Titre: Title:	Support à la conception intégrée de manipulateurs aériens
Auteur: Author:	Charles Coulombe
Date:	2021
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Coulombe, C. (2021). Support à la conception intégrée de manipulateurs aériens [Thèse de doctorat, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/9093/

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Support à la conception intégrée de manipulateurs aériens

CHARLES COULOMBE
Département de génie mécanique

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
Génie mécanique

Juin 2021

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Cette thèse intitulée :

Support à la conception intégrée de manipulateurs aériens

présentée par **Charles COULOMBE**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

Aurelian VADEAN, président
Sofiane ACHICHE, membre et directeur de recherche
David SAUSSIÉ, membre et codirecteur de recherche
Giovanni BELTRAME, membre
David ST-ONGE, membre externe

DÉDICACE

À Marie-Anne, Mathieu et Ophélie

REMERCIEMENTS

J'aimerais remercier mon directeur de recherche Pr. Sofiane Achiche pour son support, pour l'indépendance qu'il m'a laissée durant ce projet, ainsi que pour le temps pris à me transmettre ses connaissances et expériences du domaine académique. Je lui suis aussi particulièrement reconnaissant de m'avoir offert toutes ces opportunités d'enseignement depuis le début. J'aimerais aussi remercier le Pr. David Saussié pour sa codirection. Les discussions techniques sur la commande des systèmes et les drones en général, m'ont été d'un support inestimable. Je conserve un très bon souvenir des moments passés dans son bureau à dessiner des schémas pour répondre à mes innombrables questions.

Je veux remercier tous les étudiants du laboratoire CoSIM, passés et présents. D'une manière ou d'une autre, ils ont tous contribué à ce doctorat, soit par de l'aide technique, des discussions philosophiques, des parties de cartes, des activités en dehors du labo ou juste leur bonne humeur. La pandémie m'a fait réaliser à quel point vous étiez important. Dans un ordre complètement aléatoire : Olivier ($\times$2), Luis, Pr. Abolfazl, Martin, Alexandre, Gabriel ($\times$3), Jean-François, Ugo, Bahare, Yann, Cédric, Samuel, Guillaume, Laurent, Mathieu, Dominique, Mireille, ainsi que tous les stagiaires et nouveaux étudiants que je n'ai malheureusement pas pu connaître à cause de la pandémie.

Pour terminer, je veux remercier ma conjointe Marie-Anne, pour son support, ses encouragements, son aide avec les figures, et surtout son endurance à mon tempérament. Les dernières années (et la dernière spécifiquement) n'ont pas été de tout repos, et je n'aurais certainement pas pu passer à travers sans elle.

RÉSUMÉ

Les manipulateurs aériens (UAM - *Unmanned aerial manipulator*) sont des systèmes robotiques aériens combinant l'agilité de leur base, sous forme de multicoptère, à la capacité de manipulation du ou des bras robotiques montés. Ils peuvent être utilisés dans des contextes où leur manœuvrabilité, leur petite taille et leur autonomie en font des alliés intéressants, comme dans des missions de récupération à la suite de catastrophes ou encore dans des entrepôts automatisés. Les tâches pouvant être accomplies par les UAM sont variées, pouvant aller de la simple préhension d'objets pour leur transport, au perchage ou à la fermeture de valves. Afin d'accomplir ces tâches, les UAM doivent être équipés de nombreux sous-systèmes interagissant entre eux. La polyvalence de l'UAM complexifie le processus de conception, car il doit tenir compte de la ou des tâches pour lesquelles il sera utilisé.

Cette thèse propose le développement de méthodes de support à la conception pour la commande et la planification de missions. Dans le cadre de la conception d'un UAM, ou de tout produit mécatronique, le système doit être considéré dans son ensemble afin d'obtenir un design optimal. Cependant, la complexité de conception associée aux nombreux sous-systèmes nécessaires peut forcer les concepteurs à se concentrer sur la conception détaillée de ceux plus critiques. Il est alors possible de supporter la conception en utilisant des méthodes facilitant l'intégration. Puisque les requis de conceptions pour les sous-systèmes varient en fonction de la tâche pour laquelle ils sont conçus, ces méthodes sont développées spécifiquement pour une tâche. Dans le cas présent, les tâches de saisies d'objets statiques sont étudiées.

