Titre: Title:	Simulation et implémentation d'accostage autonome de multicoptère sur paroi verticale
Auteur: Author:	Mathis Celce
Date:	2022
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Celce, M. (2022). Simulation et implémentation d'accostage autonome de multicoptère sur paroi verticale [Master's thesis, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/10504/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/10504/
Directeurs de recherche: Advisors:	Lionel Birglen
Programme: Program:	Génie aérospatial

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Simulation et implémentation d'accostage autonome de multicoptère sur paroi verticale

MATHIS CELCE
Département de génie mécanique

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie aérospatial

Août 2022

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Simulation et implémentation d'accostage autonome de multicoptère sur paroi verticale

présenté par **Mathis CELCE**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Giovanni BELTRAME, président
Lionel BIRGLEN, membre et directeur de recherche
David ST-ONGE, membre

REMERCIEMENTS

Merci à Bruno Chapdelaine, ingénieur de recherche au Conseil National des Recherches Canada (CNRC), pour ses conseils, sa disponibilité, et sa confiance accordée.

Merci à Lionel Birglen, mon directeur de recherche à Polytechnique Montréal, pour son écoute et son aide à la rédaction de l'article.

Merci à mes proches, à ma famille et ma compagne pour leur précieux soutien.

RÉSUMÉ

De nombreuses structures telles que les éoliennes, les lignes électriques, les barrages ou les ponts sont particulièrement difficiles d'accès pour l'homme et doivent pourtant régulièrement être inspectées et entretenues. Pour économiser le temps nécessaire aux techniciens pour accéder aux zones d'intérêt, des solutions faisant appel à des véhicules aériens sans pilotes (UAV) sont utilisées. Cependant, ces véhicules sont d'avantage dédiés à l'inspection et la surveillance, et leur emploi pour des tâches d'entretien n'en est qu'à son balbutiement. Afin de contribuer à la démocratisation des manipulateurs aériens dédiés à l'entretien d'infrastructures, ce mémoire propose un environnement virtuel pour simuler l'accostage de drones à des parois verticales. Une stratégie d'accostage autonome y est développée et validée expérimentalement.

L'accostage est une solution proposée depuis peu pour remédier aux problèmes d'autonomie des drones, et aux limitations des performances du manipulateur embarqué. En effet, une fois le véhicule solidement accroché à la structure, il peut éteindre ses moteurs et son manipulateur devient capable d'effectuer des tâches d'intensité plus importante et de manière plus précise.

Pour mener à bien la réalisation et la validation de la simulation d'une telle manoeuvre, un véhicule de démonstration ainsi que son système d'attache, pour se fixer à une surface verticale, sont conçus. La méthode pour modéliser l'accostage consiste à modifier un environnement de simulation existant et à y implémenter un code qui décrit le comportement du dispositif d'accrochage. Pour démontrer l'efficacité de cette simulation une stratégie d'accostage autonome est élaborée dans l'environnement virtuel puis exécutée expérimentalement.

Cette stratégie doit permettre d'accoster automatiquement un drone mutlirotors à une surface verticale. On suppose qu'un pilote place approximativement le drone face à une zone d'accostage possédant un faible rayon de courbure. La surface doit aussi être suffisamment large pour permettre au véhicule d'accoster facilement malgré les imprécisions de son système de positionnement GPS. Le drone effectue ensuite seul l'approche finale, l'accroche de son système d'accostage à la paroi, l'atterrissage et l'arrêt des moteurs. Une fois les éventuelles missions remplies par le véhicule, celui-ci re-décolle, se décroche de la paroi et s'immobilise à une certaine distance de celle-ci.

Les résultats de l'accostage expérimental, comparés à ceux obtenus dans la modélisation, démontrent la capacité de l'environnement de simulation ainsi mis en place à permettre la mise au point complète et fiable d'une méthode d'accostage autonome. Ainsi, le développement de manoeuvres d'accostages pour des manipulateurs aériens, pouvant être très lourds, est rendu

plus sécuritaire et moins chronophage que leur conception directement dans le monde réel.

ABSTRACT

Many structures such as wind turbines, power lines, dams or bridges are particularly difficult for humans to access and yet need to be regularly inspected and maintained. To save the time needed for rope access technicians to reach the areas of interest, solutions using Unmanned Aerial Vehicles (UAV) are employed. However, these vehicles are more dedicated to inspection and surveillance, and their deployment for maintenance tasks is still in its infancy. In order to contribute to the democratization of aerial manipulators dedicated to infrastructure maintenance, this thesis proposes a virtual environment to simulate the perching of UAVs on vertical walls. An autonomous perching strategy is developed within the simulation and experimentally validated.

Perching is a newly proposed solution to increase the operating time of UAVs, and to improve the performance of the onboard manipulator. Once the vehicle is firmly attached to the structure, it can indeed turn off its motors, and its manipulator becomes able to perform tasks of greater intensity and more accurately.

To carry out the realization and the validation of the simulation of such a manoeuvre, a demonstration vehicle as well as its attachment system, to perch itself on a vertical surface, are designed. The approach to model the perching is to modify an existing simulation environment and to implement a code that describes the behavior of the perching device. To demonstrate the effectiveness of this simulation, an autonomous perching strategy is developed in the virtual environment and then experimentally conducted.

This strategy aims at automatically perching a multirotor UAV on a vertical surface. It is assumed that a pilot places the UAV approximately in front of a perching area with a small radius of curvature. The surface must also be large enough to allow the vehicle to successfully perch despite the inaccuracies of its GPS positioning system. The drone then performs on its own the final approach, the attachment of its perching system to the wall, the landing and the powering off of the motors. Once the vehicle has completed its tasks, it takes off again, detaches from the wall and hoover at a certain distance from it.

The results of the experimental perching, compared to those obtained in the modeling, demonstrate the capacity of the simulation environment thus set up to allow the complete and reliable development of an autonomous perching method. In this way, the development of perching maneuvers for aerial manipulators, potentially very heavy, is made safer and less time consuming than designing them directly in the real world.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

BAM	Base Attachement Module
CNRC	Conseil National des Recherches Canada
CRA	Centre de Recherche en Aérospatiale
CTFA	Centre des Technologies de Fabrication en Aérospatiale
DA	Dispositif d'Amarrage
HITL	Hardware In The Loop
I²2	Inter-Integrated Circuit
IMU	Inertial Measurement Unit
ROS	Robot Operating System
RPi	Raspberry Pi
SITL	Software In The Loop
TCP	Transmission Control Protocol
UART	Universal Asynchronous Receiver Transmitter
UAS	unmanned aircraft systems
UAV	Unmanned Aerial Vehicle
UDP	User Defined Protocol

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION ET REVUE DE LITTÉRATURE

Les travaux présentés dans ce mémoire ont été conduits dans le but de mettre en place un environnement virtuel permettant de simuler l'accostage autonome d'un multicoptère sur une paroi verticale. L'amarrage à la paroi s'opère par le biais d'un dispositif d'amarrage (DA) placé en avant du drone. L'environnement de simulation doit permettre l'exploration des différents capteurs nécessaires à cette opération d'accostage, le changement de configuration de l'environnement et du drone, et bien sûr, l'expérimentation de la stratégie d'accostage autonome, du point de vue algorithmique. L'utilisation de cet environnement permet de développer une solution d'accostage autonome sans perdre de temps avec les précautions de sécurité, la gestion de l'autonomie, les vérifications pré-vol imposées par les changements de configurations et les réparations d'éventuels dégâts.

1.1 Cadre des travaux de recherche

Ce projet s'inscrit dans un programme du Centre de Recherche en Aérospatial (CRA) du Conseil National des Recherches Canada (CNRC) dénommé Mobilité Aérienne Intégrée (IAM [1]). Les sujets de recherche du programme IAM sont entre autres axés sur la planification de trajectoire de véhicules aériens, la conception de batteries au lithium performantes et sûres ou encore, comme il en est question ici, l'accostage de drones pour des opérations de maintenance. Dans ce contexte, le drone doit embarquer un bras robotique pour assurer des tâches de contact, tel qu'un enlèvement de matière, du ponçage, de la peinture, du nettoyage, du vissage, du cloutage, etc. sur diverses structures (barrages, ponts, éoliennes, immeubles). Pour effectuer ces tâches avec précision et sans avoir à s'inquiéter de l'autonomie limitée du drone, il est proposé d'accoster le véhicule sur la structure à traiter. Un brevet [1] a été déposé par le laboratoire de robotique aérienne du Centre des Technologies de Fabrication en Aérospatial (CTFA) appartenant au CNRC, pour la conception d'un dispositif d'amarrage (DA) multi-ventouses pour l'accostage de drones sur parois verticales. Nommé le Base Attachement Module (BAM), ce système de ventouses est capable de supporter des charges de plus de 40 kg sur diverses surfaces verticales telles que du béton poreux ou un mur de parpaings peint. Ses caractéristiques permettent l'arrimage d'un drone tel que le Hercules 20 [2], d'une masse à vide de 11kg emportant un bras robotique de 9kg (masse réservée au bras sur les 15kg de charge utile). Les éléments ayant amené au choix de la solution d'accostage de

1. Integrated Aerial Mobility program
2. https ://www.dronevolt.com/fr/solutions/hercules-20/

drone équipé de bras robotique pour l'entretien de structures seront détaillés dans la section suivante, consacrée à la revue de littérature.

1.2 Plan du mémoire

La section 1.3 du chapitre 1 donne un aperçu de l'état de l'art dans les domaines connexes aux travaux présentés : l'inspection par drone, la robotique mobile au service de la maintenance, l'accostage de drones, les simulations de drones et la modélisation de ventouses. Le chapitre 2 présente le positionnement de l'article dans l'ensemble des travaux de recherche et les éléments de la problématiques auquel il répond. L'article en question, soumis au Journal of Intelligent and Robotic Systems publié par Springer, constitue le chapitre 3. Le chapitre 4 apporte des résultats complémentaires sur des éléments nécessaires à la réussite du projet, tel que le DA ou les systèmes de positionnement. Une discussion générale sur l'ensemble des travaux consignés dans le mémoire est présentée au chapitre 5 suivi de la conclusion au chapitre 6.

Les tableaux 1.1, 1.2 et 1.3 présentent la contribution des deux principales personnes ayant travaillé sur le projet d'accostage autonome. Les contributions de l'auteur du mémoire sont représentées en vert et celles de Bruno Chapdelaine en orange, et les contributions partagées sont en vert clair.

Tableau 1.1 Contributions aux différents modules du monde réel

Module du monde réel	**Sous module**	**Section du mémoire**
Dispositif d'Accostage (DA)	Conception	3.3.3, 4.1.2, 4.1.4
	Fabrication	3.3.3, 4.1.2, 4.1.4
	Système de mesures	3.3.3, 4.1.5
	Tests	4.1.4, 4.1.1
	Caractérisation de la ventouse	4.1.3
Banc d'essai pendulaire	Conception et fabrication	4.2
	Exécution des tests et traitement des données	4.2, 3.3.3
Drone Ulysse	Conception du véhicule seul	3.3.1
	Assemblage et paramétrage	3.3.1
	Conception, assemblage et paramétrage du système offboard	3.3.2, 4.3.2
Système OptiTrack	Choix des caméras et installation	-
	Configuration et connexion	3.3.2

Légende

Mathis Celce	Bruno Chapdelaine	Mathis et Bruno

Tableau 1.2 Contributions aux différents modules de la simulation

Module de la simulation	**Sous module**	**Section du mémoire**
Dispositif d'Accostage (DA)	Modèle 3D	-
	Code du comportement	3.4.3
Drone Ulysse	Modèle 3D	3.4.2
	Paramètres moteurs	3.4.2
	Simulation capteurs	3.4.2, 3.4.3
Mise en place de l'environnement de simulation		3.4

Tableau 1.3 Contributions aux différents modules communs à la simulation et au monde réel

Module	**Sous module**	**Section du mémoire**
Stratégie d'accostage	Choix des capteurs	3.3.4
	Conception et écriture du code	3.5
Données de vol	Acquisition	3.6
	Traitement des données	3.6

1.3 Revue de littérature

1.3.1 L'inspection et la surveillance par véhicules aériens sans pilote

L'utilisation de véhicules aériens sans pilote fait ses preuves depuis plusieurs années pour les tâches d'inspection et de surveillance. Tout d'abord, l'emploi de véhicules aériens confère une plus grande mobilité et une meilleure visibilité des éléments situés en hauteur. Par ailleurs, les avantages justifiant le choix des véhicules sans pilote sont nombreux : le coût, la rapidité et l'aisance du déploiement, la discrétion, la manoeuvrabilité en zones exiguës, etc. Par exemple, la thermographie infrarouge par drone est employée dans [2] pour évaluer la performance énergétique de bâtiments. Aussi, les drones peuvent être utilisés pour surveiller le respect de la distanciation sociale entre les passants pendant une pandémie [3]. Ils peuvent être déployés en réseau adaptatif pour la couverture visuelle optimale d'une foule [4]. Le rapport Nergica publié en 2018 [5] établit que les inspections d'éoliennes pourraient être 5 fois plus rapides lorsque réalisées par des drones plutôt que par des techniciens équipés de cordes. De plus, le remplacement des techniciens pour les tâches d'inspection en hauteur – lignes électriques [6, 7], barrages, ponts [8], immeubles [9], éoliennes [10, 11] – ou à proximité de zones dangereuses, contaminées par la radioactivité [12] ou par des produits chimiques, est un facteur important pour la réduction des accidents de travail dans ces milieux.

1.3.2 Maintenance par robot mobile

De la même façon que l'emploi de drones permet d'optimiser les opérations d'inspection et de surveillance, il est possible d'utiliser des robots intégrant un bras robotique pour faciliter la maintenance de structures en zones élevées, difficiles d'accès ou dangereuses. Par exemple, la maintenance des éoliennes étant, avec l'étape de construction, la principale source de décès sur un cycle de vie [13], il semble pertinent d'encourager l'utilisation de robots pour l'entretien de tels ouvrages.

Le robot grimpeur Rise Rover développé en 2015 [14] est capable d'emporter une charge utile représentant jusqu'à deux tiers (7.3kg) de sa masse à vide (10.9kg). Son application couvrirait notamment la construction et l'entretien de structures, dans le but de préserver les travailleurs de tâches dangereuses mais aussi de réduire le coût de ces opérations (plus besoin d'installer d'échafaudages). Cependant, certaines situations peuvent représenter des difficultés d'accès, même pour ce type de robot : transiter entre 2 surfaces ayant des orientations trop différentes, atteindre une pale d'éolienne, faire face à des irrégularités – parapet, rigole – sur le trajet menant à l'objectif, etc. Pour cette raison il semble intéressant de se pencher sur l'utilisation de robots aériens, pouvant porter des charges souvent moins importantes mais plus rapides pour accéder au lieu de travail et capables d'éviter les obstacles.

Cela fait plusieurs années que, au vu du succès des véhicules aériens sans pilote pour les applications d'inspection et de surveillance, l'emploi de drones pouvant physiquement interagir avec l'environnement est exploré. Le nettoyage de fenêtre [15], le déplacement d'un drone en appui contre une paroi verticale [16], la saisie et le transport d'objet [17], le maintien d'un effort important sur une structure verticale [18] en sont autant d'exemples. Plus particulièrement, la combinaison de bras robotique et de drones est envisagée [19–23]. Par exemple en 2020, un prototype de drone muni d'un bras robotique est présenté, capable d'intervenir en vol sur des tours 5G pour effectuer des tâches d'inspection et de réparation.

Que ce soit avec un bras robotique articulé ou avec un effecteur fixe, l'utilisation des manipulateurs aériens requiert l'emploi de contrôleurs prenant en compte le drone et la charge utile. Il s'agit en effet de stabiliser le drone dont la dynamique est perturbée par le déplacement du centre de gravité, les forces de réaction de l'environnement ou encore la dynamique propre du bras embarqué. Cet ajout de perturbations réduit la précision des manipulations et complique le pilotage du drone. Ces problèmes de contrôle sont traités en général soit selon une approche centralisée [24–26], dans laquelle le bras et le drone sont considérés comme une seule entité, un seul corps, soit selon une approche décentralisée [27] lorsqu'ils sont considérés indépendants. Cependant, la faible autonomie de vol des drones, dépassant difficilement la demi-heure [28] et empirant avec la masse de la charge utile, est un véritable handicap que

les travaux sur le contrôle évoqués ci-dessus ne résolvent pas. Aussi, puisque le drone n'a que ses moteurs pour supporter les forces de réactions du travail effectué, le fait d'être en vol limite l'intensité maximale des efforts applicables.

Pour remédier à ces deux obstacles à la maintenance aérienne, il peut être intéressant d'étudier des moyens de faire s'attacher le drone à la zone cible, à l'image d'un cordiste qui s'attache solidement à son espace de travail ou d'un oiseau qui se perche pour fabriquer son nid. Cette technique s'applique aux nombreuses tâches pour lesquelles l'opération se restreint à une zone localisée, pendant un certain temps.

1.3.3 Le perchage de drones

Pour pallier au problème de la durée des missions limitées par l'autonomie de vol, de nombreux chercheurs envisagent d'accoster le drone sur la structure de travail, permettant ainsi d'éteindre les moteurs et d'économiser l'énergie. L'accostage de drone permet alors aussi de se passer d'algorithmes de contrôle devant prendre en compte les dynamiques du drone et de son bras robotique. Une fois le drone amarré à la structure de travail, l'application d'efforts importants devient également plus facile.

Dispositif d'amarrage

Dans [29], les stratégies d'accostage de véhicules aériens sans pilote à des structures sont répertoriées. Les différents systèmes d'amarrage et les manoeuvres d'accostage envisagées pour des véhicules allant de 20 g à moins de 2 kg sont catégorisés. Face aux véhicules à aile fixe, l'usage des multirotors est préférable pour leur plus grande maniabilité en zones difficiles d'accès. Associée à cette catégorie de véhicules, on trouve l'emploi de ventouses, d'aiguilles, d'adhésifs et de mâchoires.

Du Velcro est par exemple utilisé dans [30], et placé sous le ventre d'un drone d'environ 0.5 kg, pour lui permettre de s'accrocher à une paroi. Évidemment, cette technique nécessite de modifier la zone visée au préalable. Dans [31], les auteurs utilisent des aiguilles pour accrocher un véhicule aérien très léger en percutant une paroi verticale d'une dureté allant du balsa au mur de béton peint. S'il est possible de s'accrocher à une grande variété de surfaces, les surfaces lisses restent exclues. De plus, la masse du drone est très limitée et l'emport d'un bras robotique devient impossible. Une mâchoire équipée d'aiguilles est présentée dans [32] pour permettre à un drone d'accoster sur des parois rocheuses martiennes. Le dispositif ultra léger permet de supporter une force allant jusqu'au kilogramme. Les micro aiguilles utilisées s'usent et n'autorisent qu'un nombre limité d'accostages mais permettent d'accoster sur

des surfaces poussiéreuses et très irrégulières. L'emploi de serres mécaniques reproduisant le comportement des oiseaux [33] est aussi possible mais il est réservé à certaines formes de structures, et n'est pas adapté pour des surfaces essentiellement planes telles que pales d'éolienne, vitres d'immeuble, barrage, murs, etc. Un système d'accrochage à base d'adhésif sec pour un quadrotor d'environ 0.5 kg est présenté dans [34]. Ce dispositif permet un accostage réussi dès 0.4 m/s de vitesse d'impact mais se limite aux surfaces très lisses. Dans [35, 36], les auteurs font l'emploi d'une ventouse passive classique qui se fixe à la surface visée, lors de l'impact du drone. Comme le système adhésif de [34], cette méthode se limite aux surfaces suffisamment lisses. Elle nécessite une plus grande vitesse d'impact du drone sur la paroi, de l'ordre de 1 m/s, pour faire le vide dans la ventouse. Malheureusement, la passivité de ces mécanismes – hormis la serre mécanique – ne permet pas de garantir le maintien de l'accroche pour toute la durée de l'opération. À cet effet, un algorithme est mis au point dans [36] pour que le drone utilisé et pesant 2.45 kg rallume ses moteurs et exerce une force sur la ventouse lorsque la pression y devient trop faible. Malheureusement, cette manoeuvre risque de perturber les opérations d'un bras robotique embarqué. En revanche, l'emploi d'une ventouse active [37–40] permet de fiabiliser l'arrimage tout en gardant le drone immobile pendant que le bras robotique opère sur la structure. Aussi, cette approche permet de contrôler plus précisément la pression sous la ventouse qu'avec le rallumage ponctuel des moteurs proposé dans [36]. Le système d'attache avec ventouse active employé dans [41], ferait consommer 20 fois moins d'énergie au drone que s'il était en vol stationnaire. En utilisant le système de ventouses actives développé dans le brevet du BAM [1], il est possible d'accoster sur de nombreuses structures présentant des surfaces planes ou à grand rayon de courbure et de rugosité plutôt variable, allant de la pale d'éolienne à la paroi d'un barrage en passant par des vitres d'immeuble. Le drone employé ici étant plus léger que le drone Hercule (3 kg contre 20 kg), le DA utilisé dans ces travaux sera une version simplifiée du BAM, avec une seule ventouse.

Manoeuvres d'accostage

En se concentrant toujours sur les véhicules multirotors, les manoeuvres d'accostage explorées dans la littérature se divisent en 2 groupes. Les approches directes, par en haut [7, 42], par en bas [38, 43, 44] ou par devant [32, 34, 38–40] ne nécessitent pas de mouvement rapide du drone. La vitesse minimale est celle requise par le système d'attache pour être déclenché, qui peut être de seulement 0.4 m/s [34]. La faible vitesse permet d'avoir une plus grande précision au moment de l'accostage et donc d'accéder à des zones plus exiguës. À l'inverse, les approches dites «agressives» requièrent des vitesses angulaires et linéaires élevées à l'instar de l'atterrissage d'avions sur paroi verticale. Nécessaire par exemple pour attacher à une

surface verticale un DA qui se trouverait sous le ventre [30,45,46] ou sur le dos du drone [47], ce type de manoeuvre fait alors perdre au drone son avantage sur l'avion en zones exiguës et difficiles d'accès. Il semble alors plus pertinent de privilégier l'approche directe, en plaçant le DA vers l'avant du drone pour que la ventouse soit perpendiculaire aux surfaces verticales, sans avoir à effectuer une manoeuvre agressive. Plus largement, pour des accostages sur des surfaces inclinées entre 0° (sol) et 180° (plafond) le DA alors placé vers l'avant du drone pourrait facilement être adapté et orienté perpendiculairement à la surface. À noter qu'un accostage par le haut sur une surface plane horizontale ou inclinée de quelques degrés relève de l'atterrissage classique et n'a que peu d'intérêt ici.

Accostage autonome

Que ce soit à cause de la proximité d'obstacles, des perturbations aérodynamiques (vent ou effet de sol), de la précision de la zone d'accostage, de la visibilité entre le pilote et le drone ou des conditions météo, l'accostage peut s'avérer ardu pour l'opérateur aux commandes. Il devient alors nécessaire d'alléger la charge mentale du pilote en automatisant la manoeuvre d'accostage ou en réduisant le nombre de paramètres de vol dont doit s'occuper ce pilote. Automatiser cette manoeuvre devient même indispensable dans le cas de missions extra-planétaire, où il serait question par exemple d'accoster un drone sur une falaise martienne [32].

L'automatisation de véhicules aériens n'est pas une nouveauté et des solutions ont déjà été proposées pour l'accostage de drones. Les auteurs de [30, 45] présentent en 2010 L'automatisation d'une manoeuvre d'accostage agressive pour un véhicule avec une mâchoire équipée de griffes. Ils s'appuient sur le modèle dynamique du quadrotor pour réaliser leur propres contrôleurs, que ce soit pour le vol stationnaire ou pour la génération de trajectoires. Ensuite, ils affinent ces contrôleurs expérimentalement par itération, directement sur le terrain. De façon similaire, pour leur drone équipé d'un dispositif d'accostage à base d'adhésif sec, les auteurs de [34] établissent des stratégies de contrôle distinctes adaptées à la dynamique de chacune des séquences de la manoeuvre d'accostage, comme il est fait dans [36] pour un drone à ventouse passive. Les contrôleurs ainsi écrits sont ensuite réglés par essai erreur. Dans [43] les chercheurs automatisent le perchage d'un drone sous un cylindre horizontal, en se guidant par reconnaissance visuelle de son environnement. Ils établissent alors de façon analytique leur contrôleur basé sur l'image capturée par la caméra embarquée. Il est aussi possible d'automatiser le contrôle du drone seulement sur certains axes [7], pour l'atterrissage d'un drone sur une ligne électrique. L'algorithme se charge alors de garder le véhicule aligné avec l'axe du câble et à la verticale de celui-ci. Le contrôleur est établi en s'appuyant sur le

modèle dynamique du drone, comme il est fait dans [18, 34, 45] . Sans toucher au contrôle du drone lui même, et toujours dans l'optique de faciliter la tâche au pilote, les travaux présentés dans [44] peuvent être cités. Le sujet porte également sur le perchage sur une ligne électrique, mais par en dessous. Le système d'accroche du quadrotor d'environ 3 kg possède un degré de liberté (DDL) en rotation et un DDL en translation. Ce dispositif se positionne automatiquement pour permettre la préhension du câble par la pince. Ainsi, le pilotage du drone autorise un placement à ± 5 cm près, et l'angle autour de l'axe vertical du véhicule peut être quelconque.

1.3.4 Simulations de vol de drones

La simulation permet de se libérer de nombreuses contraintes du monde réel inhérentes au vol des drones. Comme le suggèrent de nombreux chercheurs [48–51], l'expérimentation sur les drones dans le monde réel pose de nombreux problèmes. Le risque d'accidents est élevé pour les plate-formes volantes, sans compter que la moindre erreur dans l'algorithme à tester peut entraîner un crash. Puisque chaque changement logiciel ou matériel nécessite des vérifications minutieuses, l'utilisation de la simulation est un gain de temps en plus d'une économie de moyens. Les exemples d'accostages mentionnés précédemment ne concernent que des drones ne dépassant pas 3 kg, pour lesquels les dégâts potentiels restent limités. En revanche, pour le drone Hercule qui totalise une vingtaine de kilogrammes avec son bras robotique, il devient indispensable de disposer d'un environnement de simulation pour développer un accostage autonome. La modélisation analytique, dont la plupart des travaux présentés dans le paragraphe «Accostage autonome» de la section 1.3.3 font usage, ne considère que la dynamique du drone. Or, la prise en compte des interactions avec l'environnement et la gestion des capteurs est indispensable si l'on veut réellement valider une méthode d'accostage autonome. Il n'existe pas de telle simulation dans la littérature, mais de manière plus générale, les plate-formes open source telles que OpenUAV [52] ou XTDrone [53] sont des exemples de simulation de drones complètes, auxquelles il convient de s'intéresser. En effet, elles utilisent le logiciel libre de simulation robotique, Gazebo, qui met à disposition un environnement de simulation prenant en compte les interaction physiques entre les robots et le monde. Gazebo permet aussi l'intégration de nombreux capteurs et autres extensions permettant de concevoir des robots utilisables dans le monde réel. Gazebo est souvent utilisé en association avec Robot Operation System (ROS), qui réunit un ensemble de composants et d'outils pour le développement de robots. ROS propose notamment un système de communication par publication et abonnement. Un noeud (*node*) peut publier des données sur un sujet (*topic*) auquel un autre noeud peut s'abonner pour y récupérer ces données. Le premier noeud peut correspondre par exemple à un capteur et le second noeud à un algorithme de contrôle qui effectue certaines

actions en fonction des données publiées par le noeud du capteur. Pour contrôler le drone dans Gazebo, l'auto-pilote open source PX4 est utilisé. Celui-ci possède un mode logiciel dans la boucle (Software In The Loop, SITL) exécuté sur l'ordinateur de la simulation, qui peut contrôler des véhicules simulés, comme il le ferait avec des véhicules réels. Selon les créateurs de OpenUAV et XTDrone, Gazebo est préférable à d'autres simulateurs tels que Airsim, Xplane ou Flight Gear. En effet, ceux-ci présentent certes des simulations de vol plus réalistes, mais ne sont pas adaptés à la robotique et Gazebo aurait une meilleure compatibilité avec ROS. Cette combinaison répandue de Gazebo, ROS et PX4 est notamment utilisée pour concevoir des tâches de cartographie et localisation simultanées [49, 54] (Simultaneous Localisation And Mapping, SLAM). Avec un tel environnement de simulation, les changements de configuration du drone et de son environnement sont facilités. Par ailleurs, Gazebo ainsi que le pilote automatique Ardupilot sont utilisé pour valider une solution d'inspection autonome de pale d'éolienne [10]. Suivant la même logique de s'affranchir des contraintes du monde réel qui pèsent sur l'expérimentation avec des drones, [55] fait appel à Airsim pour la simulation– toujours avec ROS et PX4 – argumentant que son rendu visuel est plus réaliste et qu'il est plus adapté pour construire des scènes complexes. De leur coté, les auteurs de [56] font appel à Morse Simulator pour le développement d'une solution d'évitement d'obstacles pour un essaim de drones. Selon eux, Morse serait plus adapté que Gazebo aux simulations avec plusieurs robots mais il faut noter qu'il n'est plus mis à jour depuis 2020. De la même façon, dans [48] les auteurs réalisent leur propre environnement de simulation, pour permettre aux concepteurs de se concentrer sur la programmation haut-niveau. Le domaine d'application visé est notamment les missions collaboratives impliquant plusieurs drones. Dans le même champ, les chercheurs de [50], utilisent la combinaison Gazebo et ROS, et créent On leurs propres plugins afin de rendre plus réalistes les interactions de saisie et de ventousage d'objets.

Depuis peu, le simulateur robotique Issac Sim [57] , basé sur la plateforme Nvidia Omniverse, propose un environnement ultra-réaliste pour le test et l'entraînement de robots interagissant avec le monde qui les entoure. Leur objectif est de réduire au minimum l'écart entre la simulation et la réalité. L'avantage vanté est la possibilité pour plusieurs ingénieurs de travailler en même temps sur un seul et même prototype, chose impossible avec un prototype réel. Cependant, il n'est pas encore fait mention de robots volants.

1.3.5 Simulations de ventouse

L'utilisation de ventouses en robotique est répandue pour la saisie d'objets ou encore pour les robots grimpeurs. Il est possible de simplement modéliser la force de succion à l'état

statique en multipliant la surface de la ventouse par la pression relative sous la ventouse. Il faut aussi prendre en compte la force élastique de la partie déformable de la ventouse, qui s'oppose à la force de succion. Ensuite, la force tangentielle maximale avant glissement peut être déterminée avec la loi de Coulomb [3] [58], à condition de connaître le coefficient de frottement de la ventouse sur la paroi.

Pour prévoir le comportement et la force maximale que peut endurer une ventouse avant son décrochage, des modélisations de ventouses ont été proposées. Un modèle analytique de ventouse pour les robots nettoyeurs de vitres a été présenté dans [59] et ce modèle prend en compte les déformations élastiques de la ventouse qui, en s'écrasant contre la paroi, voit sa surface active diminuer. Cependant ce modèle se restreint aux cas où la force appliquée sur la ventouse est parallèle à la surface d'accroche. Dans [60] les auteurs classifient les zones d'une ventouse active en fonction de leur contact. Leur simulation par éléments finis leur permet de calculer la déformation de la ventouse et les forces élastiques en son sein. Malheureusement cette méthode requiert une importante puissance de calcul pour une simulation en temps réel [4] et ne prend pas en compte les forces d'adhésion, qui jouent un rôle important au vu de la verticalité de la paroi. Dans [61], un modèle pour ventouse active est proposé. Ce modèle permet de prévoir la création de l'étanchéité entre la ventouse et la surface, ainsi que d'estimer la charge maximale supportable par la ventouse pour des surfaces plates et courbes. L'article présente également la mise en oeuvre du modèle dans le simulateur robotique CoppeliaSim.

Cependant, ces modèles «statiques» excluent la phase critique de l'accostage contre la paroi. En effet, de nombreux paramètres entrent en jeu dans le comportement de la ventouse à l'impact tels que la raideur de la partie déformable de la ventouse (mousse ou jupe), la rugosité de la paroi accostée, la rugosité de la surface de contact de la ventouse, l'angle entre les plans de la ventouse et de la paroi, la vitesse d'accostage, etc.

1.4 Problématique

La problématique consiste à trouver comment mettre en place un environnement de simulation dans lequel il est possible d'accoster un drone. Cet environnement doit permettre d'explorer les moyens d'automatisation de l'accostage en jouant sur le choix des capteurs, la conception des algorithmes et sur la configuration de l'environnement. Il sera alors également possible de tester des situations critiques du monde réel telles que des rafales de vent, une panne du système d'accostage, le défaut d'un capteur et ainsi de prévoir des solutions

3. $F_t = \mu \cdot F_n$
4. 11 ips avec : Intel Core i7-7820HQ, CPU à 2.90GHz, 32GB de mémoire système DDR4, et une carte graphique NVIDIA Quadro P3000

adaptées à ces problèmes. Il faudra bien entendu valider le développement de cette stratégie d'accostage en effectuant des accostages dans le monde réel.

1.5 Objectifs

Pour mettre en oeuvre cette simulation et démontrer qu'elle remplit bien sa fonction d'environnement favorable au développement de la stratégie d'un accostage autonome, les objectifs suivants ont été identifiés :

1. Mettre en oeuvre un dispositif d'accostage (DA) capable de s'arrimer sur des parois verticales à porosité variée allant du mur de béton, à la vitre d'immeuble en passant par la pale d'éolienne. Le système d'attache doit être fiable et il doit être possible de pouvoir prévoir son décrochage.
2. Mettre en place un environnement de simulation pour multicoptère incluant un moyen de reproduire le comportement du DA
3. Définir dans la simulation une stratégie d'accostage autonome et sélectionner les capteurs nécessaires
4. Assurer la portabilité de cette stratégie vers le monde réel
5. Implémenter et valider la stratégie d'accostage en simulation
6. Valider expérimentalement la stratégie d'accostage
7. Évaluer la fidélité l'environnement de simulation en comparant les accostages réel et simulé

CHAPITRE 2 DEMARCHE DU TRAVAIL

L'objectif principal est de mettre en place un environnement de simulation pour pouvoir développer une stratégie d'accostage autonome de drones multi-rotors.

Les applications potentielles d'un accostage sur structure amènent cette manoeuvre à être effectuée dans des zones difficiles d'accès et donc souvent hors du champ de vision du pilote. Les chercheurs s'accordent sur la nécessitée pour cette manoeuvre ardue par nature d'être exécutée de manière automatisée. Si l'accostage autonome de drones existe dans la littérature scientifique, cela se restreint à des véhicules ne dépassant pas les 3 kilogrammes. Or, le drone Hercule, qu'il est question de percher sur des structures pour des tâches de contact avec bras robotique, pèse une vingtaine de kilogrammes. Dans ce contexte, il devient nécessaire d'utiliser un environnement de simulation pour mettre au point l'accostage autonome du véhicule, au risque de perdre beaucoup de temps et de moyens à réparer des dégâts occasionnés par de potentiels crashs. Cependant, il n'existe pas de simulation d'accostage de drone qui permettrait de développer de A à Z une stratégie d'accostage autonome. L'article scientifique présenté au chapitre 3 propose d'y remédier.

L'article répond aux principaux objectifs identifiés de la façon suivante :

- Le véhicule expérimental qui sert à tester la manoeuvre d'accostage est présenté. (Objectif 1.)
- Le choix de l'architecture logicielle de la simulation est établi à partir de la revue de littérature sur les simulations de drones. (Objectif 2.)
- Pour simuler l'accostage, le DA utilisé est caractérisé expérimentalement et modélisé dans la simulation. (Objectifs 2. et 4.)
- Pour permettre la validation de l'environnement mis en place, l'article s'attache à présenter une stratégie d'accostage fonctionnelle ainsi que son exécution expérimentale. (Objectifs 3., 5. et 6.)
- Enfin, l'efficacité de la simulation d'accostage est constatée en analysant les données d'accostages du monde réel en regard de celles de la simulation. (Objectif 7.)

Les autres éléments nécessaires à la réalisation et à la validation de l'objectif principal, consistant à mettre en place une simulation d'accostage de drones, sont présentés dans le chapitre 4.

CHAPITRE 3 ARTICLE 1 : AUTONOMOUS PERCHING AND TAKEOFF OF MULTIROTOR UAV ON VERTICAL SURFACES

Cet article a été soumis le 21 juillet 2022 pour publication dans la revue scientifique Journal of Intelligent and Robotic Systems.

Mathis Celce, Bruno Chapdelaine, Bruno Monsarrat, Charles Vidal, Lionel Birglen

3.1 Abstract

In this paper, a strategy to autonomously perch on a surface a multi-rotor vehicle equipped with a vacuum cup is developed. Perching is a known solution to extend the mission duration of unmanned aerial vehicles (UAV) used for inspection, monitoring or more recently, maintenance. To reduce operator workload, which could be very high in difficult configuration, it is necessary to automate perching. However, the experimental implementation of such a manoeuvre raises issues on safety, budget, and time especially for heavier UAVs such as the aerial manipulators dedicated to maintenance. To solve these problems, a widely used simulation environment can be successfully customized as will be shown in this paper. Autonomous manoeuvres can then be entirely developed in this environment. To minimize discrepancies, many improvements are proposed in order for the autonomous perching algorithm to be directly transferable into the real world. Finally, experimental flights are shown to validate the efficiency of the customized simulation environment to securely automatize a perching manoeuvre. Comparisons between the simulation and actual experiments clearly demonstrate a fairly accurate match for the perching manoeuvre. However, the deviations identified for the takeoff and detachment manoeuvre from the wall emphasize the requirement for a more comprehensive aerodynamic model to better predict the behaviour of the actual drone.

3.2 Introduction

The use of Unmanned Aerial Vehicles (UAVs) for inspection and maintenance has seen a tremendous increase in the past decade. Their manoeuvrability and ease of deployment allow them to efficiently reach hard-to-reach structures which would be very difficult to access by a person. UAVs are used to inspect buildings [9], dams [8], power lines [6, 7], railway infrastructure [62], and the interior or exterior of wind turbine blades [10, 11]. According to a 2018 technical report from Nergica [5], turbine inspections could be up to 5 times faster with UAVs than with rope access technicians. Similarly, it has been proven that the frequent

maintenance operations required for wind turbines [63] or other similar structures can be cost-effective by using UAVs. To achieve this type of operations, multi-rotor vehicles could be equipped with an embedded robotic manipulator [24], a configuration referred to as aerial manipulators (see Fig. 3.1 for an illustration), as demonstrated maintenance in [26] for 5G mobile towers. Unfortunately, during missions where the UAV remains static most of the time, their limited flight time becomes a major issue. Additionally, relying on aerial manipulators also raises control issues because of the potential harmful interaction forces between the structure and the robot end-effector [21, 24].

To extend the operating time of aerial manipulators, UAVs can interact with environmental structures by leaning [33] or perching on them and thereby, significantly reduce or even eliminate the energy consumption of the rotors. In [29], an exhaustive inventory of perching strategies ranging from cylinder grasping to electrostatic adhesion was shown. Different families of approaches were identified according to the type of used gripper technology and the resulting perching manoeuvre. Amongst all these families, suction cup adherence is a physical implementation commonly found for perching as shown for example in [35] where a lateral suction cup was demonstrated to successfully perch a 1.8 kg UAV on a smooth vertical surface. However, passive grasping technologies such as relying on suction cups are often not suitable for reliable perching as they may unexpectedly detach. To secure perching, a UAV was shown in [36] where the rotors were used to press the cup when the vacuum was detected to run low and expel the air leaking inside the cup. This technique obviously hinders the energy saving of the perching operation. Additionally, suction cups are generally more suitable for very smooth surfaces such as glass and do not work well with concrete or other rough medias. A more energy efficient solution can be to use a vacuum cup, as shown in [39]. Similarly to the latter approach, an optimized vacuum cup was proposed in [40] to increase tolerances to imprecision in the perching manoeuvre. However, even if these works demonstrated improvements in securing a UAV to a surface, the perching manoeuvre still requires a very high workload from the operator, as it heavily depends on the situational and environmental parameters such as visibility from the ground, weather, configuration of the landing site, etc. Consequently, automating perching manoeuvres is an attractive avenue to ensure robust perching in Beyond Visual Line-Of-Sight (BVLOS) scenarios.

To this aim, an alignment algorithm to perch on a power line was discussed in [7] which reduces flight degrees of freedom to allow the operator to control only the height and the longitudinal position. Another work dealing with autonomous perching was presented in [34] for a Micro Air Vehicle (MAV) equipped with a dry adhesive gripper. Autonomous ag-

Figure 3.1 UAV perched on a wind turbine blade with the NRC's Base Attachment Module and deploying a robotic arm

gressive manoeuvre can also be shown to work with perching as demonstrated in [45] and even be able to recover from a failed perching attempt. However, relying on experiments to develop perching manoeuvres can be a very costly and time consuming approach due to the inevitable failures [64–66]. Therefore, an accurate simulation environment that could be adapted to many hardware and software solutions would be very helpful to develop an autonomous perching manoeuvre. While software simulations of UAV are very common, for instance to implement and validate autonomous simultaneous localization and mapping (SLAM) processes, often relying on Gazebo and ROS [49, 51], to the best of the authors' knowledge, accurate perching modelling is unavailable in the most common simulation softwares. This paper aims at filling this gap.

UAV simulations are often successfully demonstrated in the literature, e.g. in [10] where the feasibility of an autonomous blade inspection method through a Gazebo-ROS simulation was demonstrated. Open-source simulation platforms have also been developed as Open-UAV [52] and XTDrone [53], both relying on a package of three main components, namely ROS, Gazebo, and PX4. On the other hand, a quad-rotor simulation environment based on Airsim, PX4, and ROS shown in [55] highlighted that Airsim is more suitable to build complex scenes and has a more realistic visual rendering. However, Gazebo is often preferred to other simulation softwares such as Airsim, Xplane, or Flight Gear, as in this work, because of

its better ROS support and more robotics-oriented approach. To control UAVs in simulations, the use of Ardupilot [10,56] or PX4 [51–53,55] as autopilots is also typically preferred. Since the primary motivation of this work is to diminish the time, resource, and efforts, creating barriers to real world perching, customizing the common simulation environment provided by Gazebo-ROS-PX4 is arguably the best approach. Therefore, a perching strategy using these software packages will be conceived in simulation in this work and experiments will be shown to illustrate the accuracy that can be achieved following the proposed methodology. Departing from the previously mentioned simulation approaches to automate perching, the simulation proposed here will include all aspects of the manoeuvre from flight dynamics, sensors, to complex environments.

The proposed software and mechanical systems are first presented in Section II. Then, Section III introduces the required customization for accurate perching simulation. An autonomous perching strategy is subsequently detailed in Section IV. Finally, results from both experiments and simulations are compared in Section V.

3.3 UAV Setup and Perching Manipulator

3.3.1 UAV Dynamic Model and Characteristics

Figure 3.2 shows the general coordinate systems used in this work. The world reference frame is defined by the $\{\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3\}$ vectors and the UAV body-fixed frame by vectors $\{\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3\}$. The latter body frame is specified in the north-east-down coordinate system and its origin is placed at the centre of gravity of the UAV. Roll, pitch and yaw Euler angles (ϕ, θ, ψ respectively) describe the rotations around the axes of the body frame and its orientation in space as shown in Fig. 3.2. The equations of motion of the UAV are given by:

$$m\ddot{\mathbf{x}} = \mathbf{R}_B^W \mathbf{F} \tag{3.1}$$

$$\mathbf{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\mathbf{I}\boldsymbol{\omega}) = \mathbf{M} \tag{3.2}$$

with:

— $\mathbf{x}$ the position of the UAV frame in the world frame,
— $\boldsymbol{\omega}$ the angular velocity of the body-fixed frame,
— $\mathbf{R}_B^W$ the rotation matrix from the UAV frame to the reference frame
— $\mathbf{F}$ and $\mathbf{M}$ the total force and moment vectors acting on the body-fixed frame,
— $\mathbf{I}$ and m the inertia matrix and the mass of the UAV respectively.

The UAV used as an example throughout this paper is a custom-built quadrotor UAV named

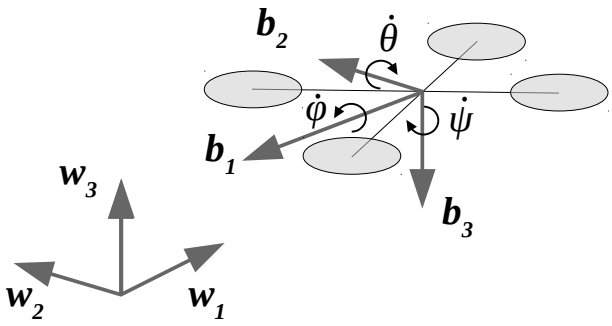

Figure 3.2 Inertial frame and body fixed frame, with Euler angles

Ulysses that is 58 cm wide and with a dry weight of 1.7 kg. It is built from a ZD 550 carbon fiber quadcopter frame. With all its equipment (i.e. companion computer, sensors, and perching arm with its pneumatic system) it weights a total of 3.2 kg. Ulysses uses Sunnysky V3508 380KV brushless motors with Tarot T series 1255 (TL2828) carbon fiber propellers. A 6S Turnigy battery with a capacity of 4000 mAh weighting approx. 600 g is also included. As shown in Fig. 3.3 which depicts the global software architecture of the UAV, a Pixhawk 4 autopilot runs PX4 and Ulysses includes a Raspberry Pi 4 4 GB running Ubuntu 20.04 as a companion computer to fly autonomously.

3.3.2 Automated Flight Framework

In this paper, the PX4 open source autopilot is used to control the UAV and combined with ROS to automate the perching sequence. The Robot Operating System (ROS) is an open source messaging system providing many tools under the form of packages, ranging from data visualization to robot interfacing. ROS code is compatible with both Python and C++ programming languages. ROS nodes are communicating between themselves through topics, using a publication/subscription protocol. A node can publish on and subscribe to various topics. The MAVROS ROS package allows communication between the PX4 flight controller and other ROS nodes. For instance, a position setpoint published on a ROS topic by a perching algorithm can be converted into a MAVLink stream and sent to the autopilot. MAVROS greatly simplifies the control of the UAV since low level control is handled downstream by the PX4 autopilot, not by the user. Therefore, focus can be kept on the perching strategy without having to develop specific control schemes.

It must be noted that experiments conducted in this paper are done indoor in a flight room where GPS signals cannot be used to determine the UAV position. Consequently, to precisely

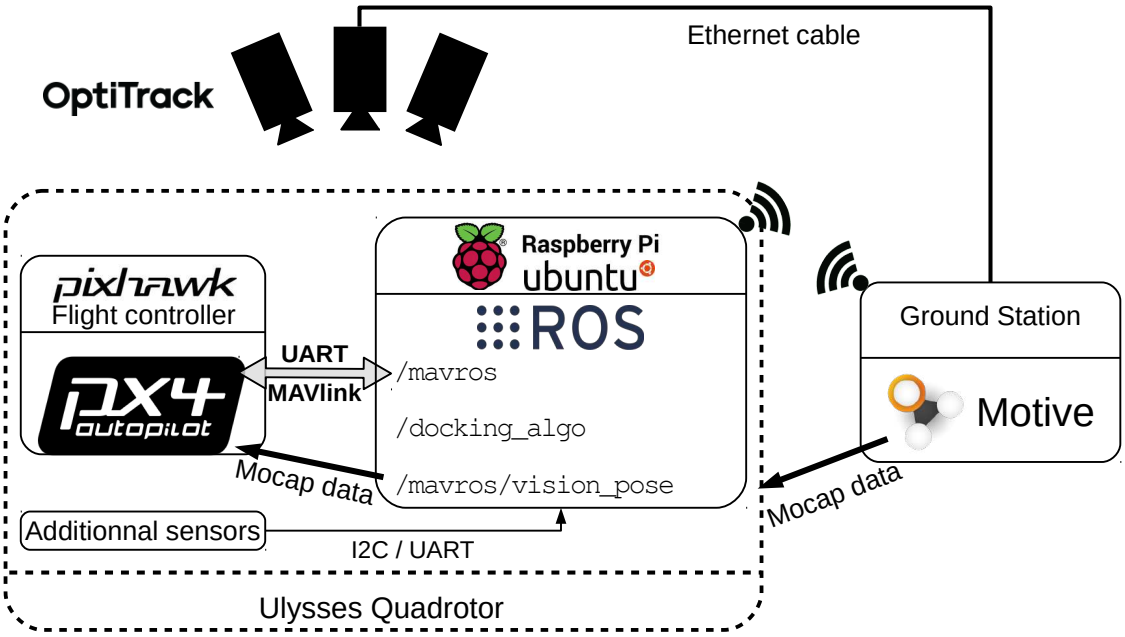

Figure 3.3 Flight environment and architecture of the UAV showing the companion computer and the motion capture system

move the UAV to a given set of coordinates, an OptiTrack motion capture system is used. As shown in Fig. 3.3, cameras from OptiTrack spatially locate trackers placed on the UAV and the Motive software then estimates the pose (position + orientation) of the vehicle. This system is far more precise than a GPS, effectively providing a millimeter accuracy. The motion capture software Motive is running on the ground control station and streams the pose estimation by UDP to the Raspberry Pi companion computer via WiFi. The motion capture data is then remapped from the `/vrpn_client_ros` topic to the `/mavros/vision_pose/pose` topic before being fed to the Extended Kalman Filter (EKF2) state estimator of the PX4. Docking algorithms can be run on the onboard computer and get data from MAVROS as position, or attitude, and the computer can get data from additional sensors such as pressure, force, or distance to a wall as well. The docking algorithm can also send setpoints to the PX4 through MAVROS to autonomously perch the UAV. During the actual experiments, all the sensors elected from the simulation (see Section 3.3.4) were communicating with the Raspberry Pi onboard computer using Universal Asynchronous Receiver-Transmitter (UART) or Inter-Integrated Circuit (I2C) protocols and Python scripts are retrieving the associated data stream and publishing them on ROS topics, as illustrated in Fig. 3.3.

3.3.3 Perching Apparatus

In order to perch on vertical surfaces, an UAV must possess a perching apparatus. With the machine considered as an example in this work, the frontal docking system presented in Fig. 3.4 is concieved. This perching device includes a vacuum cup composed of a rigid 3D printed disk and a foam ring making contact with the wall where the UAV must be attached. The rigid disk is attached to a spring loaded sliding mechanism with two carbon fiber tubes fixed to the UAV. The purpose of this slider is to prolong the contact between the vacuum cup and the surface when they impact. This arrangement gives enough time for the vacuum pump to drop pressure before the UAV bounces back from the contact surface. The latter vacuum pump is electrically driven and located at the back of the UAV to balance the vehicle. The joint between the cup and the sliding mechanism has one rotational degree of freedom, depicted with a dash-dot line in Fig. 3.4, to allow the UAV to adapt its pitch to the surface when perching.

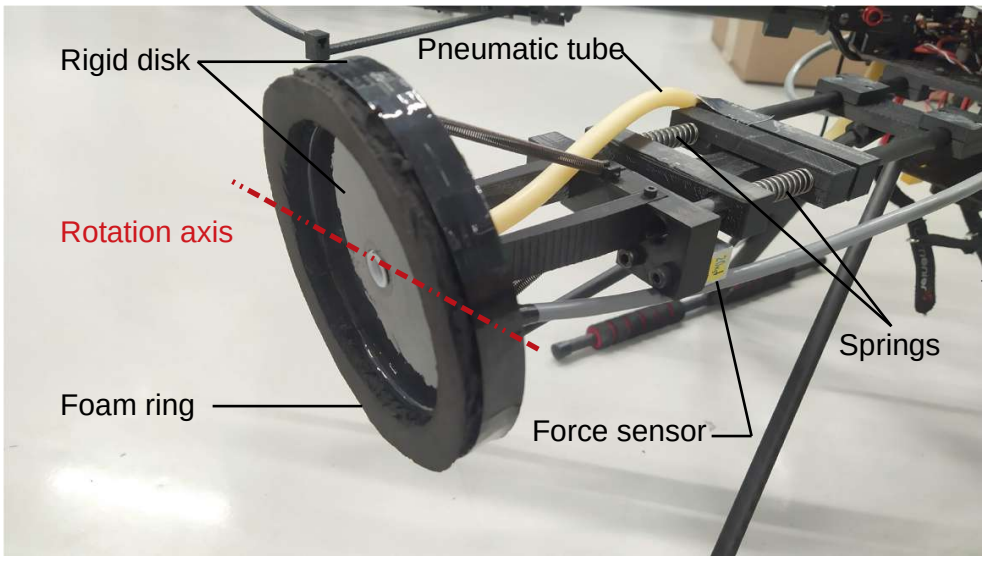

Figure 3.4 Prototype of perching device used for experiments

The vacuum cup relies on a 1cm thick closed-cell EPDM foam with high compressibility. The more compressible the foam is, the lower the impact force required to make a proper seal between the cup and the surface. Additionally, rough surfaces also benefit from the shape adaptation provided by the foam to conform to irregularities. However, a compromise must be found since with a very compressible foam, the joint established between the wall and the vehicle will also be less rigid which might be an issue for tasks requiring high precision, especially with an embedded robot manipulator conducting contact-based tasks.

Now that a perching device has been chosen, its characterization must be carried out. This characterization will be useful for establishing the ideal perching speed range but also to precisely model the behaviour of the perching device within the simulation. In our case, Fig. 3.5 shows the probability of a successful sealing of the perching apparatus on a vertical flat concrete wall as a function of the impact speed, obtained from experiments. A test bench was conceived to produce this data allowing to quickly conduct many perching manoeuvre without requiring to pilot the UAV each time. This test bench is composed of the Ulysses UAV itself with all its equipment and hanging from a five meters high pendulum. Ulysses can then be swung towards the target surface with the perching device forward and perpendicular to the wall. Pressure in the vacuum cup is subsequently recorded, as well as the force exerted on the load cell (force sensor in Fig. 3.4), and distance between the rangefinders (see Section 3.3.4) and the wall. The autopilot and the motors remain off during these experiments. A perching attempt is then considered successful if, after impact, the measured pressure falls below a threshold value indicating that a proper seal has been established. Otherwise, it is deemed a failure. According to Fig. 3.5, showing the data from 253 perching attempts, the best results are obtained for perching velocities between 1 and 1.2 $m.s^{-1}$. Faster perching speeds could also be considered to determine at what velocities the probability of success begins to decrease.

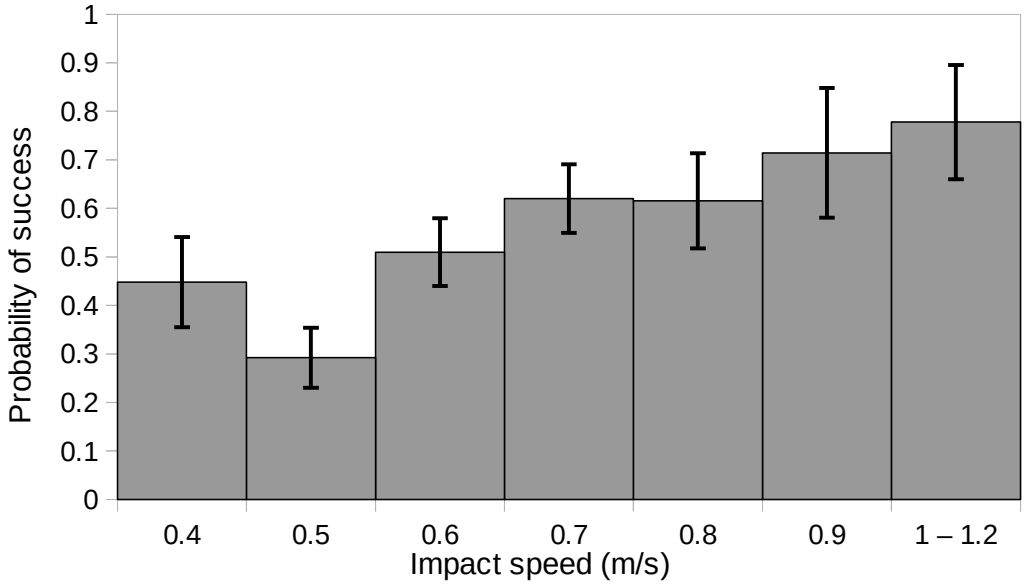

Figure 3.5 Probability of a successful attachment of the perching device relative to the impact speed

The probability of a successful perching may be determined relatively to impact speed or impact force. The choice is made here to associate the probability of success to speed rather than the force at impact. Indeed, impacts are by nature discrete and very short events which are difficult to record accurately especially due to the low sampling rate of the load cell in our case.

3.3.4 Sensors for Autonomous Perching

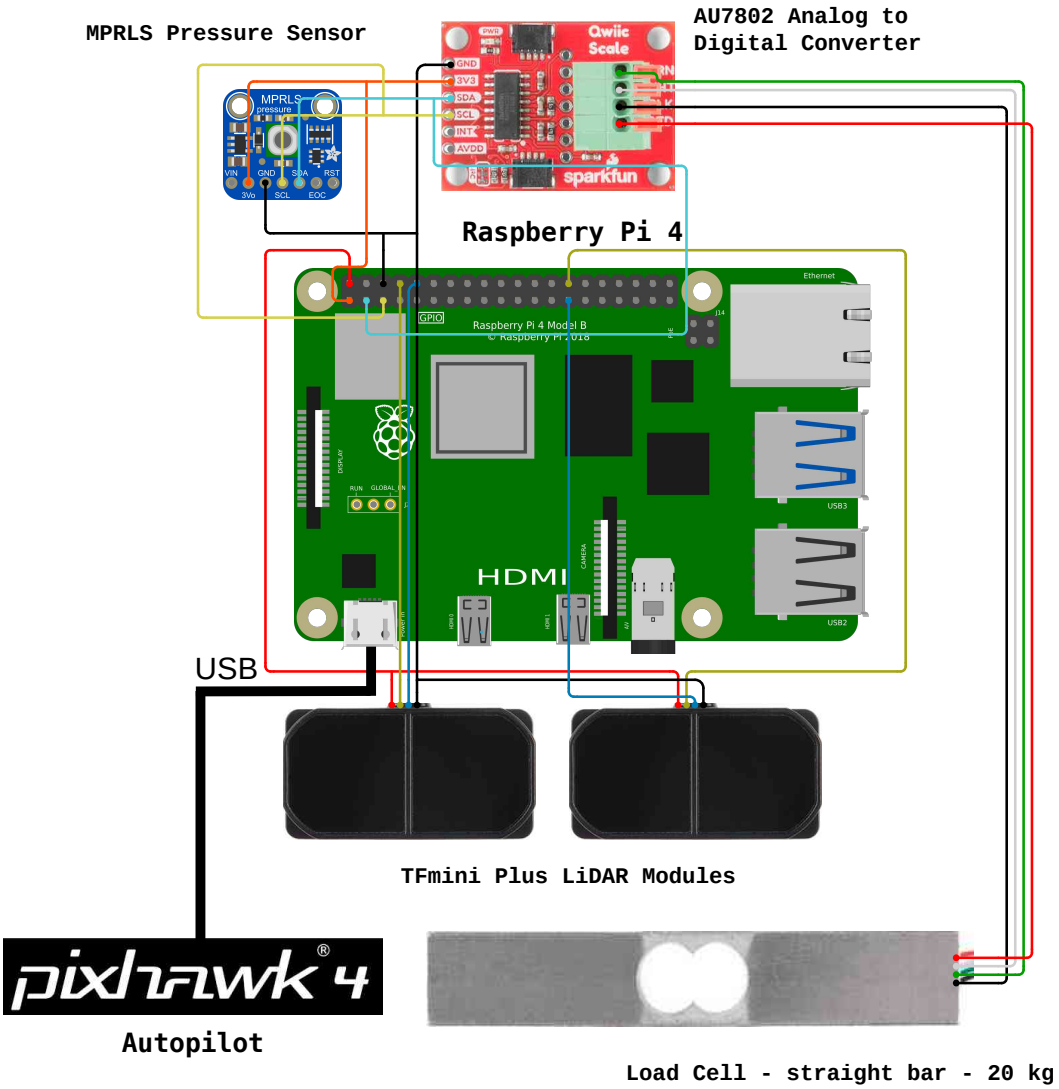

Figure 3.6 Sensors wiring with the Offboard computer

To facilitate autonomous perching, additional sensors can be used and in this work, two rangefinders, a straight bar load cell, and a pressure sensor were selected. The interfacing

of the sensors with the onboard computer is illustrated in Fig. 3.6. The rangefinders are plugged to the Raspberry UART ports 0 and 2 through the General Purpose Input/Output (GPIO) pins 14/15 and 0/1. The load cell is wired to an Analog to Digital Converter (ADC) and this ADC, along with the pressure sensor, is connected to the I2C bus of the Raspberry located at GPIOs 2 and 3. The Pixhawk is linked to the onboard computer *via* a Universal Serial Bus (USB). These sensors were tested with the perching algorithm in the simulation to validate this selection. The perching manoeuvre of the UAV is assumed to start from a position selected by an operator roughly in front of the targeted perching area and then, the perching algorithm would handle to final sequence. This means Ulysses will have to place itself perpendicularly to the contact surface and at a set distance before executing the perching sequence which must be done through sensing. In order to be adequately positioned relatively to a given surface, the UAV is equipped with two Benewake TFmini Plus single-point LiDARs, one on each front arm, with the laser rays colinear to $\mathbf{b_1}$. Using this type of dual rangefinders, the angle between the vertical surface and the vacuum cup, considering Ulysses horizontal, is given by :

$$\alpha = \arctan \frac{d_1 - d_2}{l} \tag{3.3}$$

where d_1 and d_2 are the distances to the wall as measured from the left and right arm sensors respectively and l is the fixed distance separating them by construction. The distance between the UAV and the wall can be estimated simply by computing the average of d_1 and d_2 values. This setup is considerably simpler than a depth camera, used for instance in [67], for UAV autonomous navigation but gives less information on the configuration of the perching site. A straight bar load cell is available in the perching device of the experimental UAV between the revolute joint and the structure formed by the carbon fiber tubes. This strain gauge based sensor measures the total force exerted between the UAV and the vacuum cup. By monitoring when the force reaches its maximum during a perching sequence, the sensor allows to detect the critical moments of the manoeuvre which can be used to revise the design of the vacuum cup if needed. Additionally, in the perching algorithm a force sensor can inform the UAV that it has reached the targeted surface. The perching vacuum cup also embeds a pressure sensor capable of monitoring absolute pressure from 0 to 1724 hPa (25 psi). It is indispensable to provide the UAV information on whether it has managed to attach to the surface or not before starting the landing sequence on the wall. Once landed, the pressure information may also be used to indicate when the pump must be activated to maintain a pressure low enough to securely keep the cup attached to the wall. Finally, during takeoff from the wall, the pressure sensor can be used to establish if the cup has detached, indicating to the UAV that it can proceed with the next step of the manoeuvre.

3.4 Customization of the Simulation Environment

As mentioned before, to develop a strategy for autonomous perching at a lower risk and cost, an accurate simulation environment must be established to replicate the physical behaviour of the setup presented in Section 3.3. To this aim, the perching algorithm must produce largely identical results whether the UAV is simulated or in the real world. Therefore, the environment must take into account all the important elements that the experimentally running script interacts with.

3.4.1 Software Architecture

The simulation environment, running under Ubuntu 18.04, is composed of the ROS-PX4 combination completed by the open-source Gazebo robotics simulator, see Fig. 3.7. Gazebo handles the dynamic simulation of the vehicle, its environment, and the interactions between them. Collisions between objects are managed by this software and many models of sensors are available with realistic models. Additionally, the real PX4 autopilot can be run in a Software-In-The-Loop (SITL) configuration and thus, can also be used in the simulation environment. PX4 SITL is a modified version of PX4 specifically designed to run on a computer for simulation. When operating along with Gazebo and MAVROS, PX4 SITL can control a simulated UAV in the software exactly as PX4 would do with an actual UAV. To achieve this functionality, PX4 must connect to the local TCP port of the simulation software and communicate through this port with Mavlink, as illustrated in Fig. 3.7. Furthermore, PX4 can also connect to ROS with MAVLink via UDP to exchange flight data and setpoints with an autonomous flight module, exactly as it occurs with a real system.

3.4.2 Simulated UAV model

With the simulation overall software platform set up, the Ulysses UAV that will be used for experimentation must be integrated with all its important physical parameters such as inertia, motor and propeller characteristics, etc. In Gazebo, vehicles are parameterized in SDF files using the Simulation Description Format. These files contain a description of the links and the joints of any robot with physical specifications such as mass, collision box, inertia matrices, friction and damping coefficients in joints, etc.

In our case, the actual value of the inertia matrices must first be found. To this aim, the easiest method is to model the shape of the UAV in a Computer-Assisted Design (CAD) software and specify the mass and the geometry of each component. Then, the obtained

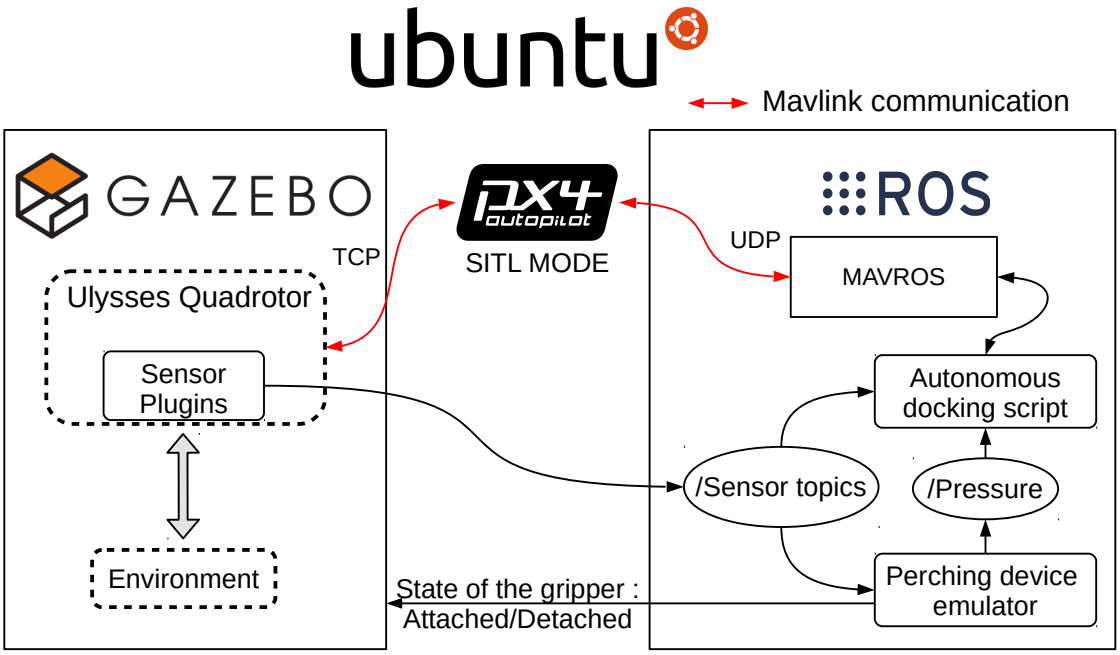

Figure 3.7 Software architecture of the simulation environment

CAD model can be converted into a URDF file (Universal Robot Description Format) using plugins available with most CAD packages, e.g. "SW2URDF" [1] in SolidWorks. Gazebo is then capable of importing these URDF files and converting them into SDF on the fly. The parameters of the *gazebo_motor_model* plugin, which simulates the lift generated by the propellers, also have to be set to match the actual motors and propellers. The `motorConstant` and `momentConstant` parameters (k_{mot} and k_{mom} respectively) as designated in the vehicle file can be computed according to standard propeller models [2] with :

$$k_{mot} = \frac{C_{T0}\rho D^4}{4\pi^2} \tag{3.4}$$

$$k_{mom} = \frac{C_{Q0}}{C_{T0}} D \tag{3.5}$$

where:

- C_{T0} and C_{Q0} are the static thrust and torque coefficients,
- D is the propeller diameter,
- and ρ is the air density.

Parameters C_{T0} and C_{Q0} can be conveniently determined thanks to the UIUC database [3].

1. SolidWorks to URDF Exporter
2. See http://web.mit.edu/16.unified/www/FALL/thermodynamics/notes/node86.html
3. https://m-selig.ae.illinois.edu/

Another parameter of importance, namely `rotorDragCoefficient`, can be taken from available literature from UAV manufacturers. In our case the DJI matrice 100 provided among the Gazebo models matches the size of our experimental UAV and was used to set that parameter. The edited motor parameters are listed in Table 3.1. The values in the latter table are compared to these of Iris, the 1.6 kg default quadrotor proposed in Gazebo.

Table 3.1 Gazebo motor model parameters

	Iris quadrotor	Ulysses UAV
`motorConstant` ()	5.84×10^{-6}	2.5×10^{-5}
`momentConstant` (m)	6.00×10^{-2}	1.9×10^{-2}
`rotorDragCoefficient`	1.75×10^{-4}	8×10^{-4}

3.4.3 Simulated Perching Device

Next, the perching device must also be accurately included to the simulation as well. To this aim, a script named “Perching device emulator", see Fig. 3.7, was created to effectively replicate the behaviour of this perching device. To emulate the attachment of the vacuum cup to the surface, the ROS plugin *gazebo_ros_link_attacher* is used and creates in Gazebo a joint between two specified links, here the vacuum cup and the surface. This joint creation is run when the “attachment service” is requested. The *gazebo_ros_link_attacher* ROS package relies on a request/reply interaction rather than subscribe/publish because the instruction to attach or detach the vacuum cup needs only to be sent occasionally. The “Perching device emulator" script can be seen as another ROS plugin and it calls the “attachment service" under certain conditions on the distance to a surface and on the force applied to the cup. Similarly, the “detachment service” is called to delete the joint previously created when the pump is stopped or when excessive forces are exerted on the vacuum cup. The algorithm of the plugin is summarized in Fig. 3.8.

The plugin also publishes on the `\Pressure` ROS topic an emulated pressure based on measurements from the actual vacuum cup used with the UAV. Figure 3.9 shows that the decrease of pressure in the vacuum cup is the slowest on porous concrete walls compared to painted cinder blocks or wind turbine blades. Similarly, the vacuum level reached with concrete is the worst of all the experimented surfaces. To model how the pressure changes (dashed lines in Fig. 3.9) after a successful perching attempt, the following relationships are proposed from

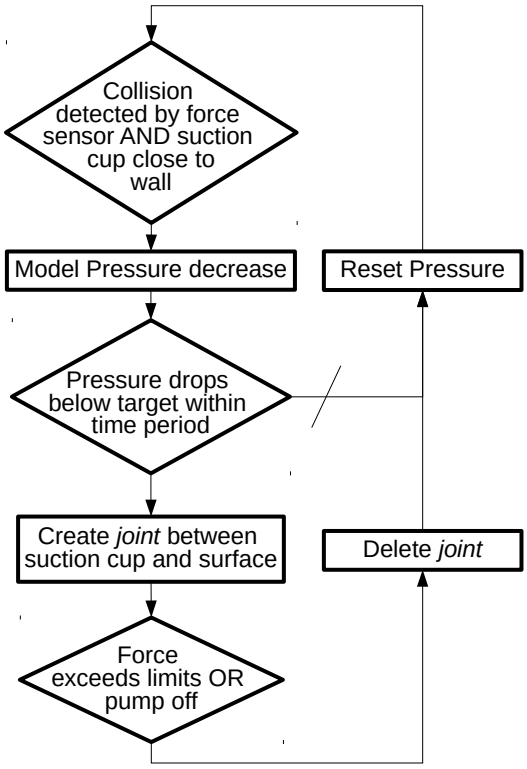

Figure 3.8 Diagram of the simulated attachment device

ground truth pressure measurements (solid lines with markers in Fig. 3.9) :

$$\text{Sealing} : P(t) = \mathrm{P_{vac}} + (\mathrm{P_{atm}} - \mathrm{P_{vac}})e^{-t/\tau_1} \tag{3.6}$$

$$\text{Unsealing} : P(t) = \mathrm{P_{atm}} - (\mathrm{P_{atm}} - \mathrm{P_{vac}})e^{-t/\tau_2} \tag{3.7}$$

where $P(t)$ is the instantaneous pressure in the vacuum cup, once the perching attempt is successful. After impact, the drone is considered to keep a neutral attitude such that little force is exerted on the perching device during sealing. P_{vac} is the minimal pressure when the vacuum cup is sealed on a surface, P_{atm} is the atmospheric pressure and τ_i $(i = 1, 2)$ are the time constants. Examples of actual values from experiments with Ulysses are listed in Table 3.2.

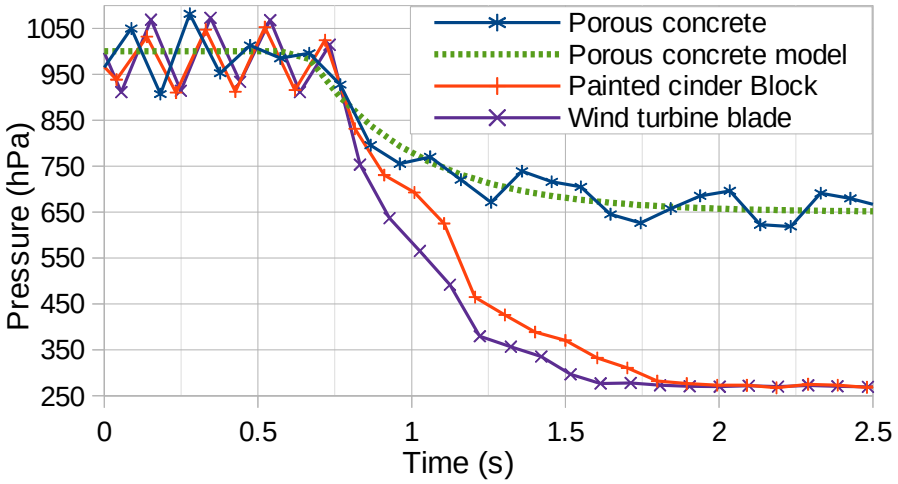

Figure 3.9 Pressure levels during sealing of the vacuum cup on various surfaces

Table 3.2 Characteristic values for the pressure model measured from experiments with sample surfaces

	Porous concrete	Cinder block	Wind turbine blade	Metallic surface
P_{vac} (hPa)	650	275	265	275
τ_1 (s)	0.33 ± 0.03	0.35 ± 0.03	0.25 ± 0.03	0.25 ± 0.03
τ_2 (s)	0.40 ± 0.03	7.5 ± 0.5	8.0 ± 0.5	16 ± 1

The pressure $P(t)$ is not required to run the simulation as the conditions to call the "attachment service" are only the distance to the contact surface and force on the latter (see Fig. 3.8). However, the pressure information can be used with an actual UAV in the perching

algorithm to validate its attached/detached status. Then, given that the goal of this simulation is to fully emulate the behaviour of the actual setup, the pressure data should be included in the simulation to ensure an accurate match between simulations and experiments.

Finally, the normal force required to detach the vacuum cup from the wall has been determined by manually pulling the actual perching system. The apparatus demonstrated the ability to handle at least 18 kg (177 N) and it was decided not to go further to preserve the integrity of the attachment device. Consequently, the limit of force that the simulated perching device can support before detaching from the surface will be set to 177 N.

3.5 Autonomous Perching Sequence

The simulation environment, now customized in order to accurately model the perching of a UAV on vertical surfaces using a vacuum device, can be used to establish a successful autonomous perching strategy. This algorithm is meant to start from a preselected location by a pilot, assumed to be near the targeted surface and conduct all required perching operations including the takeoff and detachment from the structure.

As illustrated in the lefthand side of Fig. 3.10, the UAV is first guided to a preliminary position, approximately perpendicular to the perching surface and at a fixed distance d_{pre} (GAP in Fig. 3.10) from this surface. The requirement for perpendicularity is preferable to maximize the chances of a successful perching and to avoid collision between a propeller and the structure. This relative orientation is measured by angle α, defined in Fig. 3.11 and ideally equal to 0 deg. This angle is taken between the vertical surface and $\mathbf{b}_1$, when Ulysses is considered horizontal, and is evaluated from lidar distances, see Eq. (3.3). A condition on α in Fig. 3.10 is set (namely $-2^\circ < \alpha < 2^\circ$) to take into account the uncertainty error $\Delta\alpha$ of the two lidars. According to the data sheet of the TFmini Plus lidar [4] and with the geometrical configuration of the two devices, $\Delta\alpha$ is estimated to 2 deg. in this work.

When the correct positioning is confirmed (e.g. by the lidars), the UAV can begin the actual perching sequence. During this stage, our proposed algorithm command it to go toward the wall at a constant speed, v_{dock} (DOCK_SPEED in Fig. 3.10). A speed setpoint is preferable to an attitude setpoint, as the controller will compensate for external disturbances such as wind and unlike a position setpoint, the impact speed can be imposed. For Ulysses, the ideal impact speed has been identified through experiments, see Fig. 3.5, namely 1.2 $\mathrm{m.s}^{-1}$. Once

4. https://cdn.sparkfun.com/assets/2/b/0/3/8/TFmini_Plus-01-A02-Datasheet_EN.pdf

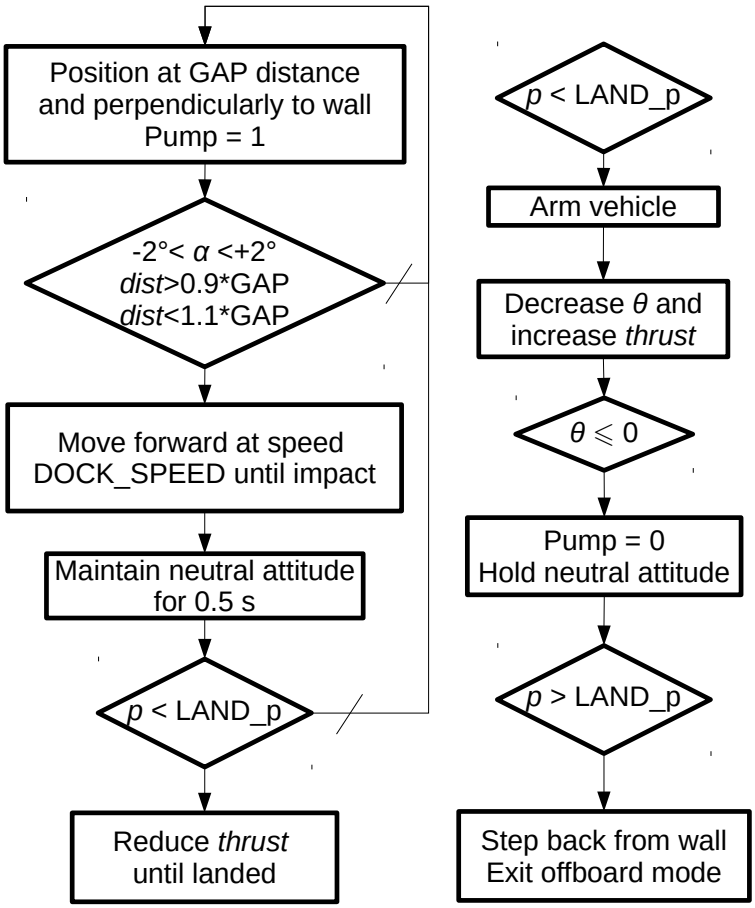

Figure 3.10 Overview of the autonomous perching manoeuvre. Preliminary placement and perching sequences (left), takeoff and detachment sequences (right).

an impact has been detected by the force sensor, the UAV is then commanded to hold its position for 0.5 s, a delay required by the vacuum cup to create a proper seal. If the pressure sensor indicates a drop of pressure below the value of $P_{land} = 880$ hPa (LAND_p in Fig. 3.10), sufficient to hold the vehicle, the landing manoeuvre begins by decreasing thrust until immobilization on the wall. Otherwise, if pressure does not drop after the 0.5 s period, the UAV is instructed to get back to the position it was before the perching sequence started and the perching can be tried again.

The takeoff and detachment sequence follows the procedure shown in the right of Fig. 3.10. After a check to ensure the UAV is perched on a surface, the takeoff can begin. Starting from its current attitude, the UAV drops its pitch angle θ down to 0. Once horizontal and stabilized, the pump is deactivated and the UAV is commanded to hold its attitude. Finally, when the pressure readings confirm that the cup is detached, the UAV is commanded to

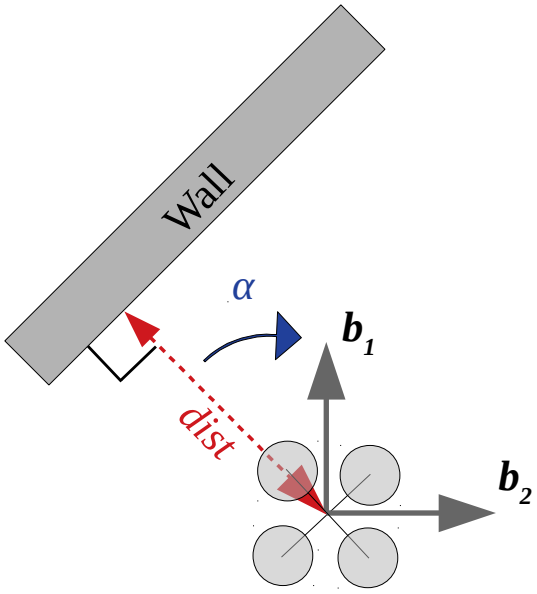

Figure 3.11 Geometry of the pre-perching configuration

move away from the wall and then, it switches to position flight mode and waits for further instructions.

3.6 Experiments

Finally, experiments with perching on vertical surfaces have been conducted both in real world and in simulation, as illustrated in Figs. 3.12 and 3.13. The series of photographs in Fig. 3.12 depicts the perching sequence from the position selected by the pilot to the landing on the wall. Figure 3.13 shows the reverse procedure, starting with the takeoff and ending with the detachment from the wall.

Data streams were recorded using the rosbag ROS package that subscribes to the desired topics and logs the data into a .bag file. The software architecture, the sensors, the published topics are identical in the simulation and in the real world. Therefore, the measurements are performed and processed in the same way. For each flight, the data is timestamped with a global ROS time, alleviating the need to synchronize the various sensors. A python script was used to extract the different topics of a flight from the saved .bag file and to generate plots such as these from Figs. 3.14 to 3.19 depicting comparisons of various parameters recorded from simulations and actual flight data .

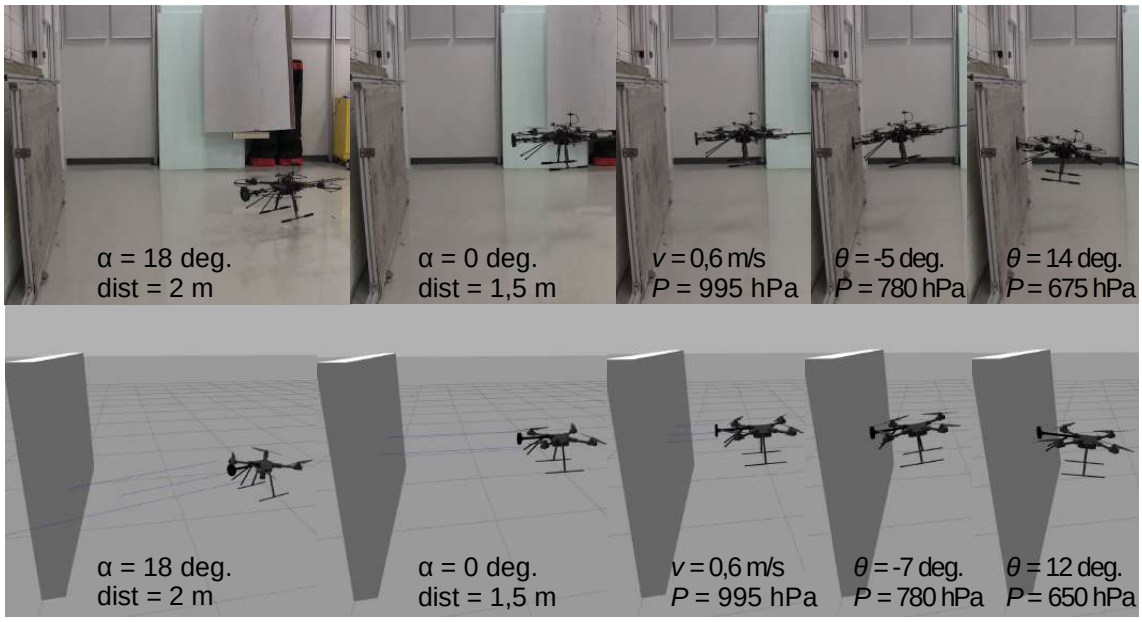

Figure 3.12 Perching manoeuvre sequence on a concrete wall: experiments (top), simulation (bottom)

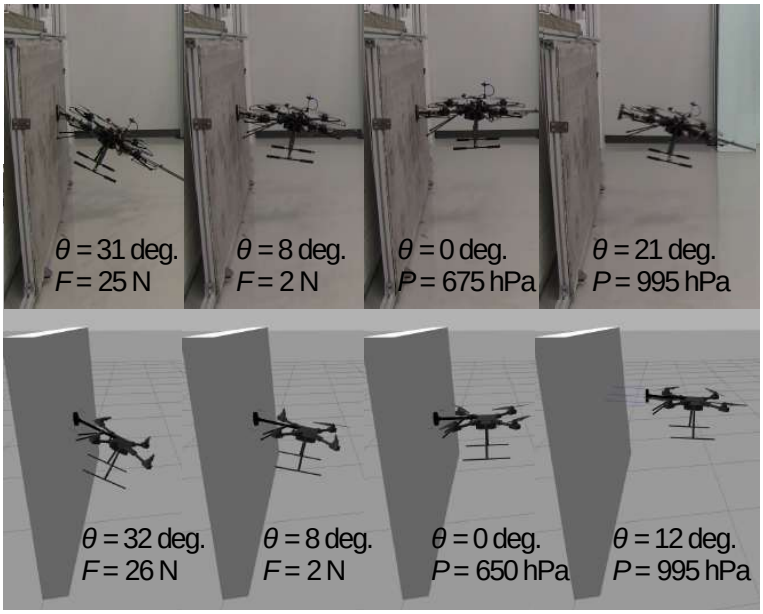

Figure 3.13 Takeoff and detachment manoeuvres from a concrete wall: experiments (top), simulation (bottom)

3.6.1 Perching on a Flat Concrete Wall

A vertical concrete wall is usually highly porous and not airtight. Since air can flow through it, the level of vacuum that the pump is able to achieve in the cup is limited, as shown in

Table 3.2.

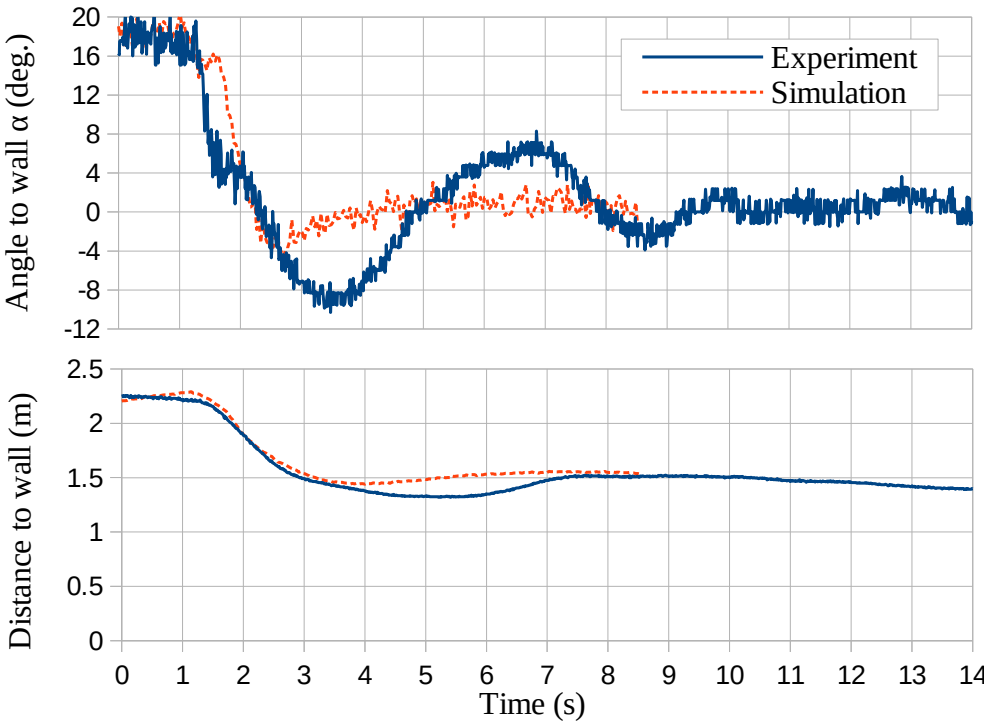

Figure 3.14 Distance and angle relative to wall during preliminary placement

Before launching the manoeuvre to perch on the concrete wall, the UAV is first positioned perpendicular to that wall at 1.5 meters distance as discussed in Section 3.5. Figure 3.14 shows that while plots are similar, control of angle α is noticeably improved in the simulation compared to the experiment. There is a 4 deg overshoot in the former but a 8 deg overshoot in the latter, i.e. twice as large. A similar difference can be noticed with the distance control, still in Fig. 3.14. This difference can mainly be explained by a default from the power supply of the two actual rangefinders, leading to an error on the measured distance and consequently, on the computed angle α too. As a result, the distance and yaw angle corrections sent to the controller are greater than what is actually required. The more efficient control in the simulation allows the perching approach to be initiated earlier, at about 8.5 s, while the actual UAV lags about 6 seconds behind.

Figure 3.15 details data from a complete perching procedure: the UAV first pitches down to reach the desired impact speed (at 0.5 s for the simulation and at 2 s for the experiment).

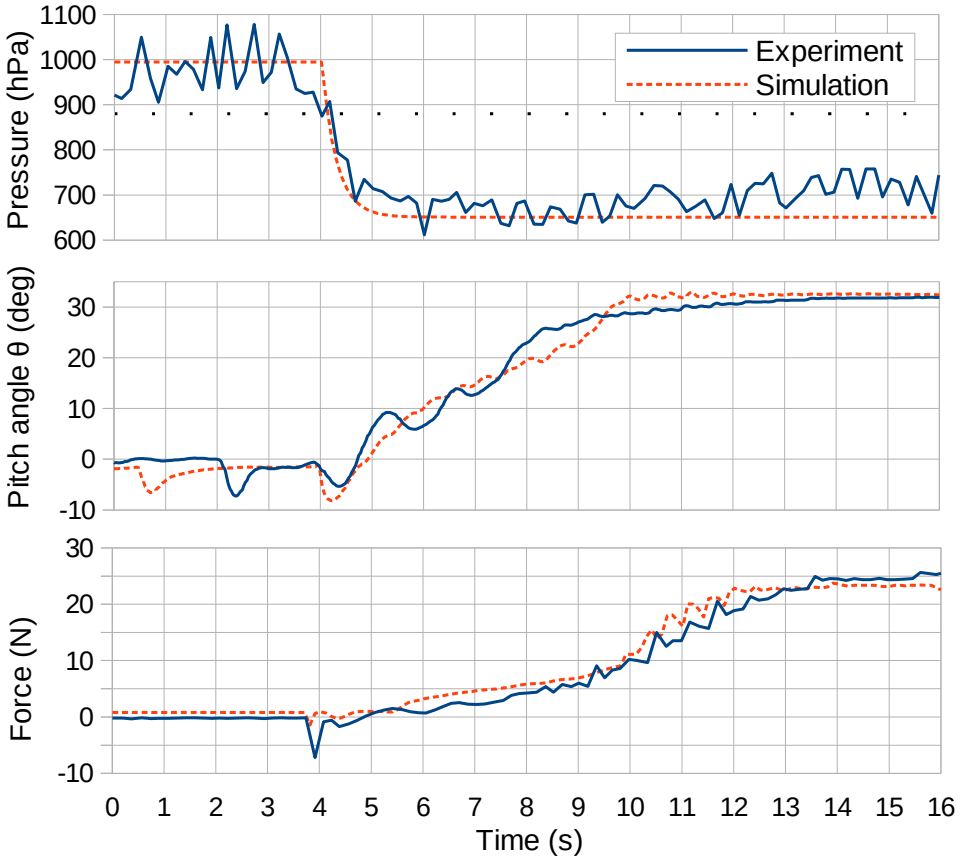

Figure 3.15 Pressure, pitch angle, and force data from an autonomous perching, concrete surface

Then, negative force spikes are observed a few seconds later allowing the controller to know that the vehicle has impacted the wall. The final approach (between the downward pitch and the impact) is shorter in the experiment because the vehicle started the final perching approach closer to the wall. Immediately after impact, the UAV pitches down because of its inertia, as can also be observed in the fourth photograph of Fig. 3.12. Once a successful attachment is achieved, characterized by a drop of pressure below 880 hPa, the UAV increases its pitch in order to land on the wall. Simultaneously, force applied on the vacuum cup increase as lift drops. The thrust subsequently slowly decreases until pitch stabilizes, indicating that the UAV is lying on the wall. The force on the vacuum cup is now at its maximum and the motors can be shut down.

Data from the autonomous takeoff and detachment procedure is similarly shown in Fig. 3.16. At the beginning of the sequence, the pressure being validated below the 880 hPa threshold, the UAV activates its motors. The pitch angle θ is progressively decreased which eases the

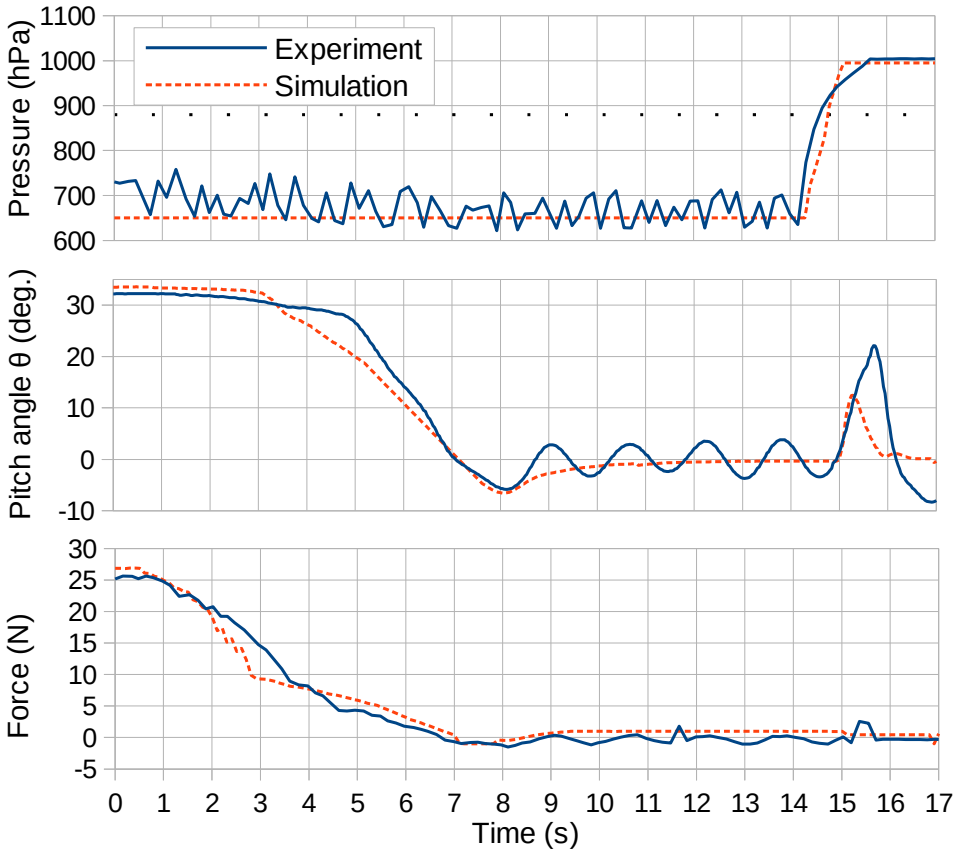

Figure 3.16 Pressure, pitch angle and force data from an autonomous takeoff and detachment, concrete surface

stress on the attachment module. Once Ulysses becomes horizontal, at 10 s, the pitch angle is kept to zero until the pressure is large enough to allow detachment. The airflow leaking through the concrete allows the pressure to build up fairly quickly, in about one second. Finally, the UAV pulls on the vacuum cup with a final upward pitch and stabilizes at a defined distance from the wall. It can be noticed in Fig. 3.16 that while waiting for the pressure to get back to atmospheric value, the pitch of the real vehicle oscillates while in simulation it remains much more stable. This 6 deg. amplitude oscillation at 0.6 Hz is conjectured to be due to unmodelled aerodynamic effects.

3.6.2 Perching on a Flat Metallic Surface

Figures 3.17 and 3.18 show the same manoeuvres as discussed in the previous section but on a metallic surface instead of a porous concrete wall. With the perching manoeuvre, the main difference observed in Fig. 3.17 with previous results is that since the metallic surface is vacuum tight, the pressure in the vacuum cup drops as low as 275 hPa. Additionally,

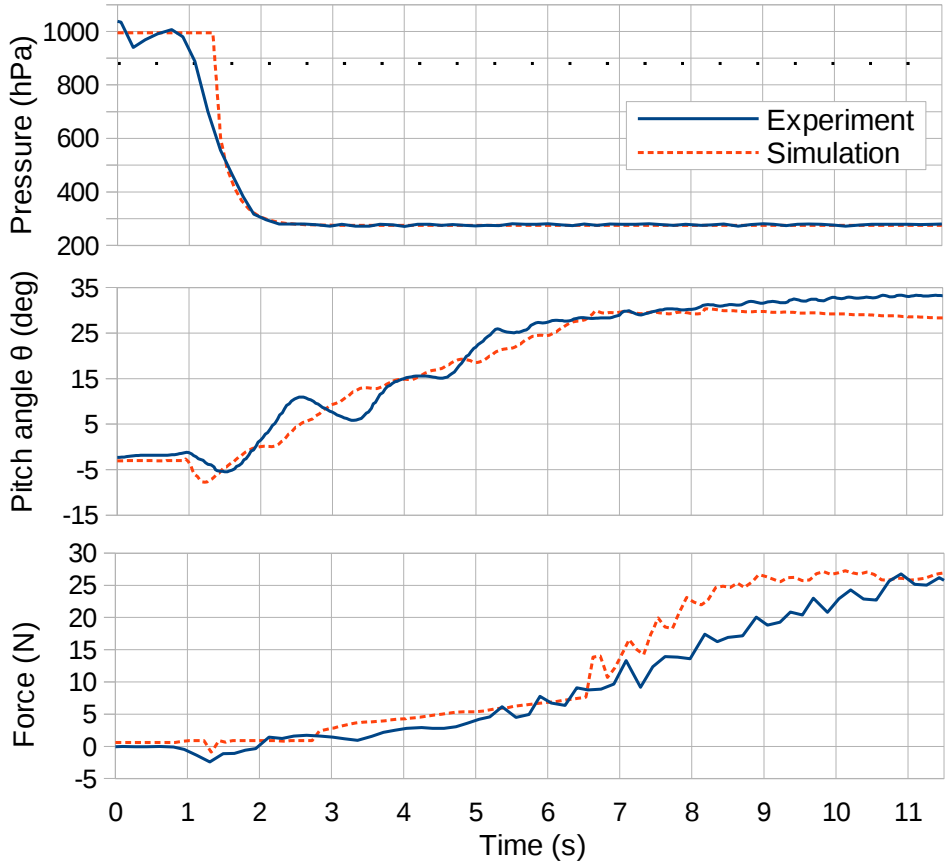

Figure 3.17 Pressure, pitch angle and force data from an autonomous perching, metallic surface

pressure fluctuations from the pump are no longer visible during experiments.

However, as shown in Fig. 3.18, the detachment sequence goes quite differently compared with the concrete wall. Since the vacuum level is eased very slowly, the UAV has to remain stationary for much longer. Consequently, the oscillations of the pitch angle, already observed in Fig. 3.16, last longer and their amplitudes grow larger, exceeding 20 deg. after 16 s. The detachment sequence was interrupted to avoid any incident. Similarly to the previous flight, the simulation does not predict these oscillations on the angle θ and it is most likely for the same reasons, namely the unmodelled aerodynamic effects. Changing the design of the attachment apparatus to allow reversing the pump or tuning the attitude controller of the PX4 could remedy this issue.

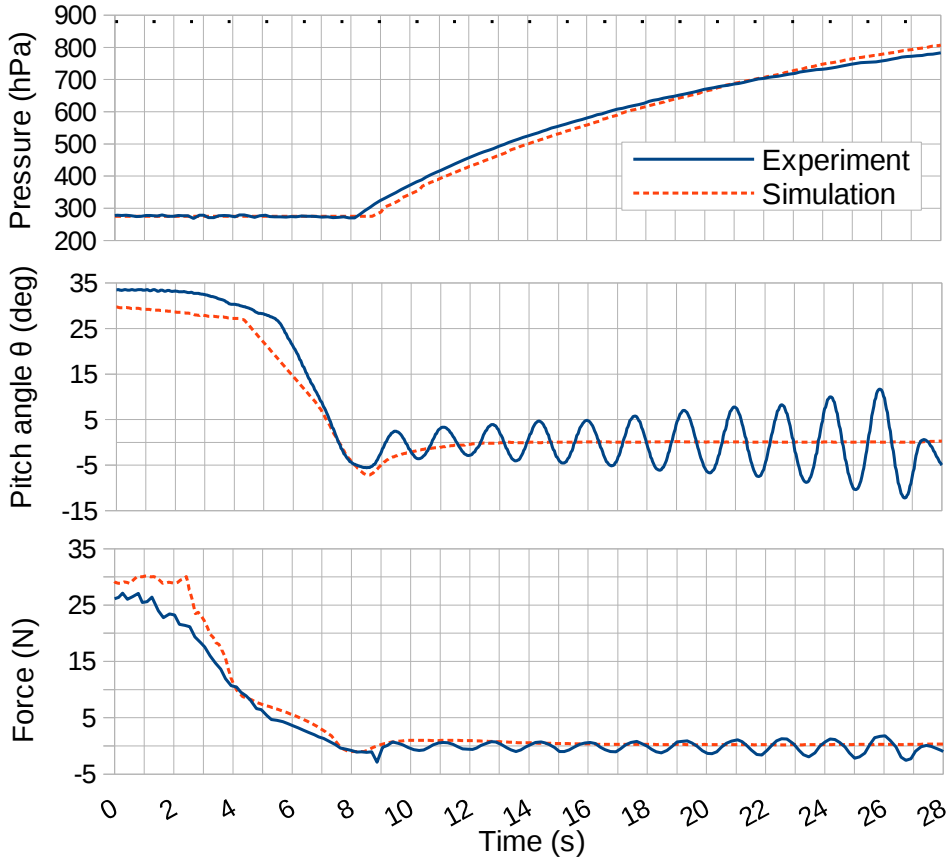

Figure 3.18 Pressure, pitch angles and force data from an autonomous takeoff and detachment, metallic surface

3.6.3 Failed Perching

Figure 3.19 illustrates a failed perching attempt of the UAV on a concrete surface. Similarly to Figs. 3.15 and 3.17, a peak of force immediately followed by a downward pitch indicates that the UAV has impacted the wall. However, contrary to Fig. 3.15 and 3.18, the pressure in the vacuum cup does not drop as expected and therefore, the manoeuvre failed. Consequently, and accordingly with the algorithm discussed in Section 3.5, the UAV tilts its pitch up to move back and returns to its standby position for a new perching attempt.

3.7 Conclusions

In this work, a custom UAV was shown and modeled in a Gazebo simulation. Thanks to ROS simulations, an algorithm for autonomous perching was devised and a corresponding plugin has been designed to allow perching on vertical surfaces. The behaviour of the perching device has been implemented in the simulation based on practical characteristics that

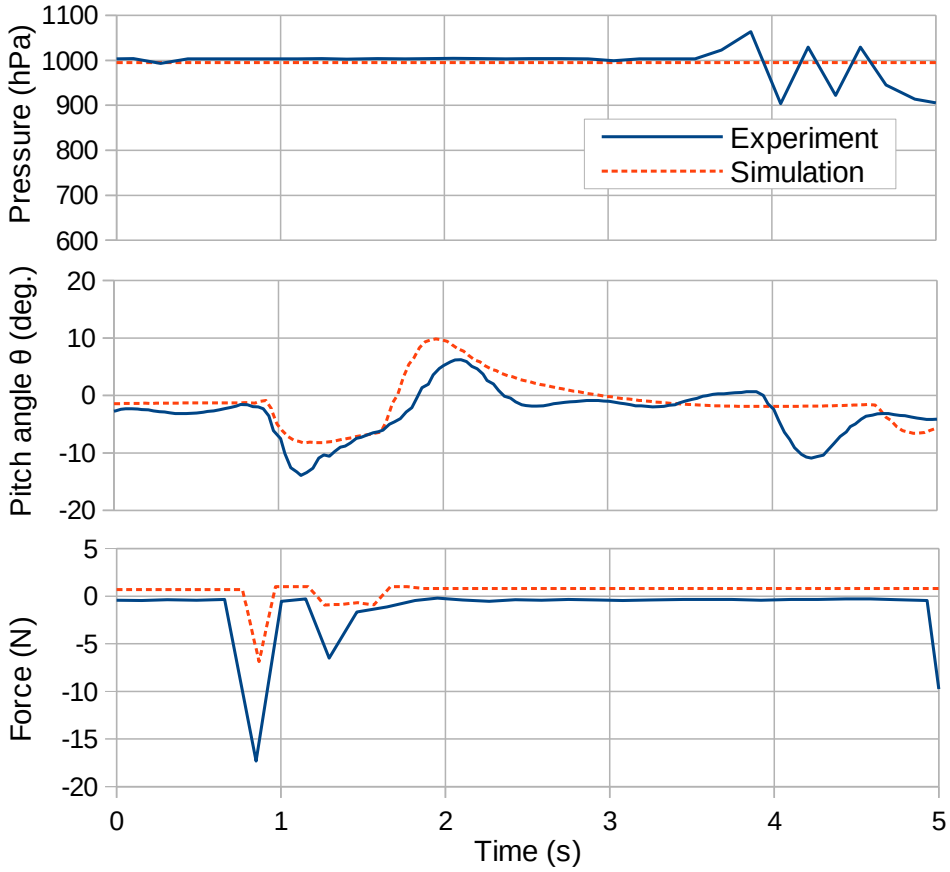

Figure 3.19 Pitch angle and force data from a failed perching attempt

were determined experimentally. The perching manoeuvre has first been automated in the simulation environment using modelled version of the same sensors as these used in the real world. The very same perching strategy has been experimented and the results have been compared to the results obtained in the simulation. The behaviour of the UAV interacting with the surface, perching on the latter, and taking off is well predicted by the resulting simulation. Minor differences can be noticed but the success or failure of each of the perching sequences is accurately reflected. However, the simulation also showed limitation in the prediction of the UAV behaviour during the takeoff manoeuvre. Future work should consider the necessity to more accurately model the aerodynamics interactions with the perching surface. Similarly and following the same methodology as the one presented here, different perching mechanisms could be investigated such as a magnetic or an adhesive perching device.

CHAPITRE 4 RÉSULTATS COMPLÉMENTAIRES

4.1 Conception du dispositif d'amarrage

Il ne sert à rien d'avoir une bonne stratégie d'accostage autonome si le dispositif d'amarrage (DA) ne permet pas au drone de s'attacher de façon robuste à une paroi. Le DA se doit d'être suffisamment fiable pour ne pas risquer de se décrocher lors de l'accostage du drone ou une fois celui-ci posé, dans le cas où un bras robotique effectuerait des tâches sur la structure qui pourraient générer d'importants efforts de réaction. En plus du choix de la mousse de la ventouse mentionné dans l'article, d'autres caractéristiques influencent grandement la capacité du DA à mener à bien l'opération d'accostage, comme son diamètre, les caractéristiques de la pompe employée, les degrés de liberté de la ventouse, etc.

4.1.1 Méthode de ventousage

Figure 4.1 Test extérieur du dispositif d'accostage avec solution à la gomme de guar

Inspiré du Base Attachement Module (BAM), le premier prototype de DA comprenait un

anneau en mousse à cellules ouvertes et nécessitait l'application de vaseline pour permettre l'étanchéité entre la ventouse et la surface. Des tests ont été conduits pour tenter d'y trouver une alternative, car cette méthode était trop salissante et ne partait pas à l'eau. La gomme de guar, mélangée à de l'eau à hauteur de 1% massique, s'est avérée être une bonne alternative puisque soluble dans l'eau et donc facilement nettoyable. Malheureusement, lors de tests en extérieur (voir Fig. 4.1) cette solution a gelé, empêchant le fonctionnement normal du DA. Retourner à l'emploi de vaseline n'a pas résolu le problème car celle-ci subissait aussi un changement de consistance à cause du froid amenant au même résultat qu'avec la solution aqueuse. Heureusement, en utilisant une mousse à cellule fermées, de très bon résultats ont été obtenus sans même nécessiter l'usage d'une solution. Pour une même mousse en éthylène-propylène-diène monomère (EDPM [1]), l'étanchéité était aussi bonne voire meilleure sans la solution qu'avec, comme en témoigne le Tableau 4.1. Cependant, la force que le DA pouvait alors supporter permettait tout juste de porter le drone une fois celui-ci accosté au mur. Il devint alors nécessaire de trouver des solutions pour augmenter la force de succion de la ventouse.

Tableau 4.1 Temps de maintien du vide dans la ventouse une fois la pompe éteinte, pour la mousse *Light duty blended EPDM*, sur mur de béton poreux

Temps sans solution (s)	Temps avec solution à 1% de gomme de guar (s)
9.0	5.0
12.0	5.0
11.5	5.7

4.1.2 Dispositif de succion

Pour utiliser la mousse mentionnée précedemment à sec (sans solution aqueuse ni vaseline), le modèle de pompe initialement utilisé, la SFE 12 V Air Pump [2], a dû être remplacé par une autre pompe, la SP 625 EC-LC-DUp-VD de Schwarzer [3]. Parmi les caractéristiques réunies dans le Tableau 4.2, le principal avantage de cette dernière est la pression minimale qu'elle peut atteindre, et donc la plus grande force de succion qu'elle confère à la ventouse. En effet, la force de traction normale à laquelle peut théoriquement résister la ventouse s'écrit :

$$F = (P_{atm} - P_v) \cdot S_a \tag{4.1}$$

1. Light duty blended EPDM foam : `https://www.mcmaster.com/8647K46`
2. SFE 12 V Air Pump : `https://www.robotshop.com/ca/en/sfe-air-pump.html`
3. SP 625 EC-LC-DUp-VD 24 VDC : `https://www.schwarzer.com/en/gas-pumps/eccentric-diaphragm-pumps/sp-625-ec-lc-dup-vd`

avec P_{atm} la pression atmosphérique, P_v la pression dans la ventouse, et S_a la surface active de la ventouse, c'est à dire la surface non recouverte de mousse.

Tableau 4.2 Comparaison des caractéristiques de la première pompe à vide (à gauche) et de la seconde (à droite)

Modèle de pompe	SFE 12V Air Pump	SP 625 EC-LC-DUp-VD
Débit libre	12-15 L/min	11.5 L/min
Pression minimale	470 hPa	285 hPa
Masse	0.280 kg	0.250 kg

D'après l'équation 4.1, la simple amélioration du niveau de vide de la pompe répertoriée dans le Tableau 4.2 permet en théorie une augmentation de 35% de la force de succion de la ventouse.

4.1.3 Disque rigide et performances de la ventouse

En plus du changement de modèle de pompe, le design de la ventouse a été modifié pour améliorer sa résistance au décrochage. Conçu sous SolidWorks, le disque sur lequel vient se placer la partie en mousse est ensuite imprimé en 3D avec du PLA (acide polylactide). Pour assurer l'étanchéité de la ventouse, la pièce est ensuite enduite de résine [4]. Le premier prototype, à gauche sur la Figure 4.2, mesure 9 cm de diamètre et présente une face plane sur laquelle est collé l'anneau de mousse. En s'inspirant du Grabo [5], un système permettant de soulever des charges allant jusqu'à une centaine de kilogrammes et reposant sur le principe de la ventouse, un nouveau prototype de ventouse a été développé. Visible à droite sur la Figure 4.2, la pièce présente un diamètre plus important ainsi qu'une rainure pour accueillir la mousse et la maintenir par friction, sans colle. Dans le cadre du prototypage, la rainure permet un changement aisé et rapide du modèle de mousse. Par ailleurs, d'une profondeur de 5 mm et d'une largeur de 10, la rainure permet d'ajouter de la rigidité à la mousse une fois celle-ci compressée, lorsque la ventouse est accrochée à une surface. En effet, lorsque le drone est perché à un mur, l'effort sur la ventouse n'étant pas nécessairement normal, la mousse ne se déforme pas uniformément, ce qui a pour effet de fragiliser l'attache au mur.

Les progrès obtenus sur les performances de la ventouse sont visibles sur la Figure 4.3 et sur le Tableau 4.3. Le prototype 1 est constitué de la petite ventouse et de la pompe SFE et le prototype 2, de la grande ventouse et de la pompe SP 625.

4. XTC-3D protective coating
5. Electric vacuum lifter : https ://grabo.com/

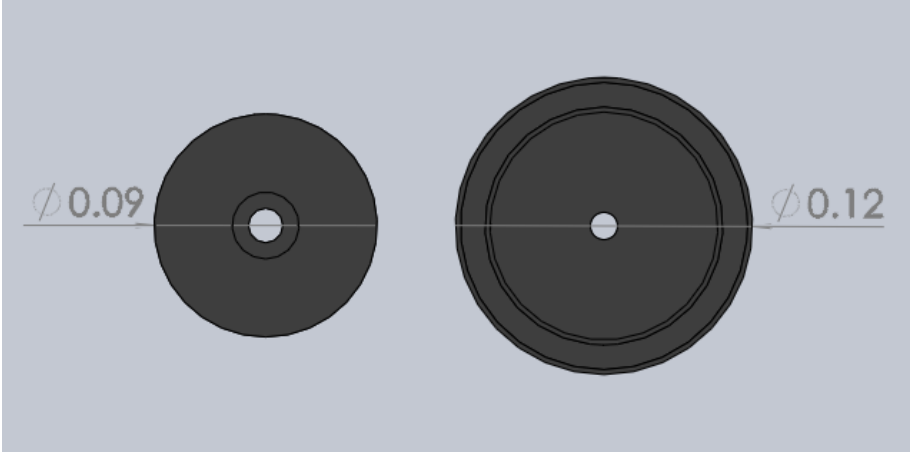

Figure 4.2 Prototypes de disque rigide. Premier prototype de disque plan (à gauche), et second prototype, avec rainure (à droite).

Tableau 4.3 Valeurs théoriques et mesurées de la force normale maximale que la ventouse peut supporter avant décrochage, sur paroi de béton poreux

	Prototype 1	Prototype 2
Force d'arrachement théorique	77 ± 7 N	275 ± 20 N
Force d'arrachement mesurée	69 ± 1 N	> 170 N

Dans le Tableau 4.3, les forces d'arrachement théoriques sont calculées d'après l'équation 4.1, en multipliant la pression relative mesurée sous la ventouse par la surface active de la ventouse. Le petit disque présente une surface active d'un diamètre de 70 mm, contre 100 mm pour le grand disque. La pression dans la ventouse P_v est mesurée lorsque la ventouse est scellée contre la paroi de béton poreux, pompe allumée. En moyenne, sur un même zone du mur, on relève une pression P_v de 800 hPa pour le prototype 1 et 650 hPa pour le prototype 2. Ces pressions ne sont pas aussi basses que les valeurs théoriques minimales données dans le Tableau 4.2, car le dispositif présente des fuites et pertes de charge et que l'air peut passer à travers la surface de béton poreuse. La force d'arrachement mesurée est calculée à partir de 26 essais de décrochage pour le prototype 1. Pour le prototype 2, la force de décrochage est tellement élevée qu'elle n'a pu être atteinte, le risque étant de détériorer le dispositif d'amarrage qui n'a pas été conçu pour porter des charges aussi lourdes. De plus la limite maximale du capteur de force est de 20 kg (196 N). On note que pour le premier prototype, la force maximale est plus faible qu'en théorie car le calcul ne prend pas en compte la force élastique exercée par la mousse compressée sur le disque.

Comme en témoigne la Figure 4.3, la pression dans la ventouse du second prototype (courbe

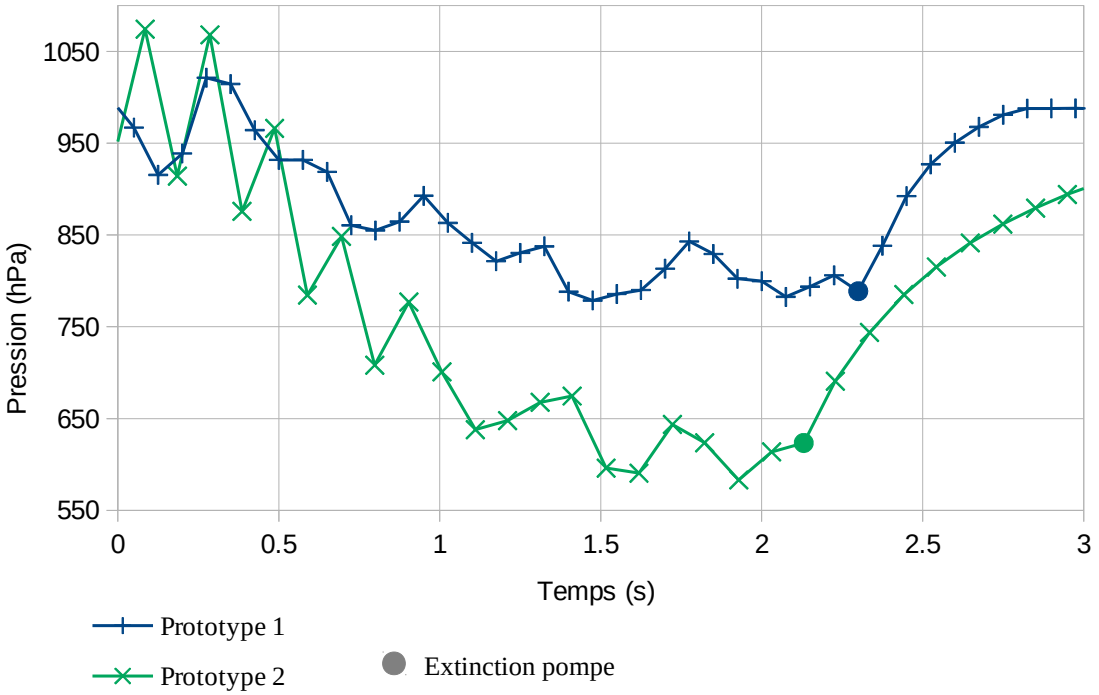

Figure 4.3 Comparaison de l'établissement du vide dans les ventouses du prototype 1 et du prototype 2, sur paroi de béton poreux

verte avec symbole de croix) atteint une valeur finale inférieure à celle du premier prototype (courbe bleue avec symbole plus), conformément aux spécifications techniques des pompes à vide. L'interprétation de l'écart en terme de vitesse d'établissement du vide n'est pas évidente. Du fait du diamètre plus important de la grande ventouse, le volume d'air à pomper devrait être plus grand mais la rainure permet de diminuer sa hauteur et tempère l'effet. Aussi, une ventouse plus large offre une plus grande surface de fuite d'air par la paroi de béton poreux et donc, il y est plus difficile et plus long de faire descendre la pression. En pratique, on observe que le vide s'établit plus rapidement dans la ventouse du prototype 2, bien que la seconde pompe est présentée sur le Tableau 4.2 comme ayant un débit plus faible.

Il est intéressant de noter par ailleurs que le simple fait d'augmenter le diamètre actif de la ventouse de 7 cm à 10 cm (S_a passe alors de 38 cm^2 à 79 cm^2), aurait permis en théorie de doubler la force supportable par le DA avant arrachement. Mais au vu des remarques précédentes à propos des fuites d'air, les améliorations conférées par l'augmentation du diamètre ont une limite, d'où la nécessité de changer aussi de modèle de pompe.

4.1.4 Degrés de liberté de la ventouse

Deux configurations ont été testées pour la liaison entre la ventouse et les tiges solidaires du drone qui portent la ventouse et sa pompe. La première configuration autorisait deux mouvements de rotation de la ventouse par rapport au drone : un autour de l'axe de tangage et l'autre autour de l'axe de lacet (axes de rotations définis dans la section 3.3.2). Hors, lors de tests d'accostage en pilotage manuel, une fois la ventouse accrochée au mur il était difficile de maintenir l'angle de lacet du drone, ce qui compliquait encore plus cette manoeuvre déjà délicate. Il fut alors décidé de ne garder que le premier degré de liberté, représenté en tirets et points dans la section 3.4, ce qui facilita grandement la manoeuvre. Cependant le second degré de liberté, la rotation autour de l'axe vertical, permettait une plus grande adaptabilité du DA lors d'accostages où la perpendicularité entre la ventouse et la surface n'était pas totalement atteinte. Mais en pilotage automatique, il est raisonnable de considérer que le maintien de la perpendicularité lors de l'accostage ne présente pas de problème pour l'algorithme. Finalement, les deux designs peuvent être utilisés indifféremment lorsque le drone est autonome.

4.1.5 Intégration des capteurs

Pour que le drone puisse connaître son état lors de la manoeuvre d'accostage, il a semblé nécessaire d'intégrer au DA un capteur de force, pour détecter les collisions avec la surface, et un capteur de pression, pour déterminer si la ventouse est scellée sur la surface.

Après modification de certaines pièces du DA, un capteur à jauge de déformation a pu être intégré entre la ventouse et les ressorts. Puisque installé en porte à faux, il fallut s'assurer que les pièces imprimées auxquelles le capteur de force est fixé soient suffisamment robustes. Enfin, une fois installé, il est nécessaire de calibrer le capteur de force.

Le capteur de pression est quant à lui installé sur le réseau pneumatique reliant la pompe à la ventouse. Mais au vu de sa disposition, proche de la pompe, le capteur mesure aussi les à-coups de celle-ci, visibles sur la Figure 4.3. L'amplitude des à-coups, d'une centaine d'hPa, empêche de bien observer l'évolution de la pression au cours du temps. Cela gène certes l'analyse des performances du DA mais surtout, cela peut poser problème pour le drone qui détermine l'état de sa ventouse scellée/non scellée, à partir de cette mesure. Pour remédier à cela, il faut placer le capteur dans la ventouse, au plus loin du branchement au réseau pneumatique. Cela nécessite de modifier le disque rigide de la ventouse et d'y percer une autre entrée pour le capteur de pression.

4.1.6 Diminution du couple subi par la ventouse

Au vu de l'inclinaison du drone une fois posé contre le mur, la direction de la force du drone sur la ventouse n'est pas perpendiculaire à celle-ci, car les tiges support ne sont pas perpendiculaires à la ventouse (première image de la Figure 3.13). Cela a pour effet de plus compresser la partie basse que la partie haute de la ventouse. Ce bras de levier qui agit sur la ventouse peut être diminué en modifiant le design du DA. Pour cela, il faut que la droite support de la force soit parallèle à la droite passant par le centre de la ventouse et le point d'application de la force. À cet effet, il faut décaler la liaison entre les tiges du DA et la ventouse vers le bas de la ventouse, comme indiqué sur la Figure 4.4.

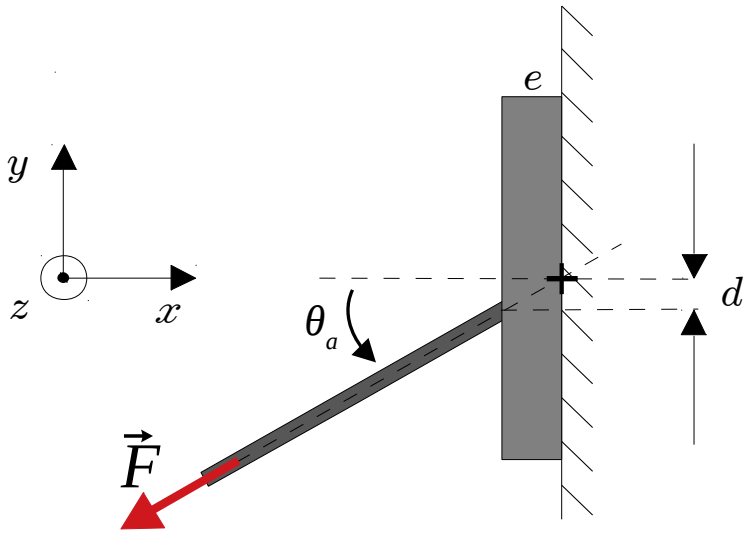

Figure 4.4 Schéma du décalage de la liaison entre la ventouse et le drone afin d'annuler le couple généré par la force de traction

La distance verticale d dont il faut décaler la liaison est simplement :

$$d = -\tan(\theta_a) * e$$

où θ_a est l'assiette du drone une fois accosté et e, la distance entre la liaison de la ventouse et le mur.

4.2 Banc d'essai pendulaire

Pour pouvoir modéliser le DA dans la simulation d'accostage, il est nécessaire de caractériser le dispositif réel. Les principales caractéristiques retenues et présentées aux section 3.4.3 et

4.1.3, et à la figure 3.5 sont :

- Le taux de succès du ventousage en fonction de la vitesse d'accostage (considérant la ventouse parallèle à la surface)
- L'évolution de la pression dans la ventouse lors de son attachement et de son détachement
- La force de traction maximale que peut supporter la ventouse avant détachement

Pour quantifier le premier paramètre le plus fidèlement possible, un grand nombre d'accostages doivent être effectués. La manoeuvre d'accostage prenant un certain temps à être exécutée, même une fois automatisée, un banc d'essai a été développé. Ce banc d'essai pendulaire, présenté succinctement à la section 3.3.3 et visible sur la figure 4.5, permet de collecter les données d'accostages pertinentes sur plusieurs dizaines de manoeuvres en quelques minutes seulement. Le pendule a été fait le plus long possible pour avoir une trajectoire la plus rectiligne possible et ainsi se rapprocher au mieux des conditions dans lesquelles ont lieu un accostage réel. L'emploi de deux câbles placés à l'avant et à l'arrière du drone permet le maintien de sa perpendicularité avec la surface d'accostage. La vitesse d'accostage pendulaire peut être modifiée en lâchant le drone plus ou moins loin de la surface. Lors des accostages pendulaires les moteurs et le Pixhawk restent éteints, seul l'ordinateur compagnon est alimenté pour effectuer les mesures. Les données acquises lors de ces essais incluent la distance entre le drone et le mur par les LiDARs, la pression dans la ventouse et la force mesurée par la jauge de contrainte du DA. Un algorithme a été écrit pour traiter ces données et ainsi détecter l'impact, connaître la vitesse du drone lors de cet impact, surveiller l'évolution de la pression dans la ventouse après l'impact, et ainsi en déduire si la ventousage a réussi (voir Fig. 3.5). Aussi, en mesurant l'évolution de la pression aux accostages et désaccostages, le comportement de la ventouse lors du scellage peut être caractérisé (voir Tableau 3.2).

Ce système d'imitation d'accostage basé sur un pendule peut permettre de caractériser le comportement de différents prototypes de DA, sur différentes surfaces et avec différents véhicules. Le gain de temps par rapport à un accostage où le drone doit être piloté, manuellement ou automatiquement, est considérable. Une fois la caractérisation faite, les données du DA réel peuvent être utilisées dans la simulation pour dicter le comportement du DA simulé.

4.3 Accostage autonome dans le monde réel

4.3.1 Réduction des risques de crash

Bien que l'environnement de simulation permette de repérer les coquilles et erreurs d'implémentation dans le script avant l'exécution de l'accostage autonome dans le monde réel, tout

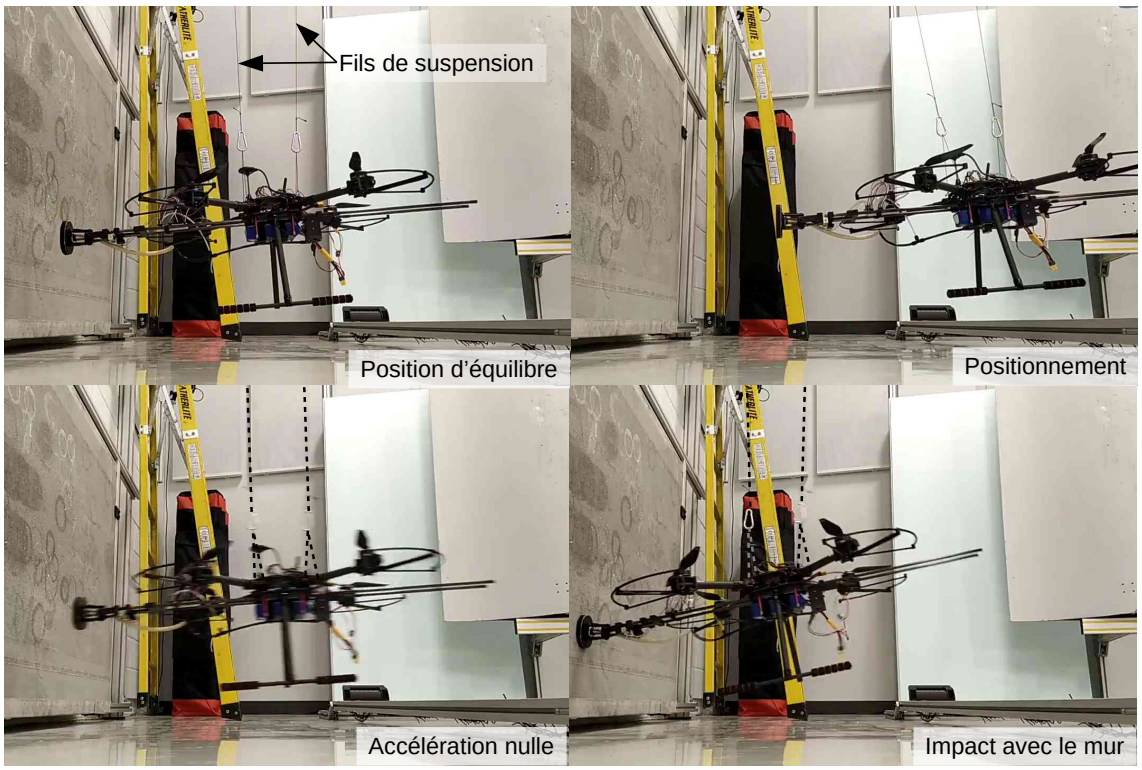

Figure 4.5 Principales étapes d'un accostage avec le banc d'essai pendulaire

incident sur le drone doit être envisagé. Un crash du véhicule, en cas de casse, ferait perdre beaucoup de temps à le réparer, ce qui est particulièrement délétère en phase de développement. C'est à cet effet qu'un environnement de simulation a été développé, afin d'explorer des stratégies d'accostage sans prendre de risques. Mais pour valider sur le terrain la solution d'accostage autonome retenue après développement, il faut être capable de pallier à toute éventualité. Dans l'algorithme d'accostage, une géo-barrière (*geofence* en anglais) a été mise en place : l'espace de vol a été restreint à un cube de quelques mètres autour du point de décollage. Cela permet d'éviter toute collision intempestive avec l'environnement de la salle de vol, hormis avec la paroi d'accostage. De plus, lorsque le drone est en vol autonome, l'opérateur garde les mains sur la radio-commande. Il a alors la possibilité de reprendre la main immédiatement : en appuyant sur un bouton, le drone interrompt l'algorithme d'accostage, quitte le mode OFFBOARD et passe en mode POSITION. C'est alors la radio-commande qui permet le pilotage du drone, même s'il l'algorithme d'accostage tourne encore. Enfin, en cas d'urgence, un autre bouton permet de stopper les moteurs en tout temps.

4.3.2 Ordinateur Compagnon

Comme mentionné dans le chapitre 3, l'ordinateur embarqué est un nano-ordinateur Raspberry Pi (RPi) modèle B avec 4 GB de RAM, sur lequel Ubuntu 20.04 est installé. Le système d'exploitation Ubuntu est installé à la place de Raspbian pour avoir la configuration la plus proche de celle de l'environnement de simulation, lui aussi sous Ubuntu (Figure 3.7). En effet, la version de ROS à installer dépend elle-même de la version d'Ubuntu utilisée. De plus, la compilation des modules ROS est plus longue et plus ardue avec Raspbian. Ainsi, l'ajout et la compilation des modules ROS se fait de la même façon sur l'ordinateur compagnon que sur l'ordinateur de la simulation.

Le RPi doit être alimenté en 5V DC avec une puissance suffisante pour assurer le bon fonctionnement de toutes les opérations nécessaires à l'accostage autonome : exécution de l'algorithme, envoi et réception de données *via* Wifi, communication avec le Pixhawk, alimentation des capteurs, etc. Pour cela, le Raspberry peut être alimenté par USB type C, ou par un de ses ports General-Purpose Input/Output (GPIO) 5V. Pour référence, l'alimentation officielle est un adaptateur secteur 5.1V USB-C pouvant délivrer jusqu'à 3A. Cependant, en alimentant le RPi avec la batterie du drone par un GPIO, comme c'est le cas lorsqu'il est embarqué sur le véhicule, ses ressources sont limitées et ce, quel que soit le courant fourni par le convertisseur 5V employé. En effet, avec une puissance de calcul amoindrie, un message d'erreur de la forme `[ WARN] [1549505093.414920941]: TM : RTT too high for timesync: 3197.44 ms` peut être retourné par MAVROS. Cela signifie que le temps qu'a mis le paquet de synchronisation temporelle pour faire l'aller-retour entre PX4 et le système distant a été trop long. Ce délai d'aller-retour entre PX4, sur le Pixhawk, et MAVROS, sur le RPi, correspond au Round Trip Time (RTT). La Figure 4.6 [6] indique que le message d'erreur est retourné après 5 occurrences d'un RTT supérieur à 100 ms. Si le RTT est trop long, le drone active son mode FAILSAFE et quitte donc le mode OFFBOARD. Ce mode de vol exécute une action pré-configurée, par exemple un atterrissage immédiat.

Le RTT augmente avec la sollicitation du RPi et représente un danger lorsque le véhicule est proche d'obstacles. Pour y remédier, il faut donc limiter toute consommation inutile de ressources. Pour cela, l'interface graphique de Ubuntu doit être désactivée sur le RPi, et les scripts d'accostage autonome doivent être exécutés via SSH, au lieu de procéder à une connexion graphique avec VNC Viewer. Un autre solution consiste à alimenter le RPi via un convertisseur branché sur la batterie du drone mais avec une sortie USB-C au lieu d'un branchement direct aux ports GPIO. Des modèles de tels convertisseurs ont été repérés [7] mais

6. Tiré de https://px4.github.io
7. USB Buck Converter DC 6-32V to 5V

```
MAX_RTT_SAMPLE = 100 ms
MAX_CONSECUTIVE_HIGH_RTT = 5


if (rtt_us < MAX_RTT_SAMPLE) {  // Only use samples with low RTT
    [...]
    } else {
    // Increment counter if round trip time is too high for accurate
     ↪  timesync
    _high_rtt_count++;

        if (_high_rtt_count > MAX_CONSECUTIVE_HIGH_RTT) {
            PX4_WARN("[timesync] RTT too high for timesync: %llu ms",
             ↪  rtt_us / 1000ULL);
            // Reset counter to rate-limit warnings
            _high_rtt_count = 0;
    }
```

Figure 4.6 Extrait de la fonction `mavlink_timesync.cpp`

le temps de livraison n'a pas permis leur intégration au projet.

Après avoir réduit la charge de calculs sur le RPi, et toujours avec une alimentation par GPIO, le message `RTT too high` continue d'apparaître de temps à autre, mais seulement au lancement de scripts ou à l'arrêt de l'enregistrement des données. Les expériences ont pu être menées sans encombre, mais ce problème doit être résolu en cas d'utilisation d'un algorithme d'accostage plus gourmand en calculs.

4.3.3 Capture de mouvement avec OptiTrack et positionnement par GPS

Les expériences d'accostage se faisant en intérieur, le positionnement par satellite ne peut pas être utilisé. C'est pourquoi le système de capture de mouvement OptiTrack est utilisé à la place. Ce dispositif utilise un réseau de plusieurs caméras réparties dans la salle de vol, pour localiser dans l'espace un corps rigide défini par plusieurs marqueurs. En plaçant judicieusement ces marqueurs sur le véhicule, il est possible d'obtenir une excellente précision sur

la position du corps rigide, de l'ordre du millimètre. Cependant, ces conditions ne sont pas représentatives des missions de maintenance en extérieur auxquelles est destiné le drone. En général, le positionnement de drones en extérieur se fait à l'aide d'un système de positionnement par satellite (global navigation satellite system, GNSS). D'après [68], la précision d'un GNSS seul est de l'ordre de plusieurs mètres. Heureusement, combiné aux données provenant des centrales inertielles (inertial measurement unit, IMU) avec un filtre de Kalman étendu (extended falman filter, EKF) la précision peut descendre sous le mètre. Cependant, dans le cas d'une mission de maintenance où il est question d'opérer sur un point bien précis, il faut s'assurer que celui-ci soit atteignable par le bras robotique embarqué un fois le drone accosté sur la structure. Aussi, dans le cas d'une zone d'accostage réduite, le drone doit être capable de s'y percher précisément. Et à priori, le positionnement par GNSS combiné à des capteurs de distance tels que ceux utilisés ici, ne garantie pas la maîtrise totale de l'emplacement d'accostage. Vérifions l'erreur de positionnement du drone de démonstration lorsqu'il utilise le GNSS pour se positionner.

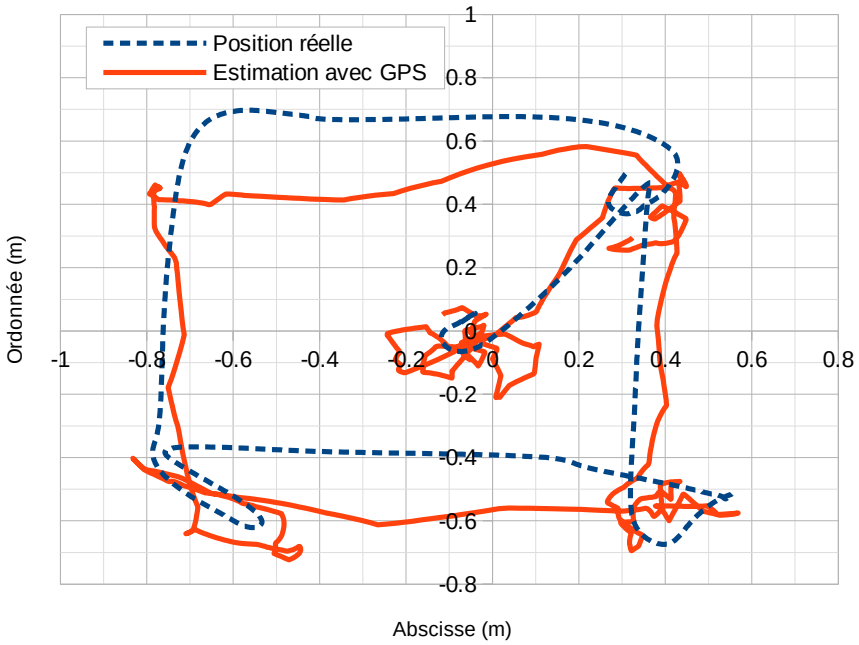

Figure 4.7 Position estimée avec l'EKF et position réelle du véhicule lorsqu'il effectue une trajectoire en carré en utilisant le GPS

Les figures 4.7 et 4.8 présentent l'estimation de la position du drone comparée à sa position réelle lorsque le drone doit effectuer une trajectoire simple. L'estimation de la pose est effectuée par PX4 en fusionnant les données de differents capteurs et en utilisant un filtre de Kalman étendu (EKF). La figure 4.7 montre que lorsque le quadrirotor calcule son position-

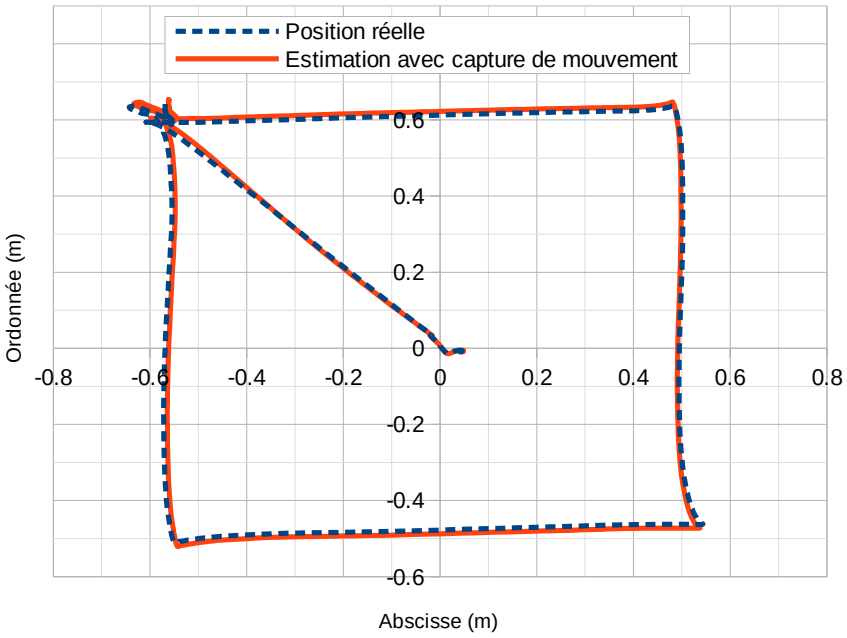

Figure 4.8 Position estimée avec l'EKF et position réelle du véhicule lorsqu'il effectue une trajectoire en carré en utilisant le système OptiTrack

nement avec des données GPS, l'erreur par rapport à sa position réelle peut être de plusieurs dizaines de cm. En revanche, en calculant sa position à partir des données de la capture de mouvement effectuée avec le système OptiTrack, cette erreur ne dépasse pas le centimètre sur la Figure 4.8.

À l'erreur de position par GNSS dans des conditions nominales, s'ajoutent les difficultés que représente la proximité du véhicule avec des structures qui pourraient perturber la bonne réception des satellites de positionnement. Par conséquent, la précision de positionnement conférée par un GNSS semble insuffisante pour un accostage précis, contrairement au positionnement avec le système Optitrack. Bien sûr, il n'est pas possible d'installer de tel dispositif sur le lieu d'intervention du drone mais, en intégrant une caméra de profondeur telle que celle présentée à la section 4.3.4, il semble possible d'obtenir la précision que le positionnement par GPS ne permet pas d'atteindre. Un autre solution consiste à utiliser la technique de cinématique temps réel (real-time kinematics, RTK) qui permet un positionnement par satellite bien plus précis et qui nécessite l'installation d'une station au sol.

4.3.4 Accostage par reconnaissance visuelle embarquée

Dans les expériences menées, l'estimation de la distance et de l'angle entre le drone et la surface à accoster se fait à l'aide de deux LiDARs mesurant chacun une distance. Bien que cette méthode s'avère fiable et précise dans le cadre des expériences, elle peut présenter des limites pour des sites d'accostages plus délicats. Une solution plus adaptée et permettant de se repérer face à des structures complexes consiste à utiliser par exemple une caméra de profondeur. Des exemples d'utilisation de ce type de capteur pour le positionnement de drones peuvent être trouvés dans [51, 67, 69]. En utilisant cette technique, une précision de l'ordre du centimètre pour le positionnement du drone est démontrée dans [67].

Les figures 4.9 et 4.10 illustrent l'utilisation de la caméra Intel Realsense D455 [8] respectivement dans la simulation Gazebo et dans le monde réel. Cette caméra était initialement prévue pour permettre le repérage du drone par rapport à la surface à accoster.

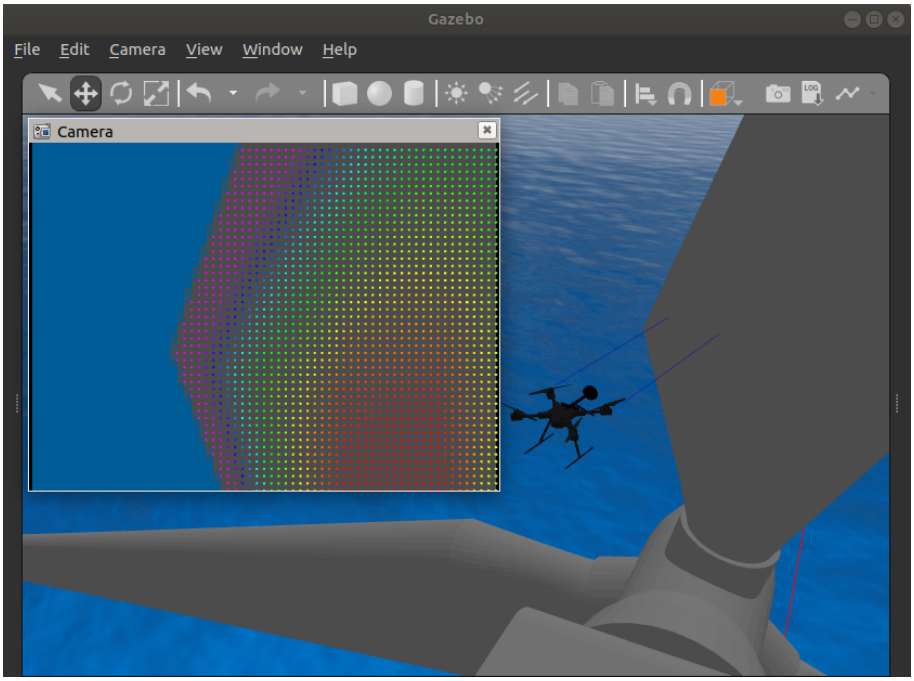

Figure 4.9 Visualisation du nuage de point dans la simulation

Sur la Figure 4.9, la fenêtre 'caméra' (à gauche) affiche le nuage de points capturé par la caméra de profondeur du véhicule placé face à une pale d'éolienne, dans l'environnement de simulation. Il est alors ensuite possible, à partir du sujet ROS publié par la caméra à 30 Hz, de calculer les coordonnées du plan face au drone. Les calculs s'effectuent sur l'ordinateur qui héberge la simulation, avec un paquet ROS extrait d'un exemple de [70]. Il

8. `https://www.intelrealsense.com/depth-camera-d455/`

est constaté que la fonction de placement préliminaire permet de positionner avec succès le véhicule perpendiculairement au mur. L'utilisation de la caméra de profondeur étant validée en simulation, il faut vérifier son bon fonctionnement pour la configuration réelle cette fois.

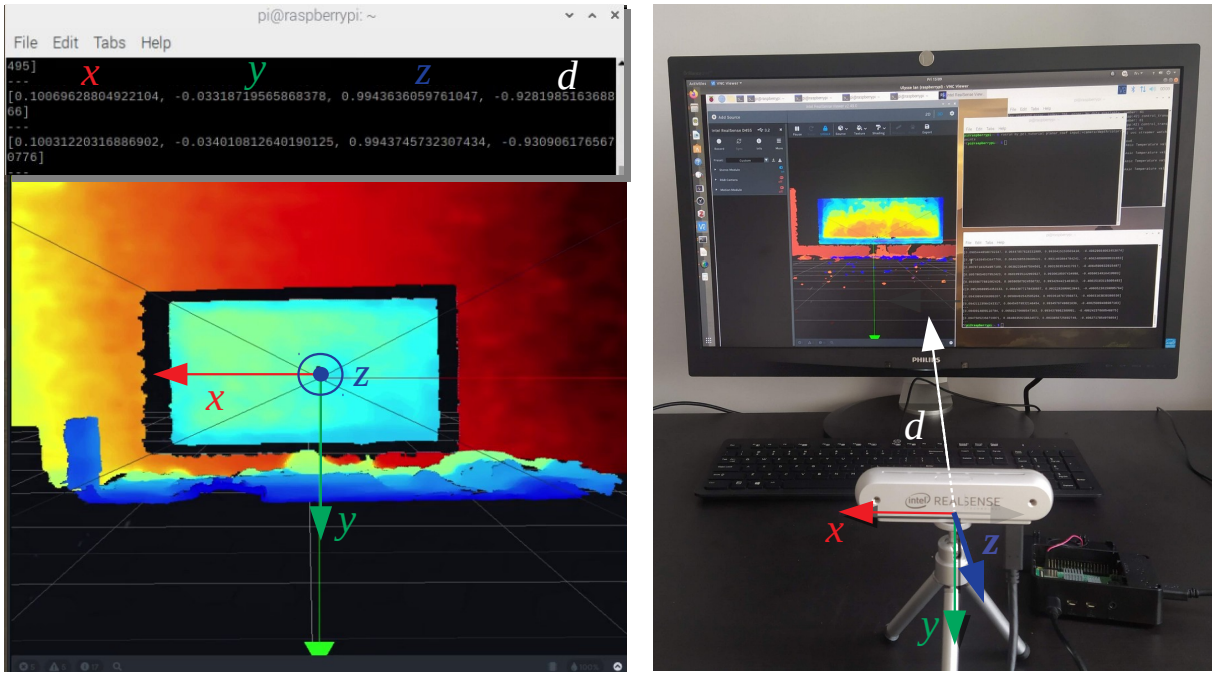

Figure 4.10 Visualisation du nuage de points extrait du monde réel et estimation des coordonnées d'un plan

La figure 4.10 montre le traitement des données de la caméra de profondeur Intel Realsense D455 (en gris à droite) et l'obtention des coordonnées du plan dans son champ de vision. Les coordonnées du plan formé par le rectangle au centre de la partie gauche sont affichées dans le terminal en haut à gauche. Les 3 premières valeurs représentent respectivement les coordonnées x, y, et z du vecteur normal au plan, exprimées dans le repère de la caméra. La valeur de d correspond à la distance entre le centre de la caméra et le plan. Cette fois-ci, les calculs sont effectués à bord de l'ordinateur embarqué, un Raspberry Pi 4 (en noir, en bas à droite). Les résultats obtenus sont conformes à ce qui avait été obtenu dans la simulation. Cependant, les calculs des coordonnées du plan sont trop gourmands en ressources pour l'ordinateur compagnon, et un délai de l'ordre de la seconde est perceptible entre le calcul de ces coordonnées et leur valeur réelle. Pour cette raison, le repérage de la surface d'accostage se fait finalement avec les deux LiDARs TFMini Plus présentés dans le chapitre 3.

CHAPITRE 5 DISCUSSION GÉNÉRALE

Dans les travaux précédents entamés sur le même sujet, il était question d'utiliser ROS2 pour communiquer avec PX4 *via* un pont FastRTPS au lieu de MAVLink. L'intérêt était de pouvoir communiquer avec les contrôleurs de PX4 à un fréquence de 500 Hz au lieu des 30 Hz maximum imposés par le protocole de communication externe de PX4. L'objectif était d'écrire des contrôleurs, de les régler et d'envoyer des consignes d'attitude au drone. Mais au lieu de re-concevoir des contrôleurs déjà présents dans PX4 et de faire de la commande bas niveau, il a semblé judicieux d'envoyer des commandes plus haut niveau et de laisser les contrôleurs du PX4 les traiter. La prise en main de l'écosystème ROS pouvant être assez longue et fastidieuse, il est nécessaire de faire le bon choix entre ROS et ROS2 au début. Heureusement, il a assez tôt été mis en évidence que ROS2 n'était pas nécessaire, d'autant plus que sa relative nouveauté induisait une moindre communauté et des forums moins développés. Le choix de l'architecture logicielle de l'environnement de simulation doit être fait judicieusement, en prenant en compte sa capacité a être personnalisée, indispensable dans le cadre de ces travaux. Il peut paraître contre-intuitif de se tourner vers des solutions plus anciennes, mais celles-ci sont souvent moins gourmandes en calculs et les ressources d'aides disponibles – forums, documentation, exemples – sont plus fournies.

Comme il est fait mention dans la section 4.3.2 au sujet de l'ordinateur compagnon Raspberry Pi, des problèmes d'alimentation induisant un dégradation des performances ont été soulevés. Pour anticiper ces difficultés, il aurait fallu développer la stratégie d'accostage en utilisant le Raspberry Pi (RPi) en matériel dans la boucle (Hardware In The Loop, HITL). Ainsi, en exécutant l'algorithme d'accostage sur le RPi et non pas sur l'ordinateur de simulation, il aurait été possible de s'apercevoir du problème de délai de réponse (*RTT too high*) entre PX4 et MAVROS. Bien entendu, des tests d'intégration des capteurs ainsi que l'exécution d'algorithmes utilisant ces capteurs ont été effectués sur l'ordinateur compagnon, en parallèle du développement de l'accostage dans la simulation. Grâce à cela, il a été conclu que, à cause du temps de calcul, la caméra de profondeur Intel Realsense ne pouvait pas être utilisée pour que le drone repère sa position et son orientation par rapport à la surface d'accostage. Cela a laissé suffisamment de temps pour valider en simulation la solution des deux LiDARs mono-point et pour commander les capteurs appropriés. De la même façon que pour la caméra, la nouvelle solution de mesure de distance et d'angle au mur a ensuite été éprouvée avec le RPi, et a cette fois été validée. La démarche a été similaire pour l'ensemble des capteurs choisis mobilisés au service de l'accostage autonome. Cependant, pour tous ces tests, l'ordinateur compagnon était alimenté sur secteur et n'était donc pas dans les conditions exactes de

vol. Ce genre de vérification est cruciale pour assurer le succès du transfert de la stratégie d'accostage depuis la simulation vers le monde réel, et doit se faire dans les conditions les plus proches possibles des expériences finales.

Pour caractériser le DA, un concept de banc d'essai avait été établi. Le principe pour modéliser la ventouse consistait à :

- imposer une pression dans la ventouse
- mesurer les forces d'arrachement normale et tangentielle de la ventouse pour cette pression
- mesurer l'évolution de la pression au cours du scellage de la ventouse
- intégrer le modèle d'évolution de la pression dans la simulation
- vérifier que, lors du scellage de la ventouse dans la simulation, la force normale et la force tangentielle n'excèdent pas les valeurs d'arrachement établies en fonction de la pression

Pour cela le banc d'essai devait permettre de monter le DA sur un châssis et d'appliquer différentes forces de traction sur la ventouse. La pression devait pouvoir être contrôlée pour mesurer la force d'arrachement de la ventouse pour différentes valeurs de pression. Les essais de traction devaient se faire à l'aide d'une vis sans fin induisant un mouvement de translation au châssis. Une fois la fonction donnant les forces d'arrachement en fonction de la pression établie, on aurait été capable de prévoir la capacité du DA à maintenir l'attache à la surface, à partir du moment où la ventouse commencerait à être scellée au mur. Il fallait donc également mesurer quelle force de compression permettait de déclencher ce scellage. Ces caractérisations assez longue du DA auraient nécessité d'être répétées pour chaque version du DA, pour chaque surface sur laquelle on souhaitait simuler l'accostage. De plus, cette approche considérait que l'accrochage de la ventouse se faisait de façon statique, or en pratique, le drone impacte dynamiquement le mur pour y accoster. Cette stratégie pour tenter de prévoir les succès du DA risquait de prendre du temps à être menée à bien, et rien ne garantissait des résultats fiables au vu de l'écart entre le banc d'essai et un accostage réel. La conception et la réalisation du banc d'essai étaient donc superflues, et leur contribution à l'aboutissement du projet ne valaient pas l'investissement en temps qu'ils nécessitaient. Suite à ce constat, c'est la méthode de «l'accostage» pendulaire qui a été retenue, qui permet d'estimer la probabilité de succès d'accostage en fonction de la vitesse d'impact. Cette approche statistique ne permet pas de prévoir si tel accostage, avec telles vitesses angulaires du drone, sur telle portion du mur, va réussir ou non. Cependant, elle permet d'estimer rapidement la vitesse idéale d'impact pour avoir les meilleures chance de réussir l'accostage.

CHAPITRE 6 CONCLUSION

Synthèse des travaux

Dans ce mémoire, un environnement pour simuler des accostages de drones sur paroi verticale a été présentée. Cette simulation, construite avec les trois logiciels Gazebo, ROS, et PX4, comprend la dynamique du véhicule, ses interactions avec l'environnement de vol, la gestion des capteurs, la modélisation du système d'accostage, et les lois de contrôle du pilote automatique. Cette simulation a été éprouvée par le développement d'une stratégie d'accostage autonome. Cette stratégie, qui comprend le choix des capteurs et l'écriture de l'algorithme d'accostage, a ensuite été validée expérimentalement. Les expériences ont montré que la simulation mise en place constitue un environnement fidèle pour développer une stratégie d'accostage autonome, bien que Gazebo gagnerait à être amélioré pour un désaccostage plus fidèle à la réalité. Ces résultats sont prometteurs pour le développement futur d'accostages autonomes de drones lourds tels que les manipulateurs aériens, et garantissent une minimisation des accidents et une optimisation de la vitesse de développement.

Limitations de la solution proposée

Dans la simulation, des hypothèses réductrices doivent être mentionnées pour la modélisation du dispositif d'accostage :

- l'arrimage à la surface est considéré comme parfait (liaison encastrement)
- la condition de scellage ne dépend que de la distance de la ventouse au mur et de l'état de la pompe allumée/éteinte
- l'élasticité de l'anneau de mousse n'a pas été modélisé car considérée comme négligeable devant l'élasticité des ressorts du DA

Par ailleurs, les défauts du drone, comme la détérioration des hélices ou la variation des performances des moteurs n'est pas prise en compte dans la simulation. Aussi, dans le monde réel, le centrage du drone varie légèrement à chaque changement de batterie. Pour remédier à ces imprécisions, la dynamique du drone pourrait être établie expérimentalement en soufflerie, comme dans [65]. Enfin, les effet de sol ne sont pas modélisés dans Gazebo et cela pourrait modifier l'effort de traction appliqué sur le DA lors de la phase d'atterrissage sur le mur. Cela pourrait aussi expliquer l'absence d'oscillations, dans la simulation, de l'angle de tangage visible sur la Figure 3.18.

Toutefois, ces différences entre la simulation d'accostage et le monde réel ne remettent pas en

question la capacité de l'environnement de simulation à permettre le développement complet d'une stratégie d'accostage autonome.

Améliorations futures

La prochaine étape consiste à valider la stratégie d'accostage en extérieur, sur une pale d'éolienne ou un bâtiment par exemple. Maintenant qu'une simulation d'accostage est déployée et que son efficacité est validée, il s'agit de mettre au point une stratégie d'accostage autonome pour le drone Hercule et de la valider expérimentalement. Pendant le développement en simulation, l'algorithme d'accostage devra idéalement être exécuté sur l'ordinateur compagnon et PX4 devra être utilisé en HITL, pour se rapprocher le plus possible des conditions réelles. Le contrôle automatique de la pression dans la ventouse, une fois le drone accosté, doit être ajouté pour n'allumer la pompe que lorsque le niveau de vide est trop bas, afin d'économiser de l'énergie. Il faut pour cela étudier les caractéristiques du BAM développé au CNRC et maintenir la pression suffisante pour assurer un maintien sécuritaire du véhicule contre la paroi. L'environnement de simulation peut aussi être utilisé pour développer une manoeuvre de récupération du véhicule en cas de décrochage accidentel de la ventouse. Il faut également revenir sur l'utilisation d'une caméra de profondeur pour le placement précis du drone par rapport à la zone à accoster. Si les caméras Intel Realsense sont trop gourmandes en ressources, il peut être pertinent de se pencher sur des caméras comme la ZED 2 [1], qui proposent la prise en charge du traitement d'image. Concernant l'environnement de simulation, il peut être pertinent de chercher à travailler à une meilleure prise en compte des phénomènes aérodynamiques.

1. Caméra ZED 2 : `https://www.stereolabs.com/zed-2/`

RÉFÉRENCES

[1] B. Monsarrat, C. Vidal, M. De Montigny et I. Mantegh, "Base attachment module for small aerial vehicles," US Brevet US20 220 063 038A1, mars 2022. [En ligne]. Disponible : https://patents.google.com/patent/US20220063038A1/en

[2] H. Zheng, G. Gao, X. Zhong et L. Zhao, "Monitoring and diagnostics of buildings' heat loss based on 3D IR model of multiple buildings," *Energy and Buildings*, vol. 259, p. 111889, mars 2022. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0378778822000603

[3] Z. Shao, G. Cheng, J. Ma, Z. Wang, J. Wang et D. Li, "Real-Time and Accurate UAV Pedestrian Detection for Social Distancing Monitoring in COVID-19 Pandemic," *IEEE Transactions on Multimedia*, vol. 24, p. 2069–2083, 2022. [En ligne]. Disponible : https://doi.org/10.1109/TMM.2021.3075566

[4] H. Huang et A. V. Savkin, "An Algorithm of Reactive Collision Free 3-D Deployment of Networked Unmanned Aerial Vehicles for Surveillance and Monitoring," *IEEE Transactions on Industrial Informatics*, vol. 16, n°. 1, p. 132–140, janv. 2020. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/8701486

[5] L. Fauteux, N. Jolin, J. Eng, M. Wadham-Gagnon, M. Eng, N. Swytink-Binnema, C. Farley, G. Roy et D. Soares, "Drone Solutions for Wind Turbine Inspections," *Technical Report - Nergica : Gaspe, QC, Canada*, 2018. [En ligne]. Disponible : https://nergica.com/wp-content/uploads/%C3%89tude-Drones_Version-finale_EN_17-d%C3%A9cembre-2018.pdf

[6] C. Martinez, C. Sampedro, A. Chauhan et P. Campoy, "Towards autonomous detection and tracking of electric towers for aerial power line inspection," dans *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, mai 2014, p. 284–295. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/6842267?arnumber=6842267

[7] P. Hamelin, F. Mirallès, G. Lambert, S. Lavoie, N. Pouliot, M. Montfrond et S. Montambault, "Discrete-time control of LineDrone : An assisted tracking and landing UAV for live power line inspection and maintenance," dans *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, juin 2019, p. 292–298. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/8798137

[8] J. H. Lee, S. Yoon, B. Kim, G.-H. Gwon, I.-H. Kim et H.-J. Jung, "A new image-quality evaluating and enhancing methodology for bridge inspection using an unmanned aerial vehicle," *Smart Structures and Systems*, vol. 27, n°. 2, p. 209–226,

2021. [En ligne]. Disponible : https://www.techno-press.org/content/?page=article&journal=sss&volume=27&num=2&ordernum=6

[9] V. Baiocchi, D. Dominici et M. Mormile, "UAV application in post-seismic environment," *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.*, vol. XL-1/W2, p. 21–25, août 2013. [En ligne]. Disponible : https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XL-1-W2/21/2013/

[10] M. Car, L. Markovic, A. Ivanovic, M. Orsag et S. Bogdan, "Autonomous Wind-Turbine Blade Inspection Using LiDAR-Equipped Unmanned Aerial Vehicle," *IEEE Access*, vol. 8, p. 131 380–131 387, 2020. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/9142230?arnumber=9142230

[11] A. Kulsinskas, P. Durdevic et D. Ortiz-Arroyo, "Internal Wind Turbine Blade Inspections Using UAVs : Analysis and Design Issues," *Energies*, vol. 14, nº. 2, p. 294, janv. 2021. [En ligne]. Disponible : https://www.mdpi.com/1996-1073/14/2/294

[12] J. Aleotti, G. Micconi, S. Caselli, G. Benassi, N. Zambelli, D. Calestani, M. Zanichelli, M. Bettelli et A. Zappettini, "Unmanned aerial vehicle equipped with spectroscopic CdZnTe detector for detection and identification of radiological and nuclear material," dans *2015 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)*, oct. 2015, p. 1–5. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/7582264?arnumber=7582264

[13] S. Asian, G. Ertek, C. Haksoz, S. Pakter et S. Ulun, "Wind Turbine Accidents : A Data Mining Study," *IEEE Systems Journal*, vol. 11, nº. 3, p. 1567–1578, sept. 2017. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/7489036?arnumber=7489036

[14] J. Xiao, B. Li, K. Ushiroda et Q. Song, "Rise-Rover : A wall-climbing robot with high reliability and load-carrying capacity," dans *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, déc. 2015, p. 2072–2077. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7419079/

[15] A. Albers, S. Trautmann, T. Howard, T. A. Nguyen, M. Frietsch et C. Sauter, "Semi-autonomous flying robot for physical interaction with environment," dans *2010 IEEE Conference on Robotics, Automation and Mechatronics*, juin 2010, p. 441–446. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/5513152

[16] L. Marconi, "Modelling and control of a flying robot interacting with the environment," p. 13, 2011. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0005109811004602

[17] D. Mellinger, Q. Lindsey, M. Shomin et V. Kumar, "Design, modeling, estimation and control for aerial grasping and manipulation," dans *2011 IEEE/RSJ International*

Conference on Intelligent Robots and Systems, sept. 2011, p. 2668–2673. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/6094871?arnumber=6094871

[18] H. W. Wopereis, J. J. Hoekstra, T. H. Post, G. A. Folkertsma, S. Stramigioli et M. Fumagalli, "Application of substantial and sustained force to vertical surfaces using a quadrotor," dans *2017 IEEE International Conference on Robotics and Automation (ICRA)*, p. 2704–2709. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7989314/

[19] F. Forte, R. Naldi, A. Macchelli et L. Marconi, "Impedance control of an aerial manipulator," dans *2012 American Control Conference (ACC)*, juin 2012, p. 3839–3844. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/6315568?arnumber=6315568

[20] S. Kim, S. Choi et H. J. Kim, "Aerial manipulation using a quadrotor with a two DOF robotic arm," dans *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, nov. 2013, p. 4990–4995. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/6697077

[21] A. Jimenez-Cano, J. Martin, G. Heredia, A. Ollero et R. Cano, "Control of an aerial robot with multi-link arm for assembly tasks," dans *2013 IEEE International Conference on Robotics and Automation*, p. 4916–4921. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/6631279

[22] G. Heredia, A. Jimenez-Cano, I. Sanchez, D. Llorente, V. Vega, J. Braga, J. Acosta et A. Ollero, "Control of a multirotor outdoor aerial manipulator," dans *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, sept. 2014, p. 3417–3422. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/6943038

[23] T. W. Danko, K. P. Chaney et P. Y. Oh, "A parallel manipulator for mobile manipulating UAVs," dans *2015 IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*, mai 2015, p. 1–6. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/7219682

[24] M. Orsag, C. M. Korpela, S. Bogdan et P. Y. Oh, "Hybrid Adaptive Control for Aerial Manipulation," *J Intell Robot Syst*, vol. 73, n°. 1, p. 693–707, janv. 2014. [En ligne]. Disponible : https://doi.org/10.1007/s10846-013-9936-1

[25] E. Cataldi, G. Muscio, M. A. Trujillo, Y. Rodriguez, F. Pierri, G. Antonelli, F. Caccavale, A. Viguria, S. Chiaverini et A. Ollero, "Impedance Control of an aerial-manipulator : Preliminary results," dans *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, oct. 2016, p. 3848–3853. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/7759566?arnumber=7759566

[26] S. TV, S. Shankar S, V. P. Yashvanth, S. K. Perepu, S. Mohan et S. Mohan, "An Autonomous Drone with Robotic End Effector for Maintenance Operation of 5G Mobile Towers," dans *2020 3rd International Conference on Control and Robots (ICCR)*, déc. 2020, p. 11–16. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/9344140?arnumber=9344140

[27] F. Ruggiero, M. Trujillo, R. Cano, H. Ascorbe, A. Viguria, C. Perez, V. Lippiello, A. Ollero et B. Siciliano, "A multilayer control for multirotor UAVs equipped with a servo robot arm," dans *2015 IEEE International Conference on Robotics and Automation (ICRA)*, p. 4014–4020. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7139760/

[28] G. M. Crutsinger, J. Short et R. Sollenberger, "The future of UAVs in ecology : an insider perspective from the Silicon Valley drone industry," *J. Unmanned Veh. Sys.*, vol. 4, n°. 3, p. 161–168, sept. 2016. [En ligne]. Disponible : http://www.nrcresearchpress.com/doi/10.1139/juvs-2016-0008

[29] W. R. T. Roderick, M. R. Cutkosky et D. Lentink, "Touchdown to take-off : at the interface of flight and surface locomotion," *Interface Focus.*, vol. 7, n°. 1, p. 20160094, févr. 2017. [En ligne]. Disponible : https://royalsocietypublishing.org/doi/10.1098/rsfs.2016.0094

[30] D. Mellinger, N. Michael et V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *The International Journal of Robotics Research*, vol. 31, n°. 5, p. 664–674, avr. 2012. [En ligne]. Disponible : https://doi.org/10.1177/0278364911434236

[31] M. Kovač, J. Germann, C. Hürzeler, R. Y. Siegwart et D. Floreano, "A perching mechanism for micro aerial vehicles," *J. Micro-Nano Mech.*, vol. 5, n°. 3-4, p. 77–91, déc. 2009. [En ligne]. Disponible : http://link.springer.com/10.1007/s12213-010-0026-1

[32] S. Backus, J. Izraelevitz, J. Quan, R. Jitosho, E. Slavick et A. Kalantari, "Design and Testing of an Ultra-Light Weight Perching System for Sloped or Vertical Rough Surfaces on Mars," dans *2020 IEEE Aerospace Conference*, mars 2020, p. 1–12. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/9172387?arnumber=9172387

[33] K. Hang, X. Lyu, H. Song, J. A. Stork, A. M. Dollar, D. Kragic et F. Zhang, "Perching and resting—A paradigm for UAV maneuvering with modularized landing gears," *Sci. Robot.*, vol. 4, n°. 28, p. eaau6637, mars 2019. [En ligne]. Disponible : https://robotics.sciencemag.org/lookup/doi/10.1126/scirobotics.aau6637

[34] A. Kalantari, K. Mahajan, D. Ruffatto et M. Spenko, "Autonomous perching and take-off on vertical walls for a quadrotor micro air vehicle," dans *2015 IEEE*

International Conference on Robotics and Automation (ICRA), p. 4669–4674. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/7139846

[35] H. W. Wopereis, T. D. van der Molen, T. H. Post, S. Stramigioli et M. Fumagalli, "Mechanism for perching on smooth surfaces using aerial impacts," dans *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, p. 154–159. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/7784292

[36] H. Wopereis, D. Ellery, T. Post, S. Stramigioli et M. Fumagalli, "Autonomous and sustained perching of multirotor platforms on smooth surfaces," dans *2017 25th Mediterranean Conference on Control and Automation (MED)*, juill. 2017, p. 1385–1391. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/7984312

[37] Y. Liu, G. Sun et H. Chen, "Optimal design of the flying and adhesive robot," dans *2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA)*, juin 2013, p. 1101–1105. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/6566531?arnumber=6566531

[38] S. Du, H. Chen, Y. Liu et R. Hu, "Unified Switching between Active Flying and Perching of a Bioinspired Robot Using Impedance Control," *Journal of Robotics*, vol. 2015, p. 1–11, 2015. [En ligne]. Disponible : http://www.hindawi.com/journals/jr/2015/763710/

[39] H. Tsukagoshi, M. Watanabe, T. Hamada, D. Ashlih et R. Iizuka, "Aerial manipulator with perching and door-opening capability," dans *2015 IEEE International Conference on Robotics and Automation (ICRA)*, mai 2015, p. 4663–4668. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/7139845?arnumber=7139845

[40] S. Liu, W. Dong, Z. Ma et X. Sheng, "Adaptive Aerial Grasping and Perching With Dual Elasticity Combined Suction Cup," *IEEE Robotics and Automation Letters*, vol. 5, nº. 3, p. 4766–4773, juill. 2020. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/9121725?arnumber=9121725

[41] Y. Liu, H. Chen, Z. Tang et G. Sun, "A bat-like switched flying and adhesive robot," dans *2012 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, mai 2012, p. 92–97. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/6392533

[42] W. Chi, K. H. Low, K. H. Hoon et J. Tang, "An optimized perching mechanism for autonomous perching with a quadrotor," dans *2014 IEEE International Conference on Robotics and Automation (ICRA)*, mai 2014, p. 3109–3115. [En ligne]. Disponible : https://ieeexplore.ieee.org/abstract/document/6907306

[43] J. Thomas, G. Loianno, K. Daniilidis et V. Kumar, "Visual Servoing of Quadrotors for Perching by Hanging From Cylindrical Objects," *IEEE Robot.*

Autom. Lett., vol. 1, nº. 1, p. 57–64, janv. 2016. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7347389/

[44] H. Paul, K. Ono, R. Ladig et K. Shimonomura, "A Multirotor Platform Employing a Three-Axis Vertical Articulated Robotic Arm for Aerial Manipulation Tasks," dans *2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, p. 478–485. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/8452699/

[45] D. Mellinger, M. Shomin et V. Kumar, "Control of Quadrotors for Robust Perching and Landing," dans *Proceedings of the International Powered Lift Conference*, 2010, p. 205–225. [En ligne]. Disponible : http://www.it.uu.se/edu/course/homepage/modbasutv/vt12/Material/IPLCmellinger.pdf

[46] J. Thomas, M. Pope, G. Loianno, E. W. Hawkes, M. A. Estrada, H. Jiang, M. R. Cutkosky et V. Kumar, "Aggressive Flight With Quadrotors for Perching on Inclined Surfaces," *Journal of Mechanisms and Robotics*, vol. 8, nº. 5, p. 051007, oct. 2016. [En ligne]. Disponible : https://asmedigitalcollection.asme.org/mechanismsrobotics/article/doi/10.1115/1.4032250/383963/Aggressive-Flight-With-Quadrotors-for-Perching-on

[47] Pope, Morgan, "Microspines Make It Easy for Drones to Perch on Walls and Ceilings," mai 2016. [En ligne]. Disponible : https://spectrum.ieee.org/microspines-make-it-easy-for-drones-to-perch-on-walls-and-ceilings

[48] M. De Benedetti, F. D'Urso, F. Messina, G. Pappalardo et C. Santoro, "3D simulation of unmanned aerial vehicles," dans *18th Workshop "From Objects to Agents", WOA 2017, June 15, 2017 - June 16, 2017*, ser. CEUR Workshop Proceedings, vol. 1867, p. 7–12. [En ligne]. Disponible : http://ceur-ws.org/Vol-1867/w2.pdf

[49] Y. Alborzi, B. S. Jalal et E. Najafi, "ROS-based SLAM and Navigation for a Gazebo-Simulated Autonomous Quadrotor," dans *2020 21st International Conference on Research and Education in Mechatronics (REM)*, déc. 2020, p. 1–5. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/9313875?arnumber=9313875&tag=1

[50] H. Liu, P. Guo, X. Jin, H. Zhang, H. Deng, K. Xu et X. Ding, "Collaborative Robots Sim : A Simulation Environment Of Air-Ground Robots With Strong Physical Interactivity," dans *2021 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, déc. 2021, p. 1841–1847. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/9739544?arnumber=9739544

[51] S. Chen, W. Zhou, A.-S. Yang, H. Chen, B. Li et C.-Y. Wen, "End-to-End UAV Simulation Platform for Visual SLAM and Navigation," *Aerospace*, vol. 9, nº. 2, p. 48, janv. 2022. [En ligne]. Disponible : https://www.mdpi.com/2226-4310/9/2/48

[52] M. Schmittle, A. Lukina, L. Vacek, J. Das, C. P. Buskirk, S. Rees, J. Sztipanovits, R. Grosu et V. Kumar, “OpenUAV : A UAV Testbed for the CPS and Robotics Community,” dans *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, p. 130–139. [En ligne]. Disponible : https://publik.tuwien.ac.at/files/publik_275473.pdf

[53] K. Xiao, S. Tan, G. Wang, X. An, X. Wang et X. Wang, “XTDrone : A Customizable Multi-rotor UAVs Simulation Platform,” dans *2020 4th International Conference on Robotics and Automation Sciences (ICRAS)*, p. 55–61. [En ligne]. Disponible : https://arxiv.org/pdf/2003.09700.pdf

[54] S. Chen, H. Chen, W. Zhou, C.-Y. Wen et B. Li, “End-to-End UAV Simulation for Visual SLAM and Navigation,” *arXiv :2012.00298 [cs]*, déc. 2020. [En ligne]. Disponible : http://arxiv.org/abs/2012.00298

[55] C. Ma, Y. Zhou et Z. Li, “A New Simulation Environment Based on Airsim, ROS, and PX4 for Quadcopter Aircrafts,” dans *2020 6th International Conference on Control, Automation and Robotics (ICCAR)*, avr. 2020, p. 486–490. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/9108103?arnumber=9108103

[56] V. Casas et A. Mitschele-Thiel, “From simulation to reality : a implementable self-organized collision avoidance algorithm for autonomous UAVs,” dans *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, p. 822–831. [En ligne]. Disponible : https://www.db-thueringen.de/servlets/MCRFileNodeServlet/dbt_derivate_00053310/ilm1-2021000074.pdf

[57] “NVIDIA Isaac Sim,” déc. 2019. [En ligne]. Disponible : https://developer.nvidia.com/isaac-sim

[58] C. A. Coulomb, *Théorie des machines simples en ayant égard au frottement de leurs parties et à la roideur des cordages.* Bachelier, 1821. [En ligne]. Disponible : https://books.google.ca/books?id=VEYOAAAAYAAJ

[59] J. Liu, K. Tanaka, L. Bao et I. Yamaura, “Analytical modelling of suction cups used for window-cleaning robots,” *Vacuum*, vol. 80, nº. 6, p. 593–598, mars 2006. [En ligne]. Disponible : https://linkinghub.elsevier.com/retrieve/pii/S0042207X05003350

[60] A. Bernardin, C. Duriez et M. Marchal, “An Interactive Physically-based Model for Active Suction Phenomenon Simulation,” dans *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, p. 1466–1471. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/8967526/

[61] J. Hudoklin, S. Seo, M. Kang, H. Seong, A. T. Luong et H. Moon, “Vacuum Suction Cup Modeling for Evaluation of Sealing and Real-Time Simulation,” *IEEE Robotics*

and Automation Letters, vol. 7, n°. 2, p. 3616–3623, avr. 2022. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/9691809

[62] K. Máthé et L. Buşoniu, "Vision and Control for UAVs : A Survey of General Methods and of Inexpensive Platforms for Infrastructure Inspection," *Sensors*, vol. 15, n°. 7, p. 14 887–14 916, juin 2015. [En ligne]. Disponible : http://www.mdpi.com/1424-8220/15/7/14887

[63] S. Sheng et R. O'Connor, "Chapter 15 : Reliability of Wind Turbines," London, UK : Academic Press, Rapport technique NREL/CH-5000-67496, mai 2017. [En ligne]. Disponible : https://www.osti.gov/biblio/1360890

[64] A. Lussier Desbiens et M. R. Cutkosky, "Landing and Perching on Vertical Surfaces with Microspines for Small Unmanned Air Vehicles," *J Intell Robot Syst*, vol. 57, n°. 1-4, p. 313–327, janv. 2010. [En ligne]. Disponible : http://link.springer.com/10.1007/s10846-009-9377-z

[65] J. Meyer, A. Sendobry, S. Kohlbrecher, U. Klingauf et O. von Stryk, "Comprehensive Simulation of Quadrotor UAVs Using ROS and Gazebo," ser. Lecture Notes in Computer Science, I. Noda, N. Ando, D. Brugali et J. J. Kuffner, édit., p. 400–411. [En ligne]. Disponible : https://asset-pdf.scinapse.io/prod/1501966803/1501966803.pdf

[66] C. Korpela, M. Orsag et P. Oh, "Hardware-in-the-Loop Verification for Mobile Manipulating Unmanned Aerial Vehicles," *J Intell Robot Syst*, vol. 73, p. 725–736, 2014. [En ligne]. Disponible : https://doi.org/10.1007/s10846-013-9950-3

[67] K. Schauwecker et A. Zell, "On-Board Dual-Stereo-Vision for the Navigation of an Autonomous MAV," *J Intell Robot Syst*, vol. 74, n°. 1-2, p. 1–16, avr. 2014. [En ligne]. Disponible : http://link.springer.com/10.1007/s10846-013-9907-6

[68] Y. Xiang, D. Li, T. Su, Q. Zhou, C. Brach, S. S. Mao et M. Geimer, "Where Am I ? SLAM for Mobile Machines on a Smart Working Site," *Vehicles*, vol. 4, n°. 2, p. 529–552, mai 2022. [En ligne]. Disponible : https://www.mdpi.com/2624-8921/4/2/31

[69] G. Flores, S. Zhou, R. Lozano et P. Castillo, "A Vision and GPS-Based Real-Time Trajectory Planning for a MAV in Unknown and Low-Sunlight Environments," *J Intell Robot Syst*, vol. 74, n°. 1-2, p. 59–67, avr. 2014. [En ligne]. Disponible : http://link.springer.com/10.1007/s10846-013-9975-7

[70] wiki, "pcl/Tutorials - ROS Wiki." [En ligne]. Disponible : http://wiki.ros.org/pcl/Tutorials

ANNEXE A UTILISER LA SIMULATION DU DA

Maintenant qu'il est développé, l'ajout d'un DA dans la simulation nécessite de :

1. installer le module ROS `gazebo_ros_link_attacher`
2. installer le module ROS `my_package` comprenant `pressure_emulator.py` et `pump.py`
3. modifier le script `pressure_emulator.py` pour :
   (a) Définir dans le module, le nom du *model* et du *link* qui jouera le rôle de ventouse
   (b) spécifier les conditions d'arrimage du DA : force et distance
   (c) spécifier la pression atmosphérique et la pression minimale de vide
   (d) éventuellement modifier la fonction `vacuum_create(t)` qui modélise l'évolution de la pression lorsque le scellage est amorcé
4. Modifier et placer le modèle de ventouse proposé qui comprend un capteur de force (module `gazebo_ros_ft`), et un capteur de distance (module `gazebo_ros_range`). Les noms des thèmes ROS doivent rester les mêmes car le noeud de `pressure_emulator.py` y est abonné.

Ensuite, en plus des actions pour lancer la simulation de vol de drone habituelle, l'utilisateur doit exécuter les 2 scripts python `pressure_emulator.py` et `pump.py` et s'assurer de lancer la simulation avec le monde monde test_attacher.world, qui inclut le module `gazebo_ros_link_attacher`. Il doit également contrôler l'état de la pompe dans un terminal où aura été exécuté `pump.py`.

Pour faciliter l'utilisation de la simulation, les 2 modules devraient être fusionnés en un seul et la gestion de la pompe devrait être facilitée.