



Bordeaux INP - ENSEIRB MATMECA
Robotique et Apprentissage

Projet Robotique Autonome

travail réalisé par

Antonin de Bouter & Nicolas Cornet

État de l'art : Expérience VR Immersive avec le
Megabot

Année universitaire 2025 - 2026

Encadrement

Julien ALLALI
Grégoire PASSAULT

Résumé

Le Megabot est un robot quadrupède-véhicule de l'équipe Rhoban développé à Eirlab. Son but est de permettre le transport d'une personne tout en exploitant les capacités de la locomotion à pattes. La plateforme d'environ trois cents kilogrammes est actionnée par des vérins électriques et une architecture parallèle. Le contrôleur de marche est encore en cours de développement, mais l'architecture mécatronique est déjà opérationnelle.

Le but premier de ce projet est de créer une démonstration des capacités mécaniques du Megabot via une expérience en réalité virtuelle (abrégé en RV ou bien VR en anglais) immersive. L'utilisateur doté d'un casque VR serait assis sur le siège de la plateforme centrale du Megabot pendant que celui ci reproduirait à l'utilisateur les sensations de l'expérience VR. Cet état de l'art a pour but d'explorer les pistes possibles pour exploiter au mieux les capacités mécaniques du Megabot pour obtenir l'expérience la plus immersive possible. Cela inclut d'abord une recherche sur l'actualité de la simulation VR de manège que ce soit par vidéo 360 ou bien par simulation complète. La deuxième partie davantage expérimentale concerne la question de comment maximiser les sensations que ce soit par les mouvements du Megabot ou bien par des dispositifs externes au Megabot qui pourrait compléter l'expérience. Une dernière partie s'intéresse au contrôle du Megabot. Le contrôle a déjà été implémenté, cependant, il est intéressant d'étudier des alternatives pour élaborer des pistes d'améliorations.

Mots-clés : *Simulateur, Motion Cueing Algorithm, Réalité virtuelle, sensations fortes.*

Table des matières

Table des figures	vii
1. Extraction de mouvement et simulation VR	3
1.1. Extraction de données télémétriques depuis des vidéos en réalité virtuelles	3
1.1.1. Données télémétriques en métadonnées parallèles à la vidéo	3
1.1.2. Traitement du flux vidéo en temps réelle	4
1.2. Récupération des données télémétriques depuis une simulation dynamique	4
1.2.1. Interface de simulation externe	5
1.2.2. Simulateur intégré au programme de traitement mécanique du Megabot	5
2. Immersion et génération de mouvement	7
2.1. Étude du champs de perception humain	7
2.1.1. L’audiovisuel	7
2.1.2. Oreille interne	7
2.2. Adaptation du mouvement à un simulateur	8
2.2.1. Algorithmes de transformation	8
2.2.2. Algorithme classic washout	10
2.2.3. Maximisation des sensation fortes	11
3. Commande et contrôle du Megabot	13
3.1. Commande	13
3.1.1. Asservissement	13
3.1.2. Comportement aux extrêmes	14
3.2. Cinématique	14
3.2.1. Cinématique Directe	14
3.2.2. Cinématique Inverse	15
Conclusion & perspectives	17
A. Annexe	19
Bibliographie	22

Table des figures

2.1. Classical washout scheme (Reid-Nahon UTIAS implementation) [11]	11
A.1. Schéma des recherches effectués pour le problème donné	20

Introduction

Pour faire des recherches pertinentes sur le projet nous avons choisi d'orienter notre état de l'art autour de trois axes : la simulation VR, l'adaptation des retours et la commande du Megabot. Ainsi, malgré une idée assez claire de l'objectif et des techniques qui pourraient nous y mener nous avons fait un état de l'art avec deux perspectives : se renseigner sur l'exécution de techniques avec lesquelles nous n'étions pas familiers et l'exploration de pistes alternatives afin de vérifier que notre solution est parmi les solutions optimales à notre problème.

1. Extraction de mouvement et simulation VR

Il existe plusieurs options pour simuler une expérience immersive en réalité virtuelle. Le projet a pour but de créer une démonstration récréative et accessible à tous. Il a été choisi de reproduire avec le Megabot l'expérience d'être dans un roller coaster, c'est à dire un manège des montagnes Russes en français. Cette attraction se fait sans aucune interaction de la part de l'utilisateur. Ce qui signifie que l'expérience VR ne provient pas obligatoirement d'une simulation de montagne russe dynamique, elle peut être adaptée d'une vidéo 360 comme il est possible de trouver aisément sur internet.

1.1. Extraction de données télémétriques depuis des vidéos en réalité virtuelles

Le traitement d'une vidéo comparé à un flux vidéo en temps réel, est une distinction non négligeable. En effet le traitement vidéo est en général plus aisé puisqu'il permet un traitement d'image en différé. C'est à dire un traitement a priori qui n'est pas obligatoirement en temps réel, autorisant alors un traitement non contraint par le temps d'exécution. Mais avec également la possibilité d'anticiper des données futures.

1.1.1. Données télémétriques en métadonnées parallèles à la vidéo

Comme traitement en amont, on peut d'abord énoncer la possibilité d'un traitement manuel de la vidéo. C'est à dire un traitement image par image, où le déplacement du wagon serait mesuré et approximé à l'œil. Les déplacements seraient ensuite joués en même temps que la vidéo. Cette méthode est très approximative et une solution précaire au problème. Une variante un peu plus précise serait de se procurer les plans des rails de l'attraction et de les traduire directement en mouvement du wagon, cela pose plusieurs problèmes, notamment dû à l'accès à ce type de document pour des attractions réelles, mais aussi à cause de l'efficacité de la méthode laissant beaucoup à désirer d'un point de vue efficacité temporelle puisque chaque attraction doit être traitée individuellement. Ces méthodes sont très chronophages et loin d'être les plus adaptées à notre problème.

Une alternative à cette extraction de données manuelle, serait l'enregistrement avec un outil. En effet, il existe des caméras ou bien des capteurs permettant d'enregistrer à la fois de la vidéo mais aussi en parallèle la position GPS de la caméra, avec d'autres capteurs comme un accéléromètre et un gyroscope, permettant avec un logiciel de reconstituer la trajectoire suivie par la caméra lors de l'attraction ce qui, couplé avec la vidéo, permettra d'avoir le mouvement du wagon relatif au point de vue de la caméra à chaque image de la vidéo.

1.1.2. Traitement du flux vidéo en temps réelle

Il est également possible, de déterminer avec la vidéo seule les mouvements de la caméra. Il existe notamment plusieurs algorithmes. Ces algorithmes peuvent fonctionner sur un flux vidéo en temps réel, mais leur temps de traitement produirait sans doute un délai dû à la faible puissance de calcul du Megabot. En revanche il est possible de traiter les vidéos en amont. L'avantage par rapport aux méthodes énoncées dans la sous-section précédentes, est la flexibilité et l'adaptabilité permise par ce genre d'algorithme, en effet il n'y a aucune restriction sur les vidéos pouvant être traités. Cela laisse donc la possibilité de choisir n'importe quels types de vidéo, quel que soit la caméra où le type d'attraction.

Une méthode fait l'usage de l'algorithme Speeded up Robust Feature Transform (SURF) [21] qui permet l'extraction de points de repère dans l'environnement sur chaque image d'une vidéo et de les mettre en correspondance avec les images qui suivent. Ces points de repères sont définis uniquement par leur voisinage indépendamment de leur position et rotation dans sur l'image. Cette mise en correspondance à travers les différentes images, permet de déterminer pour chaque paire de points caractéristiques, un vecteur différentiel de position. Couplé avec le reste des vecteurs différentiel à un moment donné il est théoriquement possible d'estimer le déplacement de la caméra. Ce qui est notamment utilisé pour stabiliser la caméra mais cela peut aussi être utilisé pour estimer plus globalement la position et rotation de la caméra. Cette estimation peut être fait notamment par un modèle génératif [17].

Une autre méthode étudiée, davantage adapté aux caméras événementielles [13], mais aussi exploitable avec une caméra conventionnelle après un traitement d'image, se base sur l'évolution des pixels enregistrés par la caméra. L'équivalent des point caractéristique de la méthode précédentes sont les cotés et bords des différents objets de l'environnement car ce sont autours de ces bords que les pixels évoluent le plus. Un décalage de la caméra fait que le pixel correspond soudainement à un nouvel objet. Ce qui engendre un changement de couleur important. Pour le tracking la méthode de EL MOUNDI ET AL. [13], utilise un algorithme de Disparity Space Image fusion (DSI) pour recréer l'environnement dans un espace 3D à partir des images perçu, puis avec cet espace la position de la caméra est estimé par rapport à la perspective de son point de vue.

1.2. Récupération des données télémétriques depuis une simulation dynamique

Les simulations dynamiques offrent une meilleure flexibilité et adaptabilité que les vidéos, au prix d'une demande en calcul plus importante en temps réel. Le Megabot possède un ordinateur, mais sachant que cet ordinateur ne possède pas de carte graphique dédiée, les ressources demandées par la simulation doivent soit être surveillées de près ou soit la simulation doit être transférée sur un support externe avec d'avantage de puissance de calcul. Dans ce dernier cas, une interface doit être créée entre l'ordinateur interne du Megabot et le support de simulation.

1.2.1. Interface de simulation externe

Plusieurs simulateurs de montagnes russes externes ont été identifiés. Ce genre de simulateur est relativement populaire dans le milieu du loisir. Il existe déjà des plateformes et d'autres robots permettant de reproduire les mouvements du wagon de la simulation en tant que mouvement de siège immersif. Cela signifie l'existence d'outil ou interface permettant l'extraction de données télémétriques correspondant à notre cas d'usage. Cependant ces outils ou interfaces ont tendance à être fermés et sont conçu pour accommoder seulement certain robots.

Un des premiers simulateurs identifiés est No Limits2 [2], un simulateur de roller coaster pourvu d'une interface de programmation d'application (API en anglais) mise à disposition sur un serveur local, pouvant être exploitée. Les nécessité techniques de ce simulateur sont comprises dans la puissance de calcul disponible avec l'ordinateur interne du Megabot. Le code de commande du Megabot étant en python un client en python de l'API est nécessaire, bien qu'il n'y en ai pas de source officiel, il existe des clients en python créés par la communauté autour simulateur [12]

Un autre simulateur identifié est Epic Roller Coaster [5], contrairement au simulateur précédent, il ne possède pas d'API et nécessite une puissance de calcul dépassant celle de l'ordinateur interne du Megabot. Cependant, il est disponible en interne dans le casque de simulation virtuel Meta quest 2. Cela permet à la simulation de tourner en dehors du Megabot. Le simulateur permet l'envoi des données télémétriques du simulateur sur le réseau local. Ces données peuvent ensuite être récupérées par le Megabot, à condition que le Megabot appartienne au même réseau local que le support du simulateur.

1.2.2. Simulateur intégré au programme de traitement mécanique du Megabot

Il est envisageable de créer un simulateur propre au Megabot permettant un contrôle total sur l'environnement de la simulation. Ainsi, il serait possible d'utiliser au mieux les capacités du Megabot, aussi bien d'un point de vue mécanique que d'un point de vue puissance de calcul. Il serait notamment possible d'envoyer des données télémétriques à l'avance, permettant un meilleur temps de réaction du Megabot avec la possibilité d'anticiper des mouvements. Plusieurs frameworks on été pris en considération.

Notamment PTVR[3] qui est un framework python permettant de développer des expériences VR. Le fait que le framework soit directement en python nous permet un emboîtement direct avec le code de commande mécanique du Megabot. Ce qui enlèverait le besoin d'une API.

Une autre possibilité, est l'utilisation d'un moteur de jeu capable de développer des expérience VR comme Unity, Unreal et Godot. Godot étant open source et moins demandeur en ressource, il serait la piste a privilégier [6]. Un moteur de jeu face à PTVR permet de meilleures performances et de meilleurs graphismes, pour essentiellement une meilleure immersion.

2. Immersion et génération de mouvement

2.1. Étude du champs de perception humain

Afin d'offrir une expérience de simulation convaincante et immersive, les stimuli sensoriaux sont essentiels car ils fabriquent notre perception du monde. Qu'ils soient visuels, auditifs, tactiles ou vestibulaires ils peuvent avoir un effet plus ou moins prononcé sur l'immersion mais aussi sur la cybercinétose[26], l'inconfort lié à l'écart sensoriel provoqué par certains dispositifs de réalité virtuelle.

2.1.1. L'audiovisuel

Le sens le plus important dans notre perception du monde est sans aucun doute la vision. En particulier dans la perception de mouvement, le cerveau suit les indications visuelles avant tout le reste [24]. Ainsi, dans la simulation de sensations fortes une représentation convaincante de l'environnement visuel est cruciale. Les casque de réalité virtuelle sont donc une bonne solution pour ce genre de simulateur car ils offrent entre autres une immersion, perception du mouvement et interactabilité largement supérieure à celle des écrans fixes et même proche de la réalité [18].

L'audio est un sens facile à oublier quand on parle d'expériences multimédia car il est déjà si présent et perfectionné qu'il est pris pour acquis. Cependant d'après les résultats d'une étude de S. Grassini et al. ainsi qu'une étude de M. Teaford et al. la stimulation audio n'aide pas de manière significative dans la sensation de présence et l'atténuation de la cybercinétose [16][24]. Ainsi le matériel audio ne sera pas un élément clé de notre installation mais restera présent car inclus dans les expériences proposées.

2.1.2. Oreille interne

L'accompagnement de simulations visuelles par un mouvement coordonné est une méthode très populaire pour les simulateurs VR et moyennant du matériel coûteux produit aussi une excellente immersion[26]. Il permet aussi de réduire la cybercinétose mieux que d'autres méthodes comme le retour haptique[16] qui a un impact faible sur ces sujets et dépendant du sexe (faible pour les femmes et inexistant pour les hommes). Une plateforme de simulation de mouvement peut avoir des capacités différentes tant en terme de degrés de liberté (degrees of freedom DOF en anglais) que d'amplitude de mouvement. Le Megabot a techniquement 6 DOF mais est sévèrement limité sur certains d'entre eux. Il est donc crucial d'adapter le mouvement aux capacités du Megabot en tenant compte du fonctionnement de l'oreille interne humaine. Ce fonctionnement différencie les mouvements lents imperceptible à l'oreille interne seule et les mouvements rapides facilement descriptible par celle ci. La caractérisation de l'oreille interne dans la littérature

est donc définie par les seuils de perception des mouvements selon les 6 mouvements communément utilisés en aviation : les trois rotations tangage, roulis, lacet et les trois translations cavalement, embardée, pilonnement (en anglais pitch, roll, yaw, surge, sway, heave). Ceux ci montrent un seuil de perception autour de 1 Hz en translation pour un mouvement sinusoïdal mais des résultats très variés entre les études allant de 0.1 Hz à plus de 3 Hz pour les rotations [14][22]. Ces seuils non nuls, bien que difficiles à déterminer, permettent aux simulateurs d'en user pour jouer des tours à l'utilisateur et simuler des mouvements largement au-delà des capacités réelles du simulateur. L'utilisation principale de ces limitations est le remplacement d'accélération longitudinale basse fréquence par une inclinaison correspondante qui est indistinguable par l'oreille interne tant que la transition est assez lente [19]. Ainsi il est possible de prendre en compte les capacités et surtout incapacités de l'oreille interne pour simuler des mouvements réalistes avec des plateformes de mouvement limitées, en degrés de liberté ou en amplitude.

2.2. Adaptation du mouvement à un simulateur

Le but d'un simulateur est d'effectuer des actions complexes à plus petite échelle. C'est pourquoi un simulateur est fondamentalement limité dans ces capacités et il est nécessaire d'adapter le mouvement demandé par l'expérience pour la reproduire fidèlement. On élabore et perfectionne donc un algorithme de transformation du mouvement afin de maximiser les sensations fortes de l'attraction simulée.

2.2.1. Algorithmes de transformation

La littérature sur les algorithmes de transformation de mouvement (motion cueing algorithms MCA) est vaste et détaillée. Les algorithmes tentent de résoudre un problème difficile à définir et encore plus difficile à mesurer - le réalisme du mouvement - et de nombreuses versions ont été développées et publiées. En pratique ils prennent en entrée l'accélération translationnelle et la vitesse angulaire pour sortir les positions en translation et en rotation. Voici un inventaire des différentes idées [22][11] :

Classical washout filters – (CL)

Le Classical Washout Filter est le premier algorithme de transformation de mouvement proposé. Publié en 1970 par Conrad et Schmidt, il reste aujourd'hui le plus utilisé pour sa simplicité d'implémentation et ses résultats difficiles à égaler. Il utilise trois briques essentielles : des transformations linéaires, des intégrateurs et des filtres (passe-haut et passe-bas). Le principe repose sur la séparation des accélérations en composantes haute et basse fréquence : les hautes fréquences sont reproduites par le mouvement translationnel de la plateforme, tandis que les basses fréquences (accélérations soutenues) sont simulées par inclinaison (tilt coordination). Des filtres passe-haut assurent l'effet "washout" qui ramène progressivement le simulateur vers sa position neutre.

Adaptive washout filters

Les filtres adaptatifs, développés initialement par Parrish et al. (1975) puis Reid et Nahon (1985), représentent une évolution des filtres classiques. Leur principe consiste

à adapter en temps réel les paramètres des filtres pour minimiser une fonction de coût basée sur les erreurs de perception. L'algorithme utilise des méthodes de descente de gradient pour ajuster dynamiquement les gains et fréquences de coupure. Cette flexibilité permet une meilleure exploitation de l'espace de travail du simulateur et une réduction des faux indices de mouvement. Cependant, leur stabilité dépend fortement des paramètres d'adaptation et leur réglage nécessite une expertise significative.

Linear optimal washout filters

Introduits par Sivan et al. (1982), ces filtres utilisent la théorie du contrôle optimal pour déterminer des filtres d'ordre supérieur. Ils intègrent un modèle du système vestibulaire humain pour minimiser l'erreur entre les sensations perçues dans le véhicule réel et dans le simulateur. La fonction de coût combine les erreurs sensorielles et les contraintes physiques du simulateur. Reid et Nahon (1985) puis Telban et al. (2005) ont progressivement affiné les modèles vestibulaires utilisés. Bien que mathématiquement élégants, ces algorithmes présentent des paramètres moins intuitifs à régler et leur efficacité dépend de la validité des modèles de perception employés.

Nonlinear optimal washout filters – OpTNon

Développés par Telban et al. (2005), ces filtres combinent les concepts des algorithmes adaptatifs et optimaux. Ils résolvent en temps réel l'équation de Riccati pour trouver les filtres optimaux à chaque pas de temps, en tenant compte de l'état actuel du simulateur. L'algorithme intègre également un modèle visuel-vestibulaire pour une meilleure cohérence multimodale. Cette approche permet d'adapter le comportement aux manœuvres en cours plutôt qu'au pire scénario, réduisant ainsi les faux indices. Le principal défi réside dans le coût calculatoire élevé nécessaire pour résoudre l'équation de Riccati en temps réel.

MCA using optimal tracking – ZyRo

L'algorithme ZyRo, proposé par Zywiol et Romano (2003), adopte une approche différente en résolvant un problème de suivi optimal de trajectoire. Plutôt que d'utiliser des modèles de perception, il cherche directement la meilleure combinaison d'accélération linéaire et d'angle d'inclinaison pour reproduire la force spécifique cible. La fonction de coût pénalise les écarts avec la force spécifique désirée ainsi que la vitesse et position du simulateur. L'algorithme utilise des filtres passe-bas simples et résout les équations de Riccati et co-état en temps différé. Son principal avantage est la transparence de ses paramètres qui correspondent directement aux quantités physiques, facilitant ainsi le réglage.

MCA using model predictive control theory

Les approches par commande prédictive (MPC), introduites par Dagdelen et al. (2004), représentent l'état de l'art actuel. Elles optimisent la trajectoire future du simulateur sur un horizon de prédiction en tenant compte explicitement des contraintes physiques (limites d'espace, seuils de perception) et d'un modèle de la dynamique. Augusto et Loureiro

(2009) ont développé une version implicite utilisant la programmation quadratique, tandis que Fang et Kemeny (2012) ont proposé une version explicite calculée hors ligne pour réduire le temps de calcul. Les MPC permettent une excellente gestion de l'espace de travail et des faux indices, mais nécessitent une puissance de calcul importante pour les versions implicites.

Comparaison des algorithmes

L'étude comparative menée par Pham révèle plusieurs conclusions importantes. Pour des facteurs d'échelle faibles ($k=0.4$), tous les algorithmes présentent des performances similaires avec un indice de qualité élevé. Cependant, lorsque l'amplitude du signal augmente, les différences deviennent significatives : les algorithmes ZyRo et MPC* maintiennent un indice de qualité élevé jusqu'à $k=0.7-0.8$, tandis que les algorithmes classiques et optimaux linéaires échouent dès $k=0.5$. Les algorithmes ZyRo et MPC* exploitent mieux l'espace de travail grâce à leur capacité d'anticipation et génèrent des mouvements préparatoires avant les changements brusques. Le temps de calcul varie considérablement : les algorithmes classiques sont quasi-instantanés (4s), les adaptatifs plus lents (600-1300s), et les MPC implicites très gourmands (1900s+), tandis que les filtres optimaux et MPC explicites nécessitent un pré-calcul long mais s'exécutent rapidement ensuite. Enfin, la facilité de réglage favorise nettement ZyRo et MPC* dont les paramètres ont une signification physique claire, contrairement aux filtres optimaux dont l'effet des pondérations est opaque.

2.2.2. Algorithme classic washout

Après cette comparaison et au regard de la difficulté d'implémentation de ces algorithmes nous avons choisi d'intégrer le Classical Washout Algorithm. La figure 2.1 décrit le fonctionnement en détail de cet algorithme. Du côté de l'accélération en translation, elle est d'abord transformée linéairement (TA) puis transformée dans le repère monde (TX) avant d'être filtrée par un passe haut (THPF) afin de ne garder que les variations et ne pas atteindre les limites du système. Enfin un double intégrateur permet de transformer la commande en accélération en commande en position. Pour la transformation des vitesses de rotation en angles le processus est le même mais avec des filtres et transformations aux coefficients différents. L'ingéniosité de l'algorithme est contenu dans le canal de coordination reliant les mouvements basse fréquence en accélération linéaire aux angles d'inclinaison en tirant parti de l'incapacité de l'oreille interne à distinguer une accélération constante additionnée à la gravité d'une inclinaison et de la gravité [10]. Ainsi ce canal est composé d'un filtre passe bas (LPF), d'un Tilt Coordinator (TC) qui fait le lien entre accélération et inclinaison et enfin un limiteur (RL) présent pour empêcher la commande d'excéder les capacités du simulateur.

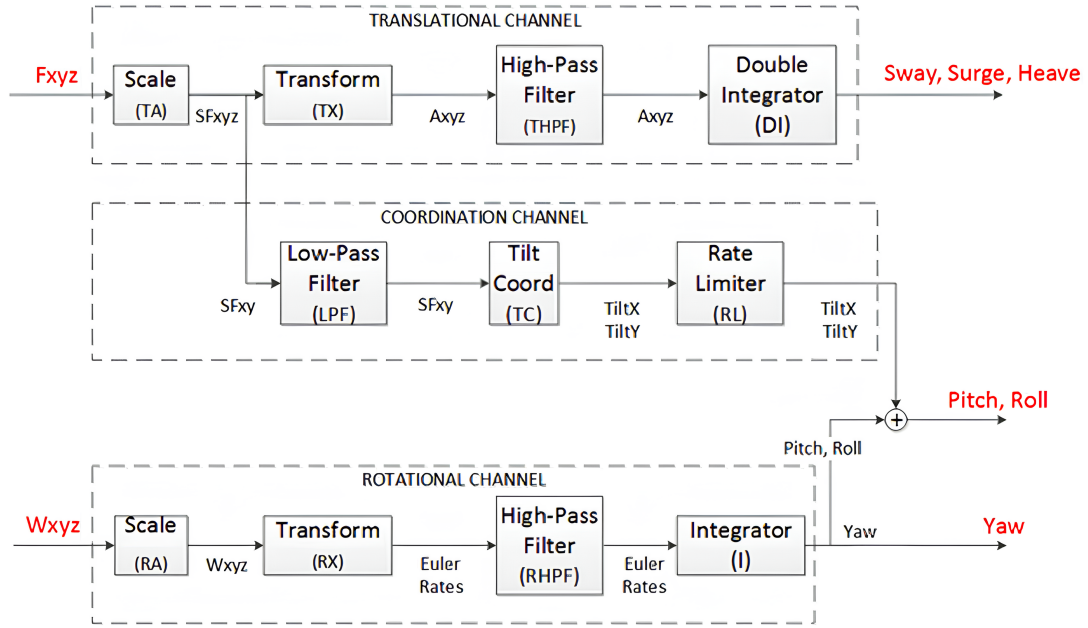


FIGURE 2.1. – Classical washout scheme (Reid-Nahon UTIAS implementation) [11]

2.2.3. Maximisation des sensation fortes

Après avoir choisi l'algorithme utilisé le plus important est d'être capable de trouver des paramètres satisfaisants pour les différents blocs. En effet les performances d'un MCA sont entièrement dépendantes de la configuration de ses coefficients lors de la phase de paramétrage. Pour cette phase il y a différentes méthodes mais la définition de métriques est un prérequis important.

En effet pour résoudre un problème il est nécessaire d'avoir des métriques sur lesquelles s'appuyer en cherchant la solution. La définition de notre métrique principale serait "l'immersion utilisateur" mais c'est extrêmement difficile à quantifier et presque entièrement subjectif. Il est tout de même possible de trouver des critères objectifs mais il n'en existe aucun de standardisé dans les MCA et les performances de tels critères seront toujours limitées [11].

La méthode la plus commune de paramétrage est donc la recherche manuelle, demandant d'avoir un pilote et un ingénieur expérimenté. Le pilote donne des retours à l'ingénieur sur les sensation de pilotage et l'ingénieur avec une connaissance profonde des paramètres de l'algorithme répercute ces retours. Le va-et-viens se fait jusqu'à ce que les paramètres soient satisfaisants. C'est une méthode qui peut être longue et qui demande une expertise au niveau des paramètres de l'algorithme, ou du moins une compréhension de leurs effets.

Une méthode qui tente de résoudre ce problème est le paramétrage automatique basé sur la simulation [10]. Celle ci propose des métriques objectives permettant l'apprentissage automatique grâce à un jumeau numérique du simulateur permettant de tester des configurations largement plus rapidement que sur le simulateur physique. L'apprentissage repose sur un algorithme génétique où les individus sont des sets de paramètres pouvant

être interprétés comme des gènes. Ainsi, leurs performances sont évaluées par la simulation et les meilleurs se reproduisent tandis que les pires disparaissent et qu'une partie des gènes mutent aléatoirement. Additionnellement les quelques meilleurs individus sont clonés dans la génération suivant afin de ne pas perdre les meilleurs éléments. Ainsi l'apprentissage n'est plus paramétré que par 4 paramètres : la taille de la population, la mortalité, l'élitisme (taux de meilleurs éléments gardés) et la probabilité de mutation. Cette méthode a prouvé être très efficace pour des temps de paramétrages courts mais ne rivalise pas avec la présence d'un retour humain si suffisamment de temps est donné pour paramétrer l'algorithme.

3. Commande et contrôle du Megabot

La commande et le contrôle du Megabot constituent l'interface essentielle entre les trajectoires générées par l'algorithme d'immersion et les actions physiques exécutées par les vérins. Cette couche assure la traduction des positions cibles en consignes moteur tout en garantissant le respect des limites mécaniques, de sécurité et de performance. La plateforme utilise aujourd'hui la bibliothèque PlaCo, un outil dédié au calcul de cinématique et d'optimisation, déjà déployé sur plusieurs projets de robotique mobile et humanoïde de l'équipe Rhoban. Cette infrastructure permet d'exploiter efficacement les six degrés de liberté du Megabot tout en maintenant un contrôle précis et temporellement cohérent avec l'application immersive envisagée [7].

3.1. Commande

La commande vise à générer des consignes moteur réalisables en fonction des contraintes mécaniques, des limites d'amplitude et des exigences de sécurité. Elle constitue l'étape qui permet de convertir les objectifs de mouvement souhaités en actions pilotables par les vérins du Megabot. Dans l'architecture actuelle, la commande est implémentée à l'aide de la bibliothèque PlaCo, qui fournit les outils nécessaires à la cinématique inverse, à la gestion des contraintes et au calcul optimisé des trajectoires en temps réel [7].

3.1.1. Asservissement

Pour atteindre la position cible, il est nécessaire de déterminer la trajectoire pour l'atteindre. Pour notre cas, il est important d'avoir une interpolation la plus rapide possible, une interpolation linéaire dans l'espace cartésien est donc loin d'être la solution, puisque elle se traduit par une trajectoire complexe dans l'espace des degrés de liberté. Il est donc généralement nécessaire d'établir une fonction de distance pondérée dans l'espace des degrés de liberté qui se rapproche de la distance temporelle.

De plus, dans le cas où l'on souhaite suivre une trajectoire bien spécifique, c'est à dire pouvant être définie par plusieurs points de passages, il existe plusieurs méthodes permettant une interpolation via des trajectoire en courbe dans l'espace des degrés de liberté. De manière similaire au problème de la cinématique inverse, cela se rapproche d'un problème d'optimisation, où l'on cherche soit à minimiser le temps mis, soit l'énergie d'où en tenant compte de l'erreur moyenne par rapport à la trajectoire [9]. La planification permet une trajectoire très adoucie. Cela peut se révéler intéressant dans le cas où l'on pré-calcule les commandes du Megabot.

En revanche si il est convenu de suivre en temps réel une trajectoire il n'est pas question de planifier la trajectoire. Il est donc impossible de générer de courbe de Bézier ou autre interpolation sans prendre de retard avec la trajectoire cible. Or le retard est quelque chose à éviter dans notre cadre de simulation immersive. Donc dans notre cas,

l'établissement de la trajectoire doit se faire localement. Dans le cas du Megabot les degrés de liberté sont tous de même nature, chaque degré concerne l'extension individuelle de chaque vérin qui sont du même modèle. De ce fait l'implémentation actuelle avec le Megabot est relativement simple. La distance dans l'espace des degrés de liberté sans pondération est suffisante. La trajectoire minimisant cette distance à la position cible est celle adopté par l'implémentation avec PlaCo.

3.1.2. Comportement aux extrêmes

Bien que la commande du Megabot est prévue pour ne jamais donner une position cible inatteignable, par mesure de sécurité le contrôle du Megabot doit être assuré pour toutes les positions y compris celles qui sont impossibles par rapport aux contraintes mécaniques. Dans l'implémentation actuelle du Megabot, lors de l'approche d'une limite d'un des vérins en question la vitesse est diminuée de moitié, et est réduite à zéro à partir d'un seuil critique. Or c'est quelque chose qu'il serait mieux d'éviter. En effet cela peut complètement déformer la trajectoire. Il est préférable de déformer la trajectoire le moins possible. Une des solutions serait de modéliser l'espace des possibles pour éviter à tout prix ces zones de "danger". Un des problèmes est comment modéliser cet espace des possibles.

Une des approches est celle de représenter l'espace des possibles sous forme de volume [23]. Ce volume nous permettrait alors de délimiter une zone de sécurité claire, permettant ainsi de contraindre les trajectoires à rester dans cette zone. De ce fait le robot ne serait jamais arbitrairement ralenti. Cependant étant donné que le Megabot dispose de douze degrés de liberté, modéliser un tel volume serait très demandeur en calculs, car en effet ce serait un volume en 12 dimension. Une des solutions serait de simplifier cet espace, notamment par effet de symétrie. Puisque les quatre pattes sont similaires, il est prévisible que le volume comporterait des axes de symétrie.

En plus des limites dans l'espace des positions, le Megabot est aussi contraint par des limites de forces et de vitesses. Assurer ces contraintes n'est pas un problème et il est même impossible de les ignorer, à l'exception du jumeau numérique du Megabot qui n'est pas restreint mécaniquement. En revanche les contraintes de vitesse et de force ont des conséquences sur le suivi de trajectoires.

3.2. Cinématique

Afin de passer d'une position à une autre, il est nécessaire de connaître la cinématique du robot dont il est question. En effet, les instructions moteurs doivent être calculées avant d'être envoyées aux vérins du Megabot.

3.2.1. Cinématique Directe

La question de la cinématique directe du Megabot comme n'importe quel autre robot, est déjà résolue mathématiquement de manière analytique et automatiquement de manière numérique. Le Megabot est constitué de quatre pattes et d'une plateforme centrale. Chaque patte est articulée par trois joints actionnés par trois vérins. La cinématique directe consiste à déterminer la position et rotation de la plateforme centrale mais aussi la

positions des pieds du Megabots selon les variables, qui sont dans le cas des Megabots, le degrés d'extension des vérins.

Chacune des pattes sont similaires dans leur conception. Par symétrie la cinématique d'une des pattes calculés par méthode analytique peut être appliqué sur les trois autres [20]. La plateforme centrale peut ensuite être calculé avec le positionnement des quatre pattes. Cette approche analytique nécessite de faire l'étude géométrique et mécanique des pattes du Megabot, mais garantis une précision et une grande vitesse de calcul.

L'implémentation actuel du contrôle du Megabot utilise le simple calcul numérique utiliser par la librairie PlaCo [7]. Le robot possède un modèle numérique URDF. Plusieurs librairies existent permettant de faire des calculs avec une modélisation numérique au format URDF. En plus de simulateur comme PlaCo. Il y a également des moteurs physiques comme PyBullet, où le moteur physique de Unity (Pouvant servir également pour la simulation VR) [8]. Ou bien directement des librairies de résolution et d'optimisation comme CasADI [15].

Dans les deux cas, ces deux méthodes donne seulement une estimation du résultat car les approches sont basées sur des modèles qui n'incluent pas les déformations des matériaux, toutes les pièces du robot sont considérées comme des solides indéformables, or avec le poids du Megabot les déformations des pièces ne sont pas totalement négligeables. Un simulateur comme Isaac Sim, avec l'usage de PhysX 5 peut traiter la déformation des matériaux [1]. Cependant cela nécessite une carte graphique et le Megabot n'en est actuellement pas équipé.

3.2.2. Cinématique Inverse

La cinématique inverse est nettement plus complexe que la cinématique directe. En effet, calculer analytiquement la cinématique inverse d'un robot peut nécessiter la résolution d'un système non linéaire complexe. Les outils mathématiques pour résoudre ces système ne sont pas disponibles. Dans le cas du Megabot, qui possède des chaînes cinématique à trois joints actionnés par des vérins, les outils mathématiques n'ont pas été trouvés.

C'est pourquoi l'implémentation actuel du Megabot, utilise une approximation numérique avec le simulateur PlaCo. L'approximation consiste à effectuer une opération d'optimisation ou plutôt de minimisation sur la distance entre la position cible est la position calculé par cinématique directe en fonction des différents degrés de liberté. Cette solution peut être implémenté de manière itérative avec un algorithme de Newton à l'aide de la Jacobienne des position des effecteurs par rapport aux degrés de liberté. Étant donné que le Megabot est soumis à des contraintes, une seule solution ne suffit pas. Il est nécessaire de trouver une solution qui satisfait aussi les contraintes mécaniques. C'est pourquoi des approches par algorithme génétique ou bien filtre de particule peuvent se révéler utile [27]. PlaCo en particulier utilise une optimisation utilisant un résolveur de programmation quadratique [4]. Qui permet une résolution de système d'ordre quadratique.

Une autre solution serait l'entraînement d'un réseau de neurones permettant une estimation des degrés pour une position données. Ce réseau de neurones peut être entraîné de manière superviser avec des données collecté à l'aide de la cinématique directe [25].

Une longue étape d'entraînement serait nécessaire pour arriver à la précision des résolveurs numériques. Cependant les contraintes mécaniques pourraient facilement être intégrées au modèle. Cette méthode pourrait éventuellement permettre une cinématique inverse plus rapide que les méthodes conventionnelles décrites précédemment.

Conclusion & perspectives

En conclusion, tous les outils nécessaires aux projets sont disponibles. Les outils déjà implémentés pour le contrôle du Megabot peuvent être ré-utilisés sans problèmes ou difficulté d'envergure avec le reste de l'implémentation. Cet état de l'art porte avant tout sur la génération de mouvement sous contraintes ainsi que sur la simulation de montagne russe d'un point de vue sensation et proprioception.

Cet état de l'art permet de surligner plusieurs chemins pour procéder à une solution du projet. Un de ces chemins est le plus prometteur étant donné les ressources à disposition. Celui-ci va être privilégié lors de l'implémentation.

Il concerne l'utilisation du simulateur dynamique Epic Roller coaster, avec une connexion au réseau local pour communiquer les informations télémétriques. Les données télémétriques seront ensuite traitées par un filtre de classical wash-out. En sortie de cet algorithme, une position cible sera obtenue. La partie contrôle commande déjà existante implémentée avec PlaCo suivra cette position cible.

Cette implémentation est un point de départ. Beaucoup d'éléments peuvent être améliorés, notamment sur la partie commande avec le suivi de trajectoire qui pourra potentiellement être entravé par les contraintes mécaniques. De plus l'efficacité du filtre classical washout dépendra de sa calibration, qui nécessitera beaucoup d'essais. En plus de cela, il serait possible d'ajouter des stimulateurs supplémentaires à l'expérience pour renforcer l'immersion.

Cet état de l'art seul ne peut pas donner la meilleure solution au problème, le travail d'essai et d'implémentation reste une tâche conséquente et nécessaire à cela, cependant l'état de l'art donne les ressources nécessaires pour y parvenir.

A. Annexe

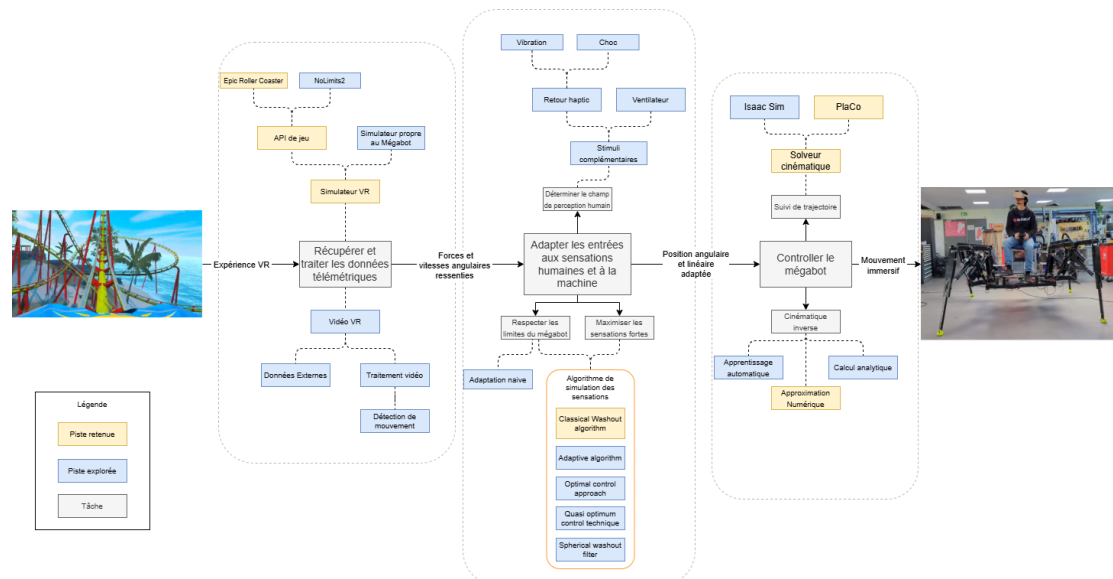


FIGURE A.1. – Schéma des recherches effectués pour le problème donné

Bibliographie

- [1] Interacting with a deformable object — Isaac Lab Documentation.
- [2] NoLimits 2 - Roller Coaster Simulation - Home.
- [3] Overview – Perception Toolbox for Virtual Reality (PTVR).
- [4] Welcome to PlaCo’s documentation — PlaCo documentation.
- [5] B4T - Telemetry for motion simulators, July 2022.
- [6] godotengine/godot, December 2025. original-date : 2014-01-04T16 :05 :36Z.
- [7] Rhoban/placo, December 2025. original-date : 2022-10-19T08 :02 :34Z.
- [8] Unity-Technologies/Unity-Robotics-Hub, December 2025. original-date : 2020-09-24T17 :23 :13Z.
- [9] Atef A Ata. Optimal Trajectory Planning of Manipulators : A Review. 2, 2007.
- [10] Sergio Casas, Inmaculada Coma, Cristina Portalés, and Marcos Fernández. Towards a simulation-based tuning of motion cueing algorithms. *Simulation Modelling Practice and Theory*, 67 :137–154, September 2016.
- [11] Sergio Casas, Ricardo Olanda, and Nilanjan Dey. Motion Cueing Algorithms : A Review : Algorithms, Evaluation and Tuning. *Int. J. Virtual Augment. Real.*, 1(1) :90–106, January 2017.
- [12] Daniel. bestdani/py_nl2telemetry, August 2025. original-date : 2018-10-05T16 :02 :06Z.
- [13] Anass El Moudni, Fabio Morbidi, Sebastien Kramm, and Rémi Boutteau. An event-based stereo 3D mapping and tracking pipeline for autonomous vehicles. In *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*, pages 5962–5968. IEEE, 2023.
- [14] Daniel C. Fitze, Fred W. Mast, and Matthias Ertl. Human vestibular perceptual thresholds — A systematic review of passive motion perception. *Gait & Posture*, 107 :83–95, 2024.
- [15] Lill Maria Gjerde Johannessen, Mathias Hauan Arbo, and Jan Tommy Gravdahl. Robot Dynamics with URDF & CasADi. In *2019 7th International Conference on Control, Mechatronics and Automation (ICCMA)*, pages 1–6, 2019.
- [16] Simone Grassini, Karin Laumann, Virginia de Martin Topranin, and Sebastian Thorp. Evaluating the effect of multi-sensory stimulations on simulator sickness and sense of presence during HMD-mediated VR experience. *Ergonomics*, 64(12) :1532–1542, December 2021. Publisher : Taylor & Francis _eprint : <https://doi.org/10.1080/00140139.2021.1941279>.
- [17] Wei He, Takayoshi Yamashita, Hongtao Lu, and Shihong Lao. SURF Tracking. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1586–1592, 2009.

- [18] Daniel Hepperle and Matthias Wölfel. Similarities and Differences between Immersive Virtual Reality, Real World, and Computer Screens : A Systematic Scoping Review in Human Behavior Studies. *Multimodal Technologies and Interaction*, 7(6) :56, June 2023. Publisher : Multidisciplinary Digital Publishing Institute.
- [19] SJ Hodge, P Perfect, GD Padfield, and MD White. Optimising the roll-sway motion cues available from a short stroke hexapod motion platform. *THE AERONAUTICAL JOURNAL*, 119(1211) :23, 2015.
- [20] Serdar Kucuk and Zafer Bingul. *Robot kinematics : Forward and inverse kinematics*, volume 1. INTECH Open Access Publisher London, UK, 2006.
- [21] Edouard Oyallon and Julien Rabin. An Analysis of the SURF Method. *Image Processing On Line*, 5 :176–218, 2015.
- [22] Duc An Pham. A Study on State-of-the-Art Motion Cueing Algorithms applied to Planar Motion with Pure Lateral Acceleration — Comparison, Auto-Tuning and Subjective Evaluation on a KUKA Robocoaster Serial Ride Simulator. November 2017.
- [23] Jianbo Su. Evaluation and Transformation of Unrealizable Tasks for Robot Systems in Representation Space. *IEEE Access*, 7 :81532–81541, 2019.
- [24] Max Teaforde, Zachary J. Mularczyk, Alannah Gernon, Shauntelle Cannon, Megan Kobel, and Daniel M. Merfeld. Joint Contributions of Auditory, Proprioceptive and Visual Cues on Human Balance. *Multisensory Research*, 36(8) :865–890, October 2023. Publisher : Brill.
- [25] Juan S. Toquica, Patrícia S. Oliveira, Witenberg S. R. Souza, José Maurício S. T. Motta, and Díbio L. Borges. An analytical and a Deep Learning model for solving the inverse kinematic problem of an industrial parallel robot. *Computers & Industrial Engineering*, 151 :106682, 2021.
- [26] Guillaume Vailland, Yoren Gaffary, Louise Devigne, Valérie Gouranton, Bruno Arnaldi, and Marie Babel. Vestibular Feedback on a Virtual Reality Wheelchair Driving Simulator : A Pilot Study. In *Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '20, pages 171–179, New York, NY, USA, March 2020. Association for Computing Machinery.
- [27] Linlin Zhang, Huibin Du, Zhiying Qin, Yuejing Zhao, and Guang Yang. Real-time optimized inverse kinematics of redundant robots under inequality constraints. *Scientific Reports*, 14(1) :29754, November 2024. Publisher : Nature Publishing Group.