Tout d'abord, les tâches pouvant être accomplies ont été colligées en une taxonomie permettant de les catégoriser selon le type de contact nécessaire, la description de la tâche, les conditions environnantes ainsi que la présence de contraintes temporelles. La catégorisation a été obtenue suite à une revue de littérature du domaine ainsi qu'une méthodologie d'acquisition d'information par vidéo. Au même titre que les taxonomies de type de saisie avec une main, cette taxonomie permet de supporter la conception d'un UAM en simplifiant la description des tâches et en facilitant ainsi la définition du problème de conception. Elle permet aussi de baser la description d'une mission complexe en plusieurs tâches simples.

Par la suite, une méthode de support à la conception de lois de commande pour UAM est développée sur la base de la synthèse $\mathcal{H}_\infty$ structurée. Le correcteur est conçu à partir de la modélisation dynamique linéarisée de l'UAM. Cette dernière est complexe au fait des interactions hautement couplées entre le multicoptère et le bras robotique. Les équations dynamiques générales d'un UAM sont dérivées à partir de la dynamique d'un multicoptère

combinée à celle du bras robotique, obtenue par la méthode de Newton-Euler récursive. La structure de correction choisie possède des gains séquencés en fonction des angles des articulations du bras robotique. Le correcteur obtenu par la synthèse $\mathcal{H}_\infty$ structurée permet de stabiliser le quadricoptère en plus de répondre au cahier des charges défini pour toutes les positions du bras robotique, soit un amortissement supérieur à 0.4 pour les pôles du système et une erreur en régime permanent inférieure à 0.1% pour les signaux contrôlés. Le correcteur développé est aussi robuste à une variation des valeurs de paramètres physiques de 10% par rapport à ceux utilisés lors de l'obtention des gains.

Pour terminer, le second sous-système étudié est une méthode pour la sélection optimale d'un UAM dans une flotte en fonction de la tâche de saisie à accomplir. La sélection est effectuée à partir d'un classificateur obtenu par apprentissage profond. Le classificateur permet de choisir l'unité la plus apte à accomplir la tâche parmi ceux disponibles dans la flotte, en prenant en entrée une description de l'environnement local, ainsi qu'une description de la tâche. L'environnement est représenté sous la forme d'un nuage de points, tandis que la tâche est définie par l'orientation de la saisie, sous la forme d'un quaternion. La position de la tâche est considérée comme l'origine du nuage de points. Afin d'entraîner le classificateur, une base de données de 20000 éléments est obtenue en combinant aléatoirement des bases de données existantes pour générer des scènes et des tâches. Les étiquettes associées à chacun des éléments sont obtenues par une méthode à base de planification de trajectoire pour sélectionner l'UAM le plus efficace à accomplir la tâche. Le classificateur est entraîné avec la base de données créée et obtient une précision de classification de 70%. Une expérience avec des scènes réelles permet de confirmer une précision similaire et donc que l'entraînement avec des données simulées peut être utilisé avec des scènes et des tâches réelles. La méthode de sélection par classificateur est alors 100 fois plus rapide que celle à base de planification de trajectoire.

ABSTRACT

Unmanned aerial manipulators (UAMs) are aerial robotic systems combining the agility of their multicopter base with the manipulation capabilities of the mounted robotic arm(s). They can be used in situations where their maneuverability, small size and autonomy make them interesting allies, such as in disaster recovery missions or in automated warehouses. The tasks that can be performed by UAMs are varied; simple gripping, valves closing, perching and force application are some of them. In order to accomplish these tasks, UAMs must be equipped with numerous subsystems that interact with each other. Their versatility complicates the design process, as it must take into account the tasks for which the UAM will be used.

This thesis proposes the development of design support method for control and mission planning of a UAM. When designing a UAM or any mechatronic product, integrated design is necessary in order to obtain an optimal system. However, the complexity of designing individual subsystems can force designers to focus on the detailed design of those most critical. Design support methods allow designers to simplify this process and thus promote integration and the integrated design aspect. Since the design requirements for subsystems may vary depending on the task for which they are designed, these methods are developed specifically for a static object grasping.

First, the tasks that can be performed are summarized in a taxonomy where they are categorized according to the type of contact required, the task description, the surrounding conditions and the presence of temporal constraints. The categorization was obtained following a literature review of the field as well as a video information acquisition methodology, in which the most recent advances in the field were collected. Like the grasping type taxonomies, this taxonomy supports the design of UAMs by simplifying the task description and thus facilitating the definition of the design problem. It also allows the separation of complex missions into multiple simple tasks.

Subsequently, the development of a design support method for a UAM controller is based on the $\mathcal{H}_\infty$ structured controller synthesis framework. The controller is designed on the linearized dynamic modeling of the UAM. The latter is complex due to the highly coupled interactions between the multicopter and the robotic arm. The general dynamic equations of a UAM are derived from the dynamics of a multicopter combined with that of the robotic arm, obtained by the recursive Newton-Euler method. The chosen controller structure is gain-scheduled with respect to the robotic arm joint angles. The controller obtained by

the structured $\mathcal{H}_\infty$ synthesis method allows to stabilize the quadcopter base in addition to answering the specifications defined at all positions of the robotic arm; a pole damping higher than 0.4 and a steady state error inferior to 0.1% for the controlled signals. It is also robust to variations in the physical parameter values up to 10% compared to those used during the controller tuning.

Finally, the second subsystem studied is a method for the optimal selection of a UAM in a fleet according to the grasping task at hand. The selection is performed based on a deep learning classifier. The classifier is used to select the most suitable unit to perform the task among those available in the fleet, using a description of the local environment and the task as input. The environment is represented as a point cloud, while the task is defined by a quaternion representing the grasping orientation. The position of the task is considered as the origin of the point cloud. In order to train the classifier, a database of 20000 elements is generated by randomly combining existing databases into novel scenes and tasks. The labels associated with each of the features are obtained using a path planning based method to select the most efficient UAM to accomplish the task. The classifier is trained with the created database and reaches a classification accuracy of 70%. An experiment with real scenes confirms similar accuracy levels and thus that the model trained with simulated data can also be used with real scenes and tasks. The classifier-based selection method is 100 times faster than the one using path planning.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

CoM	Centre de masse / Centre of Mass
DDL (DOF)	Degré de liberté / Degree of Freedom
DL	Apprentissage profond / Deep Learning
GNC	Guidage, navigation et commande / Guidance, Navigation and Control
IBVS	Commande par vision avec images / Image-Based Visual Servoing
LQR	Régulateur linéaire-quadratique / Linear-Quadratic Regulator
LTI	Système linéaire à temps invariant / Linear Time Invariant System
MIMO	Plusieurs entrées, plusieurs sorties / Multiple Inputs Multiple Outputs
ML	Apprentissage machine / Machine Learning
MLP	Perceptron à plusieurs couches / MultiLayer Perceptron
NED	Système de référence Nord-Est-Sol / North-East-Down Reference Frame
PID	Contrôleur proportionnel-dérivé-intégral / Proportional-Integral-Derivative Controller
ROS	Robot Operating System
RRT	Arbre aléatoire à exploration rapide / Rapidly-exploring Random Tree
SISO	Une entrée, une sortie / Single Input Single Output
SO	Sous-objectif / Sub-objective
UAM (AM)	Manipulateur aérien / Unmanned Aerial Manipulator
UAV	Véhicule aérien autonome / Unmanned Aerial Vehicle

CHAPITRE 1 INTRODUCTION

1.1 Mise en contexte et définition du problème

Les multicoptères sont des robots pouvant se déplacer dans les airs à l'aide de paires d'hélices complémentaires. On les rencontre plus souvent sous les formes de quadricoptères, hexacoptères ou octocoptères, possédant quatre, six ou huit hélices, respectivement. Leur facilité d'utilisation, leur petite taille et leur manœuvrabilité favorisent leur usage dans différents domaines tels que la surveillance des milieux côtiers [1] ou l'inspection de structures [2]. Leur appropriation rapide, autant par la recherche académique que dans des utilisations commerciales, a favorisé le développement technologique de ces robots. Ils demeurent cependant toujours limités par leur principale faiblesse, soit leur faible capacité à interagir avec leur environnement. En effet, les multicoptères ne peuvent que mesurer leur environnement à l'aide de capteurs embarqués. C'est donc pour pallier ces manques que le concept de manipulateurs aériens (UAM - *Unmanned Aerial Manipulator*) a été développé [3], combinant à la fois, une plateforme aérienne et un système de manipulation dans un même robot aérien.

Alors que les UAM peuvent prendre plusieurs formes, les plus communs sont composés d'une base aérienne sous forme de multicoptère ainsi que d'un bras robotique sériel à plusieurs degrés de liberté (DDL). L'ajout d'un bras robotique permet à ces systèmes d'intervenir dans des tâches et des missions plus complexes que les simples véhicules aériens autonomes (UAV - *Unmanned Aerial Vehicle*), grâce aux avantages de la plateforme de type multicoptère et en faisant varier les configurations du manipulateur ainsi que de son outil. La communauté de robotique a identifié plusieurs avenues à explorer pour les années à venir, incluant notamment la saisie d'objet automatisée, ainsi que la mitigation et le recouvrement lors de catastrophes [4]. Ce type de contexte est tout à fait adapté à l'utilisation d'UAM qui peuvent alors exploiter leur petite taille, leur manœuvrabilité et leur autonomie.

L'accomplissement de ces tâches demeure cependant difficile, en partie à cause de l'apparition de nouveaux défis de conception s'ajoutant à ceux déjà présents pour les multicoptères. On note notamment les défis liés à la conception des lois de commande qui doivent tenir compte d'une dynamique complexifiée, [5], ainsi que ceux liés à la nécessité de concevoir des méthodes de planification de tâches et de missions haut-niveau, c'est-à-dire la génération automatique des différentes parties composant une tâche [4, 6].

Les UAM, comme tout système mécatronique, sont composés de sous-systèmes mécaniques, électriques, logiciels et de commande [7]. Alors que cette multidisciplinarité favorise le dé-

veloppement de systèmes plus innovants [8], elle est aussi une source de potentiels échecs de conception dus à l'augmentation de la complexité de l'architecture du système [9]. L'élaboration de méthodes de support à la conception pour certains sous-systèmes critiques lors du design détaillé permettrait de faciliter le processus global de conception. Ces méthodes permettraient aux concepteurs de se concentrer sur les étapes de conception intégrée, soit celles qui permettent l'obtention d'un produit optimal en simplifiant la conception détaillée par domaine [10]. La possibilité d'accomplissement de plusieurs tâches s'ajoute aux défis de conception, favorisant le besoin de méthodes spécifiquement adaptées aux sous-systèmes nécessaires et considérant les tâches pour lesquelles le système est conçu [11].

Les nombreuses possibilités de développements se rapportant aux UAM et à leurs capacités d'interactions ne peuvent bien évidemment pas être toutes abordées dans une seule thèse. Le cadre de ce travail a donc été limité à des cas où il était possible d'apporter un support à la conception de sous-systèmes, appliqué à une tâche particulière.

Tout d'abord, l'ajout d'un bras robotique sous un multicoptère complexifie la dynamique à cause des interactions entre les deux systèmes. Une loi de commande de vol permettant de compenser les effets dynamiques du manipulateur et de stabiliser le système est alors nécessaire [12]. Comme montré dans la revue de littérature au chapitre 2, plusieurs lois de commandes non-linéaires ont déjà été proposées. Bien que performantes et robustes aux variations, notamment au niveau des incertitudes de modélisation, elles sont profondément ancrées dans la théorie de la commande non-linéaire, les rendant difficiles à analyser [13] et à adapter à différentes configurations. Une méthode d'aide à la conception adaptée se basant sur une optimisation pourrait simplifier l'obtention d'une telle loi de commande. Ceci aurait pour effet de réduire les efforts nécessaires à son obtention, comme dans le cas des lois de commande non-linéaires devant être ajustées par des experts et où l'obtention de performances dynamiques désirées est plus difficile à atteindre.

Les capacités d'interactions des UAM ouvrent la porte à de nouveaux types de tâches en opposition aux multicoptères [14]. Cependant, les travaux effectués sont encore très préliminaires et appliqués au cas par cas, sans cadre général permettant de classifier les tâches et de guider le processus de design du manipulateur aérien pour ces tâches spécifiques. L'introduction de ces tâches amène aussi de nouvelles possibilités de planification haut-niveau des tâches [3], sujet qui est très peu abordé dans la littérature [15]. La planification haut-niveau inclut les prises de décisions et la planification de missions et est résumée par le terme guidage. Un exemple de ce type de sous-système est utilisé dans une mission de sauvetage après une catastrophe, où un UAM doit reconnaître des victimes humaines et leur déployer un bracelet émetteur pour leur sauvetage et suivi [16]. Comme le guidage doit généralement

être adapté à son utilisation spécifