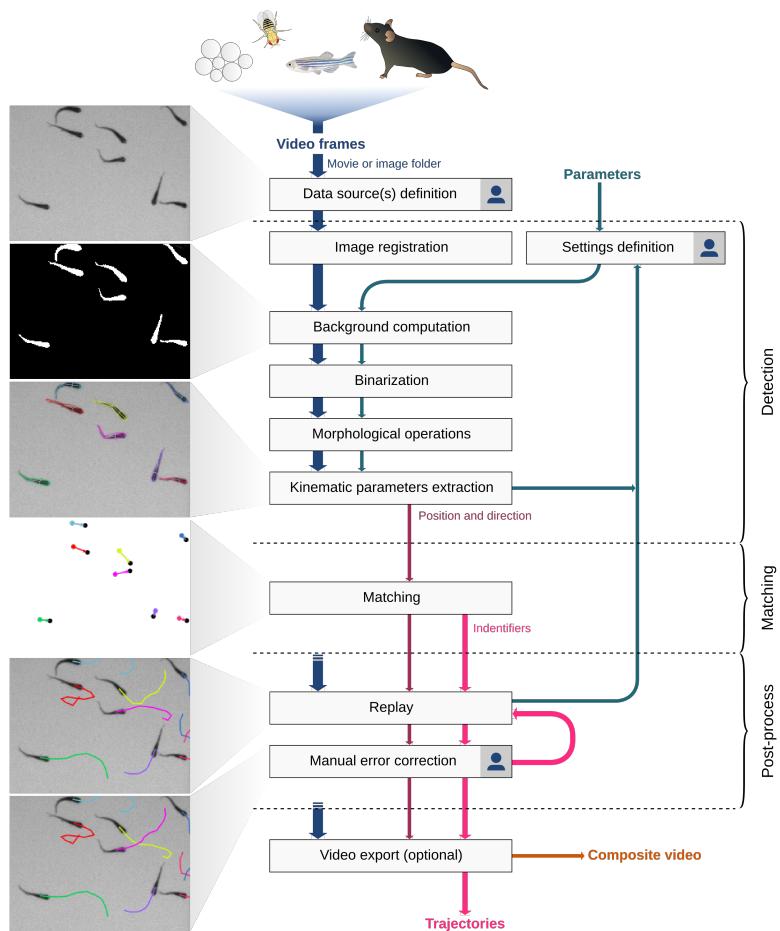
pour les autres plateformes.

### 2.3.2 Documentation

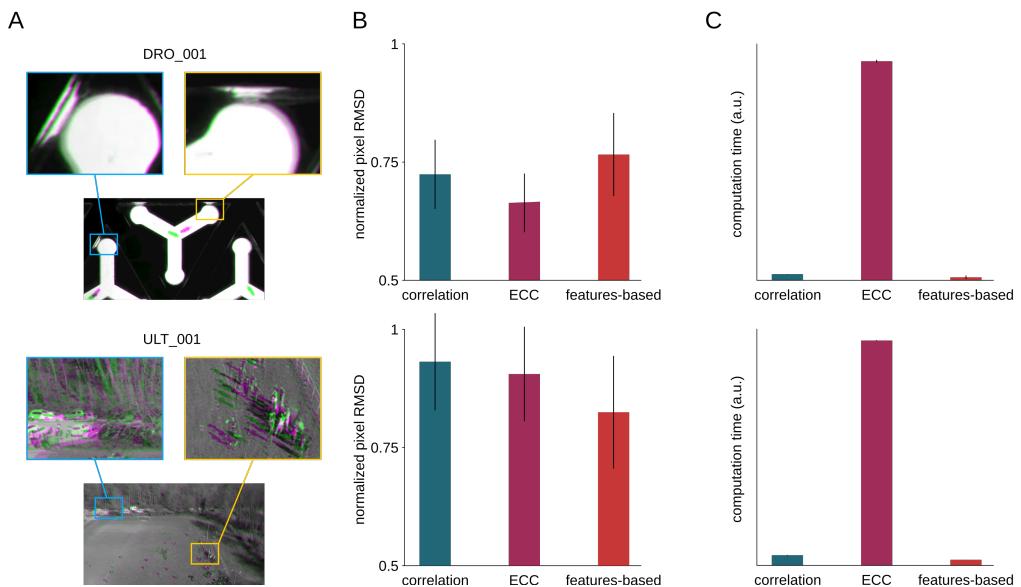
Une documentation extensive est disponible, elle se sépare en deux parties : une à l'usage des utilisateurs et une autre à l'usage des développeurs.

**Utilisateur** La documentation utilisateur est disponible à l'adresse <https://www.fasttrack.sh/UserManual/docs/intro.html>. Cette documentation est mise en forme à partir du logiciel Docusaurus et les utilisateurs peuvent y contribuer <https://github.com/bgallois/FastTrack/>. Elle regroupe l'ensemble des informations nécessaires à l'utilisation du logiciel ainsi que des vidéos d'explication pour aider à prendre le logiciel en mains.

**Développer** La documentation du développeur est disponible à l'adresse <https://www.fasttrack.sh/API/index.html>. Elle est automatiquement générée grâce au logiciel Doxygen à partir de la documentation présente dans le code source de FastTrack. Elle regroupe l'ensemble des informations nécessaires aux développeurs voulant modifier ou contribuer à FastTrack.



**Figure 2.1 – FastTrack flux de traitement.** Il se divise en trois parties distinctes : la détection, l'association et le post-traitement.



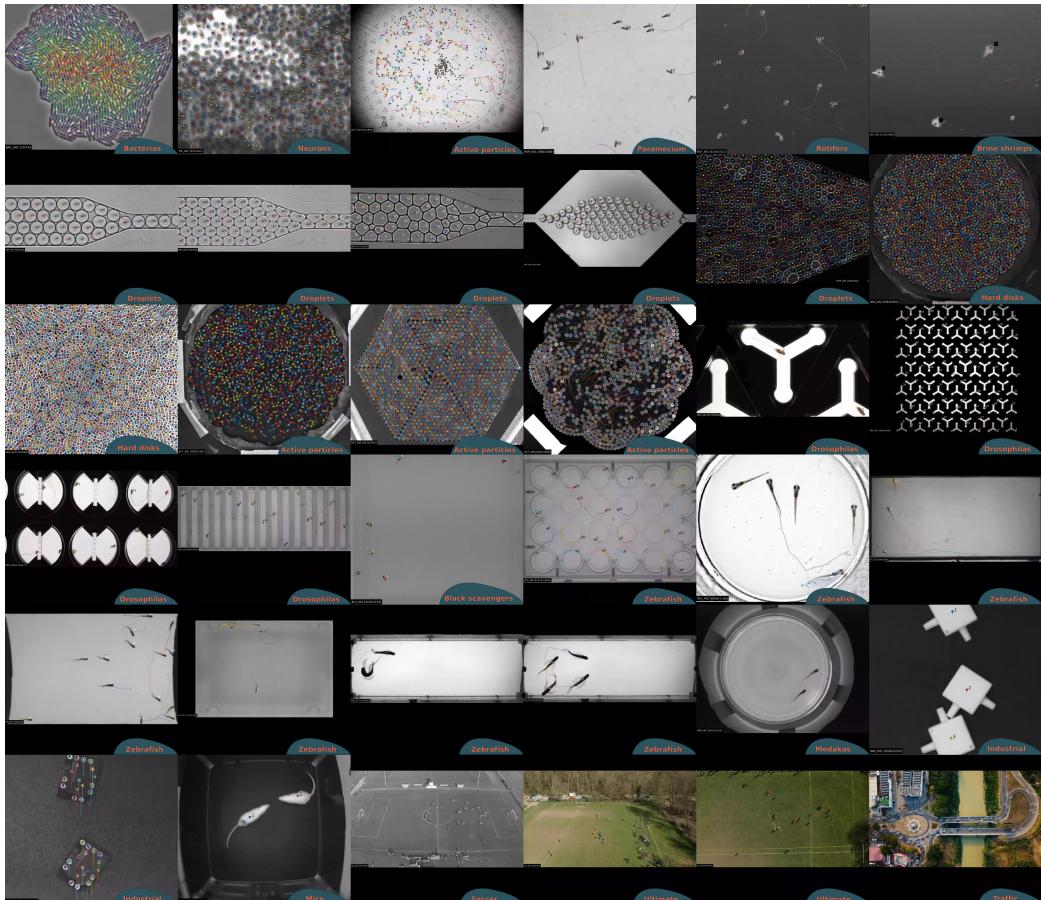
**Figure 2.2 – Registrations.** Deux films avec une dérive sévère sont utilisés comme référence (DRO\_001 en haut ULT\_001 en bas). **(A)** Comparaison entre une image (magenta) avec la première image (vert), avec agrandissement des détails en insert. **(B)** Déviation moyenne carrée (RMSD) de l'intensité des pixels avant et après registration, moyennée sur toutes les images et normalisée par la RMSD sans registration pour les trois méthodes. Barres d'erreur : déviation standard. **(C)** Temps moyen relatif pour les trois méthodes (unités arbitraires). Barres d'erreur : déviation standard.

# Chapitre 3

## Base de données de films

Pour démontrer que FastTrack peut être utilisé pour analyser des films provenant d'une grande diversité de systèmes, nous avons compilé une base de données de films nommée  $TD^2$ . Cette banque de films peut être téléchargé à l'adresse suivante URL. Les films proviennent soit de données déjà publiées dans la littérature, soit ils nous ont été fourni par les auteurs eux-mêmes sous licence CC-BY-NC-SA. Chaque film est identifié par un code à 3 lettres définissant le système (ex. ACT : active matter, ZFA : zebrafish adult etc...) et 3 chiffres pour indexer les films provenant de systèmes identiques.  $TD^2$  regroupe actuellement 41 films comprenant différent types d'objets de nature et de taille très différents : 7 espèces d'animaux allant du poisson à la mouche, des cellules, des particules actives, des gouttes microfluidiques ainsi que des objets macroscopiques tels des joueurs d'ultimate ou des voitures. Une vidéo donnant un rapide aperçut de tous les systèmes utilisés est disponible à l'adresse suivante URL.

Un autre aspect important à considérer est le nombre d'objets par film ainsi que leurs éventuelles apparitions, disparitions et chevauchements. Dans 22 films sur 41, le nombre d'objets est variable et des objets sortent ou rentrent dans le champ de la caméra durant l'enregistrement. Dans 19 films sur 41, des objets peuvent se chevaucher créant un phénomène d'occlusion que le logiciel doit gérer pour retrouver l'identité des objets.



**Figure 3.1 –  $TD^2$ .**

# Chapitre 4

## Résultats

### 4.1 Analyse de la base de données

Analyser des films provenant de systèmes aussi différents que ceux compilés dans  $TD^2$  est un véritable challenge. Cela est dû en partie aux conditions d'enregistrement qui peuvent être très diverses et qui complexifient la tâche de détection des objets. Deux difficultés récurrentes sont à soulever : les variations de l'illumination (ex. réflexion dans GRA\_001, ombres dans SOC\_001) et le chevauchement des objets (ex. HXB\_001). Dans le cas de films provenant du milieu académique, les systèmes sont souvent conçus pour limiter ou contourner ces deux difficultés. Il est commun de trouver des films dont l'illumination est uniforme et constante. de même un confinement quasi-2D ainsi qu'un nombre d'objet restreint dans le champ de la caméra permettent de réduire le nombre d'occlusions. Dans  $TD^2$ , 23 films ont une illumination assez bonne pour être analysés directement avec FastTrack, les autres ayant dû subir un pre-processing individuel spécifique avant de pouvoir être analysées. Deux films ayant un nombre d'occlusions extrêmement élevé ont été écarté (HXB\_001 and ZFL\_001) car ne pouvant pas être analysé avec le logiciel. Les 39 films restant ont pu être analysés avec FastTrack sans difficulté. L'algorithme de Kuhn-Munkres étant de complexité  $O(n^3)$  le temps de calcul est en général assez rapide. Sur l'ordinateur utilisé pour l'analyse (ordinateur de bureau moderne), on observe une vitesse d'analyse pouvant aller jusqu'à 500 images/secondes. Les films les plus volumineux en taille et nombre d'images et en nombre d'objets n'ont pas pris plus de quelques dizaines de minutes. Chaque film a ensuite été corrigé manuellement grâce à l'outil intégré dans

le logiciel.

## 4.2 Estimation de la charge de travail de correction

FastTrack est conçu pour que la phase de post-traitement soit la plus faible possible, mais cette charge de travail varie grandement suivant les films analysés. On va ici montrer que cette charge de travail peut être estimée rapidement pour un film donnée. On assumera pour la suite que la détection et l'association ont déjà été réalisées.

La méthode repose sur l'évaluation de la probabilité d'incursion  $P_{inc}$ . Une incursion étant définie par la sortie d'un objet de sa cellule de Voronoï, prise en  $t_1$ , durant un trajet entre  $t_1$  et  $t_2$ . Le nombre d'incursions dépend de la distribution des déplacements, de la densité d'objets, de la géométrie de la cellule de Voronoï et du degré d'alignement des déplacements. On peut écrire cette probabilité :

où  $\rho = r\sqrt{d}$  est le déplacement réduit adimensionné,  $R(\rho)$  la distribution des  $\rho$  et  $p_{inc}(\rho)$  la probabilité géométrique d'incursion qui ne dépend que des propriétés géométriques de la disposition des objets. On calcule  $p_{inc}$  en prenant les cellules de Voronoï de tous les objets sur toutes les images, puis en déterminant la proportion des angles pour lesquels un déplacement de  $\rho$  implique un incursion dans une cellules de Voronoï voisine. Intuitivement, on peut se convaincre que  $p_{inc}$  va de 0 quand  $\rho \rightarrow \infty$ , à 1 quand  $\rho \gg 1$ . La forme de cette fonction est sensible à la densité des objets, compacts (ex. ACT\_002), clairsemés (ex. PAR\_001), et à la taille globale du système quand celui-ci est restreint par des murs (ex. ZFA\_001).

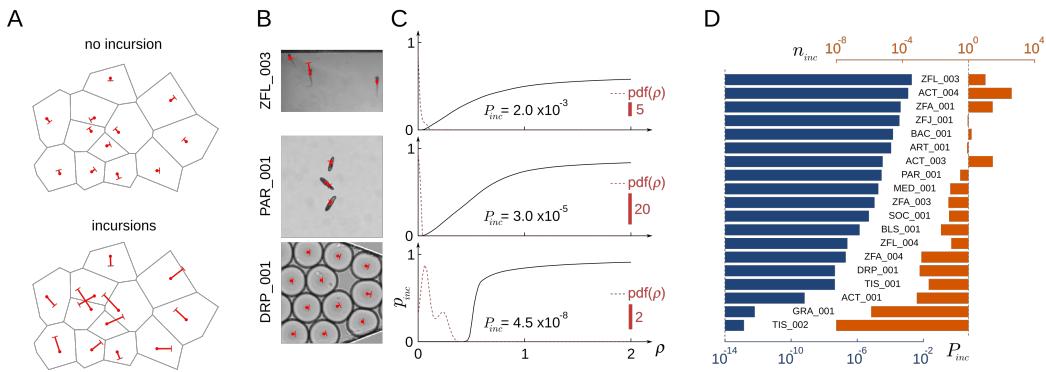
Les distributions de  $\rho$  et  $p_{inc}$  sont représentées figure ?? pour trois films de  $TD^2$  représentatifs de trois systèmes très différents. Il n'est pas nécessaire d'avoir un tracking parfait pour avoir une bonne estimation de cette valeur car les quelques erreurs possibles ont une influence marginale sur la distributions des  $\rho$  donc sur  $P_{inc}$ .

On a calculé  $P_{inc}$  pour tous les films de  $TD^2$  figure ??, de manière à en

estimer le nombre d'incursion  $n_{inc}$  :

$$n_{inc} = P_{inc} * N_{obj}$$

avec  $N_{obj}$  le nombre total d'objet dans les films. Chaque incursion ne résultant pas à une inversion de l'identité de deux objets, donc à une erreur, on gardera en tête que cette valeur représente la borne maximale possible d'erreurs. Dans  $TD^2$ , seulement 5 films ont  $n_{inc} > 1$  ce qui suggère que pour la grande majorité des films, le post-traitement se résume à un simple contrôle. Les 5 films restants sont connus pour leur difficulté à analyser dont 2 (ACT\_003 et ACT\_004) ont nécessités le développement d'un algorithme spécifique, 2 autres (BAC\_001 et ZFA\_001) des logiciels dédiés pour être analysés.



**Figure 4.1 – Probabilité d'incursion** **(A)** Définition des incursions : objets faisant de petits déplacements à l'intérieur de leurs cellules de Voronoï (haut), grands déplacements générant des incusions dans les cellules de Voronoï voisines (bas). De telle incusions sont susceptibles de générer des erreurs de tracking. **(B)** Images extraits de 3 films avec le déplacement instantané des objets (rouge). La position initiale est marqué d'un point. **(C)** Probabilité géométrique d'incusion  $p_{inc}$  en fonction du déplacement normalisé  $\rho$  (noir) et de la densité de probabilité de  $\rho$  (rouge) pour les trois films correspondant. La probabilité d'incusion  $P_{inc}$  correspond à l'aire de recouvrement des deux courbes. **(D)** Probabilité d'incusion  $P_{inc}$  (bleue) et nombre d'incusion  $n_{inc}$  (orange) pour 19 films de  $TD^2$ . Le reste des films ayant  $P_{inc} < 10^{-20}$

### 4.3 Optimisation des paramètres

L'optimisation des paramètres d'analyse peut parfois s'avérer être une tâche délicate, on présentera ici une méthode basée sur la minimisation du nombre d'inversion d'identités (swap). Les étapes suivies pour le calcul sont présentés figure ???. Pour compter le nombre d'inversions, il est seulement nécessaire d'avoir un seul film du système étudié parfaitement analysé, ce qui peut être réalisé par un post-traitement méticuleux. On peut ensuite définir la probabilité d'inversion  $P_{swap}$  et varié la paramètre afin de la minimiser. Pour se faire, on utilisera l'interface en ligne de commande FastTrack\_cli qui permet d'appeler FastTrack directement à l'intérieur d'un script Python pour automatiser la minimisation. On définit la probabilité d'inversion :

$$P_{swap} = \frac{N_{swap}}{N_{obj} - n_{ap}}$$

avec  $N_{swap}$  le nombre total d'inversions,  $N_{obj}$  le nombre total d'objets sur toutes les images, et  $n_{ap}$  le nombre de fois qu'un nouvel objet apparaît. Si le nombre d'objets est constant et noté  $n$ , alors  $n_{ap} = n$  et  $*N_{obj} = nT$  avec  $T$  le nombre d'images dans le film, alors on peut simplifier  $P_{swap}$  :

$$P_{swap} = \frac{N_{swap}}{n(T - 1)}$$

On a étudier en premier lieu l'impact du paramètre  $h_d$ , la distance maximale de déplacement autorisé entre deux images successives. La figure ?? représentera l'évolution de  $P_{swap}$  en fonction de  $h_d$  pour trois films tirés de  $TD^2$ . On voit que pour un faible  $h_d$ ,  $P_{swap}$  est essentiellement donnée par la distribution des déplacements des objets, ce qui s'explique par le fait qu'un grand nombre d'erreurs est générées quand un objet n'est pas autorisé à se déplacer plus qu'on son déplacement typique. Pour des  $h_d$  grands, c'est la distribution des distances aux voisins (défini par la tessellation de Voronoï) qui influence le plus  $P_{swap}$ , algorithme devenant plus sensible aux incursions et pouvant faire plus d'erreurs avec les entrées et sorties du champ de vision. Entre ces deux cas extrêmes, il existe en général un minimum, ce qui est particulièrement évident pour les systèmes denses dont la distribution des distances aux voisins est très piquées. Par exemple pour DRP\_001 on voit que  $P_{swap}$  tombe à 0 pour un intervalle de  $h_d$ . La fréquence d'acquisition joue un rôle important ici, pour des films très résolue temporellement, on aura une

distribution de déplacement décalée à gauche vers des distances plus courtes, cela va donner une séparation claire et de petite valeurs en  $P_{swap}$ . Pour des films avec une résolution temporelle médiocre (ZFJ\_001), les deux distributions se recouvrent et  $P_{swap}$  reste toujours bloquée à de grandes valeurs.

On peut faire une analyse similaire pour les autres paramètres. On a représenté  $h_d$  en fonction de  $h_t$  (nombre d'image où l'objet disparaît autorisé), on voit qu'on peut trouver une bande de paramètres où  $P_{swap}$  est minimale.

Pour résumer, on peut faire cette analyse pour un film parfaitement analysée grâce à FastTrack et à l'outil de post-traitement, puis en dérivé un jeu de paramètre optimaux à appliquer sur des films similaires de manière à réduire la charge de travail en post-traitement.

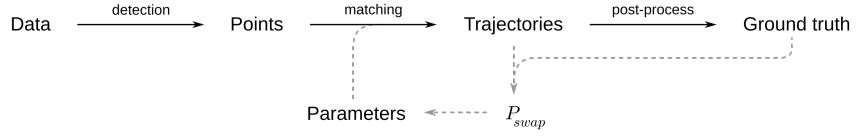
On une autre question souvent posée est de savoir avec quelle résolution temporelle acquérir le film. Une grande résolution permet de diminuer  $P_{swap}$  et donc de réduire le post-traitement mais n'est pas toujours envisageable (limitation de la caméra, de l'éclairage, nécessité d'un grand espace de stockage). Pour étudier l'impact de la fréquence d'enregistrement, on a sélectionné 7 films avec une bonne résolution temporelle et nous les avons dégradés pour à chaque fois calculer  $P_{swap}$ . tracer en fonction du facteur de dégradation adimensionné  $(f - f^*)\tau$ , on voit que tous les  $P_{swap}$  suivent une courbe maîtresse :

$$L(x) = \frac{1}{1 + e^{-1}a}$$

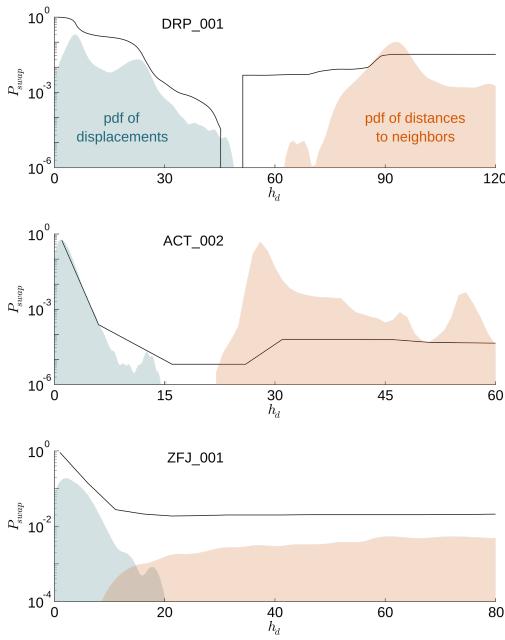
qui est une fonction logistique standard où  $f^*$  correspond au facteur de dégradation où  $P_{swap} = 0.5$ ,  $\tau$  est un facteur de normalisation qui dépend des propriétés dynamiques de chaque système.

A l'aide de ses résultats, on peut donc proposer un méthode permettant de déterminer la fréquence d'acquisition optimale. Cette méthode nécessite un seul enregistrement de bonne qualité déjà analysé avec FastTrack. Il faut ensuite répéter la procédure de dégradation et calculer le  $P_{swap}$  résultant. Les  $P_{swap}$  peuvent être ajusté à la fonction logistique pour en extraire les paramètres  $f^*$  et  $\tau$ . Il suffit ensuite de choisir la fréquence d'acquisition en fonction de la quantité de travail que l'utilisateur est prêt à fournir en post-traitement  $N_{swap} = P_{swap}(N_{obj} - n_{ap})$ .

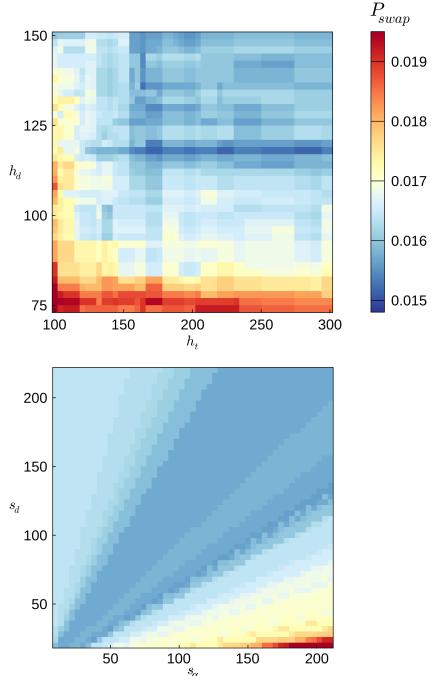
A



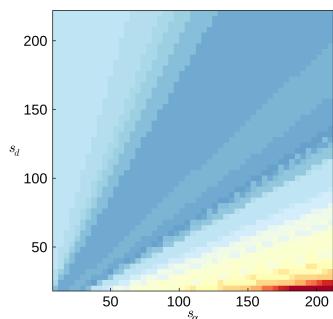
B



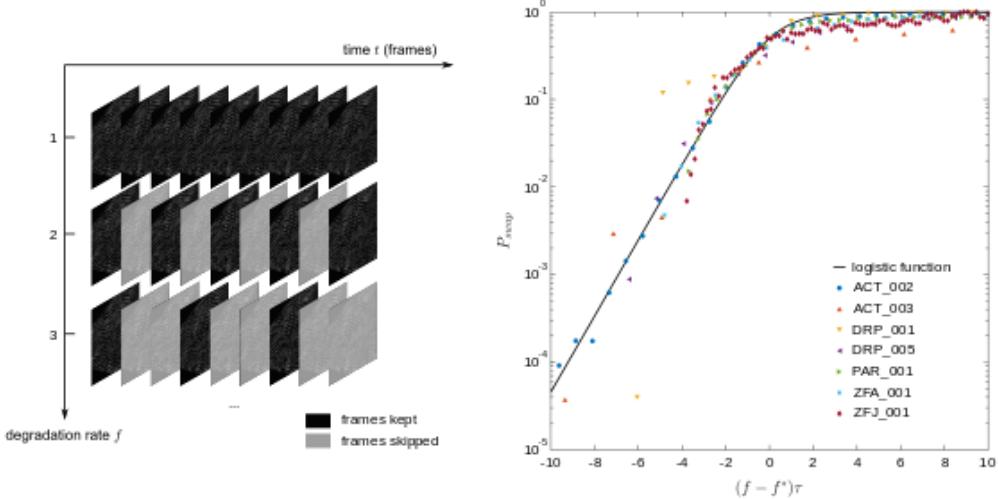
C



D



**Figure 4.2 – Optimisation des paramètres d’analyse basé sur  $P_{swap}$  (A)**  
 Schéma du workflow d’optimisation : le résultat parfait de tracking est utilisée pour calculer  $P_{swap}$  et créer une boucle de rétroaction sur les paramètres d’analyse. (B)  $P_{swap}$  (noire) en fonction de la distance maximale  $h_d$  (en pixels) pour trois films typiques. Les lignes verticales pour DRP\_001 indique que  $P_{swap}$  descend à 0. Les distributions des déplacements entre deux images successives (bleue), et de distance aux voisins (orange) sont montrées pour comparaison. (C)  $P_{swap}$  en fonction de la distance maximale  $h_d$  et du temps de disparition maximal  $h_t$  (en nombre d’images) pour PAR\_001 pour  $s_d = 95$  et  $s_\alpha = 60$ . (D)  $P_{swap}$  en fonction de la distance  $s_d$  (en pixels) et de l’angle de normalisation  $s_\alpha$  (en degrés) pour PAR\_001 avec  $h_d = 210$  et  $h_t = 90$ .



**Figure 4.3 –  $P_{swap}$  et vitesse d’acquisition** **(A)** Schéma de la procédure de dégradation. **(B)**  $P_{swap}$  en fonction du taux réduit de dégradation  $(f - f^*)\tau$  pour 7 systèmes très différents.  $f^*$  et  $\tau$  ont été déterminés par un ajustement sur la fonction logistique standard (noire).

# Chapitre 5

## Perspective

On a vu dans cette partie le logiciel FastTrack. On a détaillé les outils utilisés ainsi que l'implémentation en elle-même. On a ensuite testé le logiciel sur une grande quantité de films présentant des systèmes extrêmement variés et évalué sa performance. On a décrit deux méthodes permettant de trouver facilement les paramètres optimaux d'analyses et la fréquence d'acquisition optimale qui permet de guider l'utilisateur de la conception de l'expérience jusqu'à l'analyse des films.

On a aussi abordé un point central du logiciel qui est son appartenance au mouvement open-source. Les codes du logiciel sont entièrement disponible à l'adresse URL. Cela permet d'une part de collaborer au projet pour régler des bugs où ajouter des fonctionnalité. D'intégrer tout ou une partie du projet dans un logiciel ou une expérience déjà existante. Cela renforce la pérennité du projet qui ne dépend pas d'une seule personne pour rester à jour. Enfin la mise en place de la livraison continue (CD) permet de s'affranchir d'une personne dédié à la création de chaque version pour les trois plateformes supportées et donc de réduire le temps nécessaire à la parution de correctifs.

La collaboration pour implémenter de nouvelles fonctionnalités, participer à la documentation ou régler des bugs est encourager et peut se faire grâce au système de pull-request de GitHub. Le logiciel à d'abord été testé au sein du laboratoire où il a assez vite été adopté. Il compte à cette date (avant publication de l'article) plus de 500 téléchargements et une base d'une dizaine d'utilisateurs quotidiens.

**Deuxième partie**

**Dual**

# Bibliographie

- [1] Airbus. *ATTOL : Autonomous Taxiing, Take-Off and Landing*, 2020. <https://www.airbus.com/newsroom/stories/autonomy-aerial-mobility.html>.
- [2] Y.-X. Bai, S.-H. Zhang, Z. Fan, X.-Y. Liu, X. Zhao, X.-Z. Feng, and M.-Z. Sun. Automatic multiple zebrafish tracking based on improved hog features. *Scientific reports*, 8(1) :1–14, 2018.
- [3] G. Bradski. The OpenCV Library. *Dr. Dobb’s Journal of Software Tools*, 2000.
- [4] A. I. Dell, J. A. Bender, K. Branson, I. D. Couzin, G. G. de Polavieja, L. P. Noldus, A. Pérez-Escudero, P. Perona, A. D. Straw, M. Wikelski, et al. Automated image-based tracking and its application in ecology. *Trends in ecology & evolution*, 29(7) :417–428, 2014.
- [5] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele. Deepcut : A deeper, stronger, and faster multi-person pose estimation model. In *European Conference on Computer Vision*, pages 34–50. Springer, 2016.
- [6] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, X. Zhao, and T.-K. Kim. Multiple object tracking : A literature review. *arXiv preprint arXiv :1409.7618*, 2014.
- [7] A. Mathis, P. Mamidanna, K. M. Cury, T. Abe, V. N. Murthy, M. W. Mathis, and M. Bethge. Deeplabcut : markerless pose estimation of user-defined body parts with deep learning. *Nature neuroscience*, 21(9) :1281–1289, 2018.
- [8] A. Mathis and R. Warren. On the inference speed and video-compression robustness of deeplabcut. *BioRxiv*, page 457242, 2018.

- [9] T. Nath, A. Mathis, A. C. Chen, A. Patel, M. Bethge, and M. W. Mathis. Using deeplabcut for 3d markerless pose estimation across species and behaviors. *Nature protocols*, 14(7) :2152–2176, 2019.
- [10] A. Pérez-Escudero, J. Vicente-Page, R. C. Hinz, S. Arganda, and G. G. De Polavieja. idtracker : tracking individuals in a group by automatic identification of unmarked animals. *Nature methods*, 11(7) :743–748, 2014.
- [11] Z.-M. Qian, S. H. Wang, X. E. Cheng, and Y. Q. Chen. An effective and robust method for tracking multiple fish in video image based on fish head detection. *BMC bioinformatics*, 17(1) :251, 2016.
- [12] A. Rodriguez, H. Zhang, J. Klaminder, T. Brodin, and M. Andersson. Toxid : an efficient algorithm to solve occlusions when tracking multiple animals. *Scientific reports*, 7(1) :1–8, 2017.
- [13] A. Rodriguez, H. Zhang, J. Klaminder, T. Brodin, P. L. Andersson, and M. Andersson. Toxtrac : a fast and robust software for tracking organisms. *Methods in Ecology and Evolution*, 9(3) :460–464, 2018.
- [14] F. Romero-Ferrero, M. G. Bergomi, R. C. Hinz, F. J. Heras, and G. G. de Polavieja. Idtracker. ai : tracking all individuals in small or large collectives of unmarked animals. *Nature methods*, 16(2) :179–182, 2019.
- [15] M. Shahin, M. A. Babar, and L. Zhu. Continuous integration, delivery and deployment : a systematic review on approaches, tools, challenges and practices. *IEEE Access*, 5 :3909–3943, 2017.
- [16] StandardCppFoundation. *Cpp Language*. <https://isocpp.org/>.
- [17] TheQtCompany. *The Qt Framework*, 2020. <https://www.qt.io/>.
- [18] V. Viswanathan and R. Hussein. Applications of image processing and real-time embedded systems in autonomous cars : a short review. *International Journal of Image Processing (IJIP)*, 11(2) :35, 2017.
- [19] A. Wikström et al. Benefits and challenges of continuous integration and delivery : A case study. 2019.
- [20] A. Yilmaz, O. Javed, and M. Shah. Object tracking : A survey. *Acm computing surveys (CSUR)*, 38(4) :13–es, 2006.
- [21] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu. Object detection with deep learning : A review. *IEEE transactions on neural networks and learning systems*, 30(11) :3212–3232, 2019.