Rapport de stage

Estimation de la direction de braquage par vidéo embarquée dans le cadre du projet FastTrack

Licence 3 Informatique

U.F.R. Sciences Fondamentales et Appliquées

Stage du 20 avril au 31 juillet 2026

Année universitaire 2025–2026

Réalisé par
Dmytro Honcharenko

Tuteur de stage
Cédric Joulain

Enseignant référent
Rita Zrour

SERLI
Avenue Thomas Edison
86360 Chasseneuil-du-Poitou

Abstract

This report presents work carried out during an internship at SERLI, focused on developing a system to estimate the yaw rate of a go-kart from onboard video footage, without any physical sensor. Multiple approaches were explored iteratively : a classical prototype using ORB feature detection and homography estimation, an evolution toward deep learning methods (SuperPoint/LightGlue), and a production-ready pipeline based on depth-weighted homography decomposition with MAD outlier rejection and chunked video processing. The work spanned six development phases and produced a pipeline validated against reference telemetry from the Assetto Corsa simulator, achieving a Pearson correlation of 0.934 and a directional accuracy of 81.7% exact (99.6% within one category).

Résumé

Ce rapport présente les travaux réalisés dans le cadre d'un stage au sein de la société SERLI, portant sur le développement d'un système d'estimation de la vitesse de lacet (yaw rate) d'un kart à partir de vidéo embarquée, sans aucun capteur physique. Plusieurs approches ont été explorées et implémentées de manière itérative : un prototype classique basé sur la détection de points caractéristiques ORB et l'estimation d'homographie, une évolution vers des techniques de deep learning (SuperPoint/LightGlue), et un pipeline de production exploitant la décomposition géométrique de l'homographie pondérée par profondeur, avec rejet d'outliers par MAD et traitement par blocs. La progression s'est organisée en six phases de développement, aboutissant à un pipeline validé par comparaison avec la télémétrie de référence du simulateur Assetto Corsa, avec un coefficient de corrélation de Pearson de 0,934 et une précision directionnelle de 81,7 % exacte (99,6 % à une catégorie près).

Remerciements

Je tiens à exprimer ma sincère gratitude à Cédric Joulain, pour son encadrement, sa disponibilité lors de nos réunions hebdomadaires ainsi que la pertinence de ses conseils tout au long du projet. Son soutien et son expertise m'ont permis de progresser et d'aborder cette mission dans les meilleures conditions.

J'adresse également mes remerciements à Jérôme Petit et Orianne Tisseuil pour l'opportunité qu'ils m'ont offerte d'effectuer ce stage au sein de SERLI ainsi que pour la confiance qu'ils m'ont accordée. Cette expérience, menée dans un environnement à la fois stimulant et enrichissant, a constitué une étape particulièrement formatrice dans mon parcours.

Je souhaite également remercier l'ensemble de l'équipe de SERLI pour leur accueil, leur disponibilité et la qualité de l'environnement de travail qu'ils ont su créer au quotidien. Enfin, j'adresse mes remerciements à Rita Zrour, tutrice pédagogique à l'Université de Poitiers, pour son suivi et son accompagnement tout au long de cette mission.

Table des matières

Introduction

Extraire la télémétrie d'une course (vitesse, trajectoire, orientation) suppose habituellement des capteurs coûteux et difficiles à embarquer, qu'il s'agisse d'un kart de club ou d'une voiture de course. Le projet FastTrack, mené au sein de la société SERLI, fait le pari inverse : reconstituer ces données à partir de la seule caméra embarquée, sans aucun matériel additionnel. Le karting sert ici de cas d'application, mais l'approche vise tout véhicule de compétition filmé par une caméra embarquée. Ma mission portait sur un maillon précis de cette chaîne : estimer, image par image, la vitesse à laquelle le véhicule change de cap, sa vitesse de lacet, à partir de la vidéo.

Ce rapport intermédiaire couvre la première phase d'un stage de trois mois (20 avril au 31 juillet 2026), réalisé dans le cadre de la Licence 3 Informatique de l'Université de Poitiers, sous la tutelle pédagogique de Mme Rita Zrour. Il retrace la démarche suivie, de l'analyse du problème à la validation quantitative du pipeline, à travers six itérations de développement, et rend compte de la conduite de projet propre à une recherche appliquée exploratoire.

Présentation de la structure d'accueil

L'entreprise

SERLI est une Entreprise de Services du Numérique (ESN) fondée en 1981 par d'anciens étudiants de l'Université de Poitiers. L'entreprise compte aujourd'hui une cinquantaine de collaborateurs et dispose d'une seconde implantation à Niort, place forte du secteur des assurances et des mutuelles en France.

Son activité principale couvre le conseil et le développement en ingénierie logicielle, avec une expertise historique dans l'écosystème Java et les serveurs d'applications. L'entreprise intervient également sur des sujets de cloud computing, d'intelligence artificielle appliquée et de formations techniques. Elle se définit comme un organisme de formation en plus de sa dimension de conseil, organisant des événements internes tels que des coding dojos, des conférences techniques et des ateliers d'innovation.

L'organisation interne est structurée autour d'équipes projet de petite taille, dans un esprit de proximité et de collaboration directe entre les collaborateurs. Le projet FastTrack, dans lequel s'inscrit ce stage, relève du pôle Data & IA de la structure. Encore en phase de recherche active, il s'inscrit dans un écosystème de projets internes complémentaires. LispTick, un autre projet SERLI, vise à permettre le traitement de flux vidéo en streaming plutôt que par chargement complet du fichier. Une intégration future avec LispTick accélérerait sensiblement le pipeline développé durant ce stage, en supprimant le principal goulot d'étranglement lié au chargement des vidéos longues.

Mon poste

Mon poste officiel dans la convention de stage est celui de développeur full-stack. En pratique, l'essentiel de la mission a été consacré au traitement d'images et à la vision par ordinateur, ce qui reflète la nature exploratoire et transverse du projet FastTrack. J'ai travaillé de façon largement autonome : l'ensemble des décisions d'implémentation et

d'architecture ont été prises de ma propre initiative. Cédric Joulain intervenait comme interlocuteur principal lors de réunions hebdomadaires, permettant de valider les orientations et de débloquer les difficultés rencontrées.

Contexte de la mission

Problématique

La télémétrie de karting (mesures de vitesse, trajectoire, accélération, direction) permet aux pilotes et entraîneurs d'analyser finement les performances lors de séances d'entraînement ou de compétition. Les systèmes professionnels s'appuient sur des capteurs coûteux (IMU, GPS différentiel, capteur d'angle de volant) difficiles à intégrer sur des karts de club. Au-delà du coût, ces capteurs présentent des limites techniques importantes dans ce contexte. Le GPS grand public présente une marge d'erreur de positionnement de l'ordre de 10 m, supérieure à la largeur courante d'une piste de karting (environ 8 m) : il ne permet donc même pas de situer le kart sur la piste, et encore moins de restituer ses changements de direction rapides. L'IMU accumule une dérive d'intégration sur la durée d'une session. Les vibrations du moteur et de la route contaminent les lectures angulaires. La synchronisation entre capteurs opérant à des fréquences différentes constitue un défi supplémentaire sans équipement dédié.

FastTrack propose une approche radicalement différente : extraire cette télémétrie directement depuis une caméra embarquée grand-public (ex. GoPro), sans aucun matériel additionnel. La mission confiée portait sur l'estimation de la vitesse de rotation du kart dans le plan horizontal (sa vitesse de lacet), grandeur clé pour caractériser l'engagement dans les virages et évaluer la précision de la trajectoire d'un pilote.

Un travail préalable réalisé par un étudiant en alternance avait fourni un point de départ documentaire. À la lecture de ce rapport, l'approche SuperPoint/LightGlue s'est dégagée comme la piste la plus prometteuse, ce qui a orienté les premières décisions techniques du stage.

Défis techniques

L'estimation du yaw à partir d'une vidéo présente plusieurs difficultés fondamentales :

— **Absence de stéréoscopie :** avec une seule caméra, l'échelle absolue est irrécouvrable, seul le déplacement angulaire relatif est estimable.

— **Distorsion de la lentille grand-angle :** les caméras d'action présentent une distorsion en barillet significative qui biaise l'estimation géométrique.

— **Scène dynamique :** autres karts, spectateurs, ombres introduisent des mouvements non-rigides perturbant l'appariement de points caractéristiques.

— **Conditions lumineuses difficiles :** reflets spéculaires, surexposition (éblouissement solaire sur asphalte mouillé), changements rapides d'exposition.

— **Vibrations et trépidations :** le kart génère des vibrations haute-fréquence importantes qui dégradent la qualité des images.

— **Ambiguïté parallaxe / changement de cap :** les objets proches semblent se déplacer à cause de la translation vers l'avant, non d'un changement de direction. Seules les zones lointaines encodent fidèlement le yaw.

Cahier des charges

Objectifs initiaux

Le cahier des charges n'existait pas formellement avant mon arrivée. Il a été co-construit au cours des premières semaines du stage, à partir des échanges avec Cédric Joulain et du rapport de l'alternant précédent. Les objectifs définis à l'issue de cette phase de cadrage étaient les suivants :

— Développer un algorithme capable d'estimer, image par image, la vitesse de lacet du kart (yaw rate), c'est-à-dire la vitesse à laquelle il change de cap, exprimée en degrés par seconde, à partir d'une vidéo embarquée de karting.

— Fonctionner sans capteur physique externe (pas de GPS, IMU, ni capteur d'angle de volant).

— Être compatible avec des caméras grand-public de type GoPro (résolution HD/4K, objectif grand-angle).

— Produire une estimation lissée et cohérente temporellement, sans sauts aberrants.

— Permettre la comparaison avec des données de télémétrie de référence pour validation quantitative.

Contraintes

— **Matérielles :** traitement sur machine standard (16–32 Go RAM), GPU recommandé pour les phases deep learning.

— **Logicielles :** rester dans un écosystème Python open source, sans dépendance propriétaire ni licence payante ; réutiliser LightGlue depuis son dépôt officiel plutôt que de réimplémenter l'appariement.

— **Temporelles :** production d'une version fonctionnelle et validée avant la fin du stage (31 juillet 2026).

— **Volumétriques :** le pipeline doit pouvoir traiter des vidéos de plus de 30 minutes à 30–60 fps (jusqu'à 100 000 images).

Livrables

— Scripts Python organisés et versionnés sur GitHub.

— Fichiers de configuration YAML paramétrables permettant l'adaptation sans modifier le code.

— Script de validation par comparaison avec télémétrie.

— Rapport de stage documentant la progression et les résultats.

Travail réalisé

Le travail s'est organisé en six phases itératives, chacune apportant des améliorations substantielles au pipeline précédent.

Phase 1. Prototype classique (ORB + Homographie)

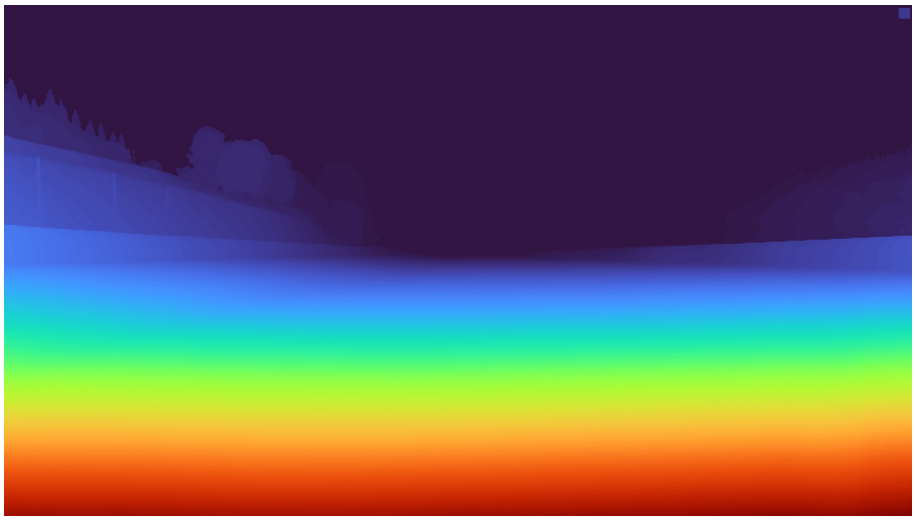
La zone d'analyse est limitée aux 30 % inférieurs de l'image, filtrés par seuil de couleur HSV pour ne retenir que les pixels de l'asphalte.

L'estimation de l'angle repose sur une homographie ORB [6] entre paires de frames consécutives, avec un schéma de pondération empirique 40/40/20 appliqué à trois signaux extraits de la matrice H : le cisaillement horizontal, la translation normalisée, et l'angle de rotation. Une fallback par flot optique dense Lucas-Kanade est déclenchée si l'homographie échoue (trop peu d'inliers RANSAC).

Cette première version, bien que fonctionnelle, s'est révélée fragile face aux changements d'éclairage et aux textures répétitives de la route.

Phase 2. Masquage par profondeur

Cette phase introduit un masquage par profondeur estimée via un modèle de profondeur (Distill-Any-Depth [3], HuggingFace Transformers). La carte de profondeur est normalisée et deux seuils percentiles sont appliqués : les pixels les plus proches (zone kart, sol immédiat) et les plus lointains (ciel, nuages) sont exclus, de sorte que seule la zone intermédiaire, piste et barrières à distance utile est conservée pour l'analyse. Cela améliore la robustesse face aux éléments non pertinents de la scène.



Carte de profondeur estimée (Distill-Any-Depth)

Zone d'analyse après masquage

FIGURE 1 – Estimation de profondeur monoculaire et masquage de la zone d'intérêt

Phase 3. Cadre de comparaison deux tours

Cette phase change de paradigme : plutôt qu'estimer une valeur absolue image par image, on compare deux tours du même circuit. Les vidéos sont synchronisées temporellement et les différences de vitesse de lacet calculées point par point. L'objectif est concret : permettre à un entraîneur d'identifier, virage par virage, si un pilote engage sa trajectoire plus tôt ou plus tard que lors d'un tour de référence, information directement exploitable en coaching sans nécessiter de valeur absolue de yaw.

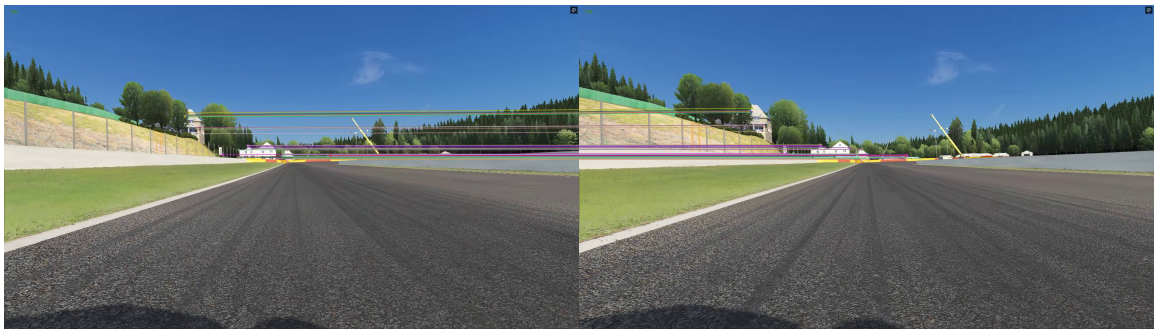
Cette comparaison deux tours suppose toutefois une estimation par tour suffisamment stable. Elle est donc pour l'instant mise en pause, et sera reprise une fois le pipeline mono-vidéo fiabilisé, objectif prioritaire des phases suivantes.

Phase 4. Pipeline Deep Learning (SuperPoint/LightGlue)

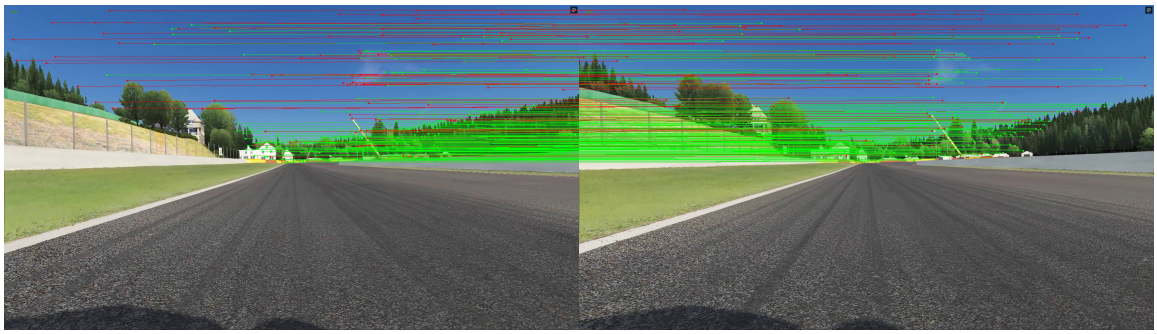
Un changement majeur est opéré dans la détection et l'appariement de points : ORB est remplacé par `SuperPoint` [1] (extracteur de keypoints neuronal) associé à `LightGlue` [2]

(apparieur par mécanisme d'attention de type Transformer). Ces modèles offrent des correspondances beaucoup plus robustes aux flous de mouvement, aux changements d'éclairage et aux variations de point de vue au prix d'une exigence GPU et d'une latence plus élevée (∼1–5 fps vs. temps réel pour ORB).

Des mécanismes complémentaires sont ajoutés : filtrage des keypoints par percentile de profondeur (cf. Phase 5), et validation spatiale des clusters de correspondances sur une grille.



Appariements ORB (post-RANSAC)



Appariements SuperPoint/LightGlue

FIGURE 2 – Comparaison visuelle des appariements ORB et SuperPoint/LightGlue sur deux frames consécutives

Phase 5. Décomposition géométrique de l'homographie et configuration adaptative

Le calcul de l'angle est refondé géométriquement. Plutôt qu'extraire le yaw par heuristique (pondération 40/40/20), la matrice d'homographie H [5] est décomposée en candidats de rotation, dont le plus cohérent avec les points visibles est sélectionné automatiquement. Les correspondances sont pondérées par profondeur estimée avant la décomposition, favorisant les zones lointaines moins affectées par la parallaxe.

Une fenêtre adaptative est également introduite : si la confiance de l'estimation est insuffisante avec une fenêtre de 10 frames, le calcul est relancé sur 20 frames. Si la confiance reste faible, la valeur est retournée comme NaN.

Phase 6. Pipeline de production

La phase finale rassemble l'ensemble des avancées dans un pipeline unifié, robuste et orienté production. Les principales améliorations apportées sont :

— **Traitement par blocs :** division de la vidéo en segments temporels traités séquentiellement, avec stockage temporaire sur disque, pour rendre le traitement viable sur des vidéos de 30+ minutes sans saturer la RAM.

— **Détection automatique des phases d'arrêt :** lorsque le kart est immobile (vitesse estimée nulle), la valeur de yaw est remplacée par NaN, évitant que les estimations parasites sur fond statique ne contaminent le signal final.

— **Rejet des outliers par MAD :** les rotations physiquement impossibles sont détectées et supprimées avant le lissage.

— **Estimation multi-paires :** le yaw est estimé sur plusieurs paires de frames consécutives et la médiane est retenue, réduisant la sensibilité aux mauvaises frames isolées.

— **Lissage :** un filtre gaussien appliqué après rejet des outliers a été testé en complément du filtre médian. Un problème dans le script de comparaison rend les métriques de cette variante ininterprétables ; seul le filtre médian est donc retenu pour les résultats publiés ci-dessous.

— **Configuration YAML :** l'ensemble des hyperparamètres du pipeline est externalisé dans un fichier de configuration (taille des blocs, paramètres de filtrage, seuils de rejet).

Validation : comparaison avec la télémétrie

Un script de validation indépendant a été développé pour évaluer quantitativement l'estimation en la comparant à des données issues du moteur physique d'Assetto Corsa, simulateur de course utilisé pour capturer une télémétrie de référence à 10 Hz. Cette fréquence a été retenue comme compromis optimal entre précision temporelle et niveau de bruit, et se prête directement à la comparaison avec les 60 fps de la vidéo après rééchantillonnage. La méthodologie comprend une synchronisation temporelle automatique par corrélation croisée, un rééchantillonnage sur une base de temps commune, puis le calcul de métriques standard.

Les résultats obtenus sur une session de 10 922 échantillons (durée d'overlap : 1 092 s) sont présentés dans le tableau 1.

Table 1 – Résultats de validation par comparaison avec la télémétrie

Métrique	**Valeur**
Corrélation de Pearson	0,934
MAE	3,20 deg/s
RMSE	5,01 deg/s
Biais moyen	+0,385 deg/s
Précision directionnelle (exacte)	81,7 %
Précision directionnelle (à une catégorie près)	99,6 %

Sur cette session, l'estimation suit fidèlement la forme du signal de référence : la corrélation de Pearson atteint 0,934 et le sens du virage (gauche, droite, tout droit) est correctement

classé dans 81,7 % des cas exactement, et dans 99,6 % des cas à une catégorie près. Le biais moyen de +0,385 deg/s traduit une légère dérive systématique vers la droite, dont l'origine reste à identifier. Les erreurs résiduelles se concentrent sur les virages serrés et rapides, là où la distorsion de l'objectif et la saturation des keypoints dégradent le plus l'appariement.

Ces chiffres doivent toutefois être lus pour ce qu'ils sont. Les premières itérations ont été éprouvées sur de vraies vidéos de karting, mais aucune télémétrie image par image n'était disponible pour ces séquences : il était donc impossible de vérifier quantitativement l'estimation sur données réelles. Le simulateur Assetto Corsa a été retenu précisément pour cela, car il fournit une vérité terrain propre et synchronisable, là où les capteurs physiques (GPS, IMU) sont eux-mêmes fortement bruités et constitueraient une référence peu fiable. La validation présentée ici porte donc sur une vidéo et une télémétrie toutes deux issues du simulateur : elle établit la viabilité de la méthode en conditions contrôlées, mais sa robustesse sur image réelle (distorsion grand-angle, reflets, vibrations) reste à éprouver.

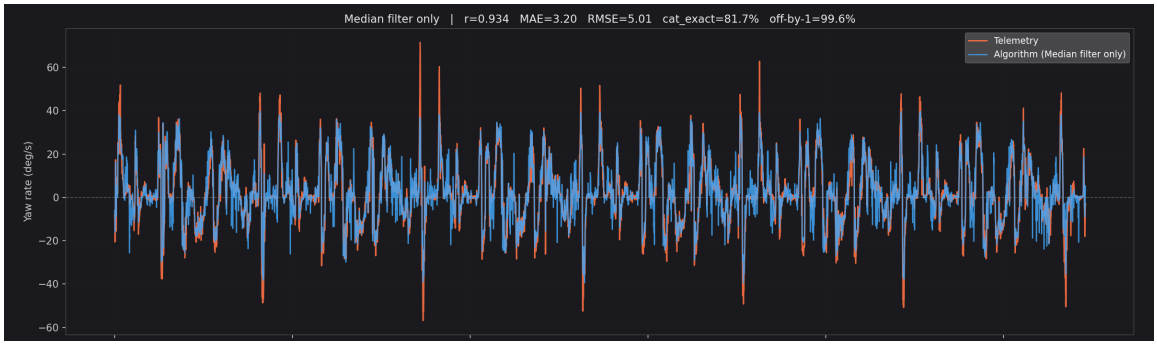


FIGURE 3 – Vitesse de lacet estimée (deg/s, filtre médian) vs données de référence sur la session complète (∼18 min)

Technologies et ressources utilisées

L'ensemble du pipeline repose sur Python 3, avec OpenCV [4] pour les opérations de vision par ordinateur, NumPy et SciPy pour le traitement numérique et le filtrage, et PyTorch comme moteur d'inférence GPU. La détection et l'appariement de points caractéristiques s'appuient sur SuperPoint et LightGlue, installés depuis leur dépôt GitHub officiel. L'estimation de profondeur monoculaire utilise le modèle Distill-Any-Depth via HuggingFace Transformers. La configuration est gérée par fichiers YAML (PyYAML) et les diagnostics produits avec Matplotlib. Le développement a été réalisé sous VS Code, avec versionnement Git sur GitHub.

Les phases classiques (1–3) sont légères et fonctionnent sans GPU, avec une empreinte mémoire faible dépendant de la résolution de la vidéo. Les phases deep learning (4–6) nécessitent un GPU recommandé pour une vitesse acceptable (∼1–5 fps), le traitement par blocs (chunked processing) maintient la consommation mémoire à un niveau raisonnable indépendamment de la durée de la vidéo.

Table 2 – Évolution des techniques entre les phases classiques et hybrides

Aspect	**Phases 1–3 (Classique)**	**Phases 4–6 (Hybride / DL)**
Détection de points	ORB + BFMatcher	SuperPoint + LightGlue
Masquage de scène	Seuil HSV + ROI	Profondeur + ROI + détection éblouissement
Estimation angle	Homographie (heuristique 40/40/20)	Décomposition $H \to R \to$ yaw
Robustesse	Basique	MAD, multi-paires, détection arrêt
GPU requis	Non	Oui (recommandé)
Vitesse de traitement	Temps réel ou plus rapide	$\sim$1–5 fps

Gestion de projet

Organisation et planification

Le projet a été conduit selon une méthodologie itérative et incrémentale. Chaque phase représente un cycle complet de conception, implémentation, test et analyse des résultats, orientant les décisions de la phase suivante. L'ensemble des décisions techniques est tracé dans l'historique Git de la branche de travail.

Planning rétrospectif

Le tableau 3 présente le planning rétrospectif du stage, comparant la durée prévue initialement pour chaque bloc de travail avec la durée effectivement consacrée. Plusieurs phases ont été menées en parallèle : les trois variantes classiques (phases 1 à 3) lors des premiers jours, puis la refonte géométrique du calcul d'angle (phase 5) en marge de la mise en place du pipeline deep learning (phase 4). L'essentiel du temps a été absorbé par les deux blocs les plus lourds : l'intégration de SuperPoint/LightGlue, puis le pipeline de production et sa validation. La rédaction de ce rapport intermédiaire a débuté en parallèle de la finalisation de cette dernière phase.

Table 3 – Planning rétrospectif du stage

Période	**Activité**	**Durée prévue**	**Durée réelle**
20/04 – 21/04	Prise en main, lecture du rapport de l'alternant	2 jours	2 jours
22/04 – 24/04	Phase 1 : Prototype classique (ORB)	3 jours	3 jours
22/04 – 24/04	Phase 2 : Masquage par profondeur	3 jours	3 jours
22/04 – 24/04	Phase 3 : Comparaison deux tours	3 jours	3 jours
27/04 – 11/05	Phase 4 : SuperPoint/LightGlue	2 semaines	2 semaines
27/04 – 30/04	Phase 5 : Décomposition homographie + YAML adaptatif	4 jours	4 jours
11/05 – 25/05	Phase 6 : Pipeline de production + validation	2 semaines	2 semaines
25/05 – 29/05	Rédaction du rapport (intermédiaire)	1 semaine	5 jours

Répartition des tâches

L'ensemble du travail sur le pipeline de vision par ordinateur a été réalisé individuellement, sans répartition entre plusieurs contributeurs.

Suivi et réunions

Les réunions de suivi hebdomadaires réunissaient également les autres stagiaires et alternants de l'équipe, offrant un cadre d'échange plus large sur l'avancement de chacun. Une collaboration ponctuelle s'est engagée avec l'un des alternants sur la question de la comparaison entre les données algorithmiques et la télémétrie issue du simulateur, notamment pour déterminer la fréquence d'échantillonnage optimale.

Au-delà des réunions de suivi, les interactions au sein de l'entreprise ont été régulières. Orianne Tisseuil a été sollicitée pour des questions administratives et légales liées au déroulement du stage. Des échanges ont également eu lieu avec le responsable de la communication ainsi qu'avec l'ensemble des collaborateurs présents dans les locaux. Cédric Joulain, en tant que tuteur et interlocuteur principal, jouait un rôle de client interne, validant les orientations techniques et priorisant les axes de développement.

Difficultés rencontrées et solutions

Table 4 – Difficultés rencontrées et solutions apportées

Difficulté	**Impact**	**Solution apportée**
Distorsion grand-angle	Sous-estimation dans les virages serrés	Fenêtre d'analyse adaptative : extension automatique de la fenêtre de calcul pour compenser la perte de précision dans les virages serrés
Faux mouvements (parallaxe)	Surestimation du yaw en ligne droite	Pondération par profondeur, restriction champ lointain
Vidéos longues saturant la RAM	Crash sur vidéos >15 min	Traitement chunked avec stockage disque
Rotations physiquement impossibles	Signal incohérent	Rejet par MAD
Éblouissement solaire (glare)	Perte massive de keypoints	Détection automatique et masquage

Objectifs atteints

À l'issue de cette première phase, les objectifs initiaux sont atteints, à une réserve près : la compatibilité avec les caméras réelles de type GoPro reste à éprouver sur le terrain.

— Pipeline fonctionnel d'estimation du yaw depuis vidéo embarquée.

— Fonctionnement sans capteur physique.

— Compatibilité d'entrée avec des flux de type GoPro (HD/4K, objectif grand-angle).

— Signal lissé et cohérent temporellement.

— Validation quantitative par comparaison avec données de référence (Pearson $r = 0{,}934$).

— Traitement de vidéos longues (>30 min) grâce au chunked processing (contrainte volumétrique satisfaite).

Plus-value pour la structure

La mission apporte à SERLI un module de traitement vidéo fonctionnel et validé pour le projet FastTrack, avec une documentation technique détaillée de la progression algorithmique. Ce composant constitue une brique réutilisable pour les développements futurs du projet, notamment l'intégration dans une application de coaching destinée aux pilotes de karting. L'approche itérative en six phases a également produit une documentation comparative des conditions de performance de chaque méthode, constituant un référentiel méthodologique directement réutilisable pour la suite du projet.

Bilan personnel et professionnel

Compétences techniques acquises

Ce stage a été ma première expérience pratique approfondie en vision par ordinateur. Le cursus de Licence 3 Informatique à Poitiers couvre notamment l'algorithmique, la programmation (impérative, orientée objet, fonctionnelle) et les mathématiques. Ces bases ont été indispensables, en particulier l'algèbre linéaire pour comprendre les transformations géométriques et l'algorithmique pour concevoir des pipelines efficaces, mais la vision par ordinateur n'y est pas enseignée directement.

Tout le volet CV a donc été appris en autonomie pendant le stage :

— **Vision par ordinateur :** flot optique, homographie, matrice essentielle, calibration de caméra.
— **Deep learning appliqué :** utilisation de modèles pré-entraînés (SuperPoint, LightGlue, Distill-Any-Depth) via PyTorch et HuggingFace Transformers.
— **Vision monoculaire** et reconstruction de mouvement à partir de séquences d'images.
— **Conception de pipelines vidéo** robustes et efficaces en mémoire (chunked processing).
— **Démarche de recherche appliquée :** itération rapide, analyse des échecs, documentation des décisions techniques.

Apport de la formation

Bien que la vision par ordinateur ne soit pas au programme de la L3, plusieurs enseignements se sont révélés directement utiles :

— **Mathématiques (algèbre linéaire) :** compréhension des matrices de rotation, des projections homogènes, de la décomposition SVD utilisée pour le raffinement pondéré de l'homographie, et de la décomposition géométrique de H en rotation R.
— **Algorithmique :** conception des structures de traitement par blocs, des mécanismes de détection d'outliers, et de l'optimisation des parcours de données.

Bilan professionnel

Ce stage m'a permis de découvrir le fonctionnement d'une entreprise de conseil IT à taille humaine, avec ses réunions d'équipe régulières et son organisation par projets. Travailler seul sur un problème ouvert, sans solution évidente au départ, m'a appris à structurer ma réflexion, à prioriser les pistes, et à accepter que certaines approches prometteuses s'avèrent décevantes en pratique, en tirant des enseignements constructifs plutôt que de les subir comme des échecs.

Conclusion

Ce stage a démontré la faisabilité de l'estimation du yaw de kart à partir de la seule vidéo embarquée. La progression en six phases, allant d'un prototype classique léger jusqu'à un

pipeline hybride deep learning et vision monoculaire, illustre la complexité du problème et la nécessité d'une approche itérative rigoureuse.

En conditions de validation contrôlées, l'estimation reproduit la dynamique de référence avec une corrélation de 0,934 et une classification correcte du sens du virage dans 81,7 % des cas exactement (99,6 % à une catégorie près), de quoi justifier une validation sur données terrain avant toute intégration dans FastTrack. La limite principale demeure les virages les plus rapides, où la distorsion résiduelle et la saturation des keypoints produisent les écarts les plus importants. La méthode étant validée en simulation, l'enjeu se déplace désormais vers son comportement sur données réelles et son intégration dans le produit.

Perspectives

Plusieurs prolongements se dégagent de ce travail, à court comme à plus long terme :

— **Estimation de la vitesse longitudinale** en complément du yaw, pour une reconstruction complète de la trajectoire 2D sans capteur.

— **Entraînement d'un modèle bout-en-bout** (vidéo → yaw) sur un dataset de courses annoté, pour dépasser les limites de l'approche géométrique.

— **Intégration dans l'application FastTrack** pour utilisation par les pilotes en analyse post-session.

— **Amélioration du traitement en temps réel :** optimisation du pipeline pour approcher le temps réel sans GPU dédié.

Bibliographie

[1] *SuperPoint* — extracteur de points caractéristiques neuronal (inclus dans LightGlue). https://github.com/cvg/LightGlue.

[2] *LightGlue* — appariteur de points par mécanisme d'attention. https://github.com/cvg/LightGlue.

[3] *Distill-Any-Depth* — modèle d'estimation de profondeur monoculaire. GitHub : https://github.com/Westlake-AGI-Lab/Distill-Any-Depth. HuggingFace : https://huggingface.co/xingyang1/Distill-Any-Depth.

[4] *OpenCV* — bibliothèque de vision par ordinateur open source. https://opencv.org.

[5] OpenCV contributors. *Homography tutorial : Basic concepts and decomposition.* https://docs.opencv.org/4.x/d9/dab/tutorial_homography.html.

[6] OpenCV contributors. *ORB (Oriented FAST and Rotated BRIEF).* https://docs.opencv.org/4.x/d1/d89/tutorial_py_orb.html.

Glossaire

1. **Yaw (lacet).** Rotation du kart autour de l'axe vertical. On en mesure ici la vitesse (vitesse de lacet), c'est-à-dire le nombre de degrés de cap gagnés par seconde ; positive vers la droite, négative vers la gauche.

2. **ORB.** Oriented FAST and Rotated BRIEF — détecteur/descripteur de points caractéristiques classique.
3. **SuperPoint.** Réseau convolutif d'extraction de keypoints entraîné de façon auto-supervisée.
4. **LightGlue.** Réseau d'appariement de points basé sur l'attention (Transformer), successeur de SuperGlue.
5. **Homographie.** Transformation projective 3×3 liant deux vues d'une même surface plane.
6. **Distill-Any-Depth.** Modèle d'estimation de profondeur monoculaire par distillation, utilisé pour générer les masques de scène.
7. **MAE.** Mean Absolute Error (Erreur Absolue Moyenne) — moyenne des valeurs absolues des écarts entre l'estimation et la référence.
8. **MAD.** Median Absolute Deviation — statistique robuste pour la détection d'outliers.
9. **RANSAC.** Random Sample Consensus — algorithme d'estimation robuste rejetant les outliers.
10. **Keypoint.** Point caractéristique détecté dans une image, utilisé pour l'appariement entre frames.
11. **Corrélation croisée.** Mesure de similarité entre deux signaux en fonction d'un décalage temporel — utilisée ici pour synchroniser automatiquement l'estimation algorithmique et la télémétrie de référence.

Annexes

Annexe A : Extraits de code clés

A.1 : Extraction du yaw par décomposition géométrique de l'homographie

```python
def rotation_from_homography(H, K, pa_inliers, pb_inliers):
    # n_sol : nombre de solutions valides (2 ou 4 en general)
    # Rs : matrices de rotation candidates (3x3 chacune)
    # ts : vecteurs de translation (direction seulement, echelle inconnue)
    # ns : normales au plan de reference (utilises par le filtre OpenCV)
    n_sol, Rs, ts, ns = cv2.decomposeHomographyMat(H, K)
    if n_sol == 0:
        return 0.0

    # Filtrage des candidats incoherents avec les points visibles
    # reshape(N,1,2) : format attendu par OpenCV (N points, 1 canal, 2 coords)
    pts1 = pa_inliers.reshape(-1, 1, 2).astype(np.float32)
    pts2 = pb_inliers.reshape(-1, 1, 2).astype(np.float32)
    kept = cv2.filterHomographyDecompByVisibleRefpoints(Rs, ns, pts1, pts2)

    # Extraction de l'angle de lacet depuis la matrice R retenue
    R = Rs[kept[0]]
    # R[0,2] = composante x, R[2,2] = composante z -> rotation horizontale
    return -math.degrees(math.atan2(R[0, 2], R[2, 2]))
```

A.2 : Rejet des outliers par MAD

```python
def reject_outliers_mad(angles):
    """Supprime les angles aberrants via MAD avant lissage."""
    valid = ~np.isnan(angles)
    med = np.median(angles[valid])
    mad = np.median(np.abs(angles[valid] - med))
    sigma_est = mad * 1.4826    # MAD -> estimateur de sigma gaussien
    outlier = np.abs(angles - med) > 3.0 * sigma_est
    angles[outlier & valid] = np.nan  # remplis par interpolation ensuite
    return angles
```

A.3 : Affinement de l'homographie par pondération profondeur (SVD pondéré)

```
def weighted_homography(pa, pb, weights):
    """Estime H en donnant plus de poids aux correspondances lointaines.
    Les poids (1 - profondeur_normalisee) favorisent les zones eloignees,
    moins affectees par la parallaxe de translation."""
    A = []
    W = []
    for (x, y), (xp, yp), wi in zip(pa, pb, weights):
        # Deux lignes du systeme Ah=0 par correspondance (x,y) <-> (xp, yp)
        A.append([-x, -y, -1,  0,  0,  0, x*xp, y*xp, xp])
        A.append([ 0,  0,  0, -x, -y, -1, x*yp, y*yp, yp])
        W.extend([wi, wi])
    A = np.asarray(A)
    W = np.asarray(W).reshape(-1, 1)
    _, _, Vt = np.linalg.svd(A * W)  # SVD du systeme pondere
    H = Vt[-1].reshape(3, 3)         # solution = dernier vecteur singulier
    return H / H[2, 2]               # normalisation homogene
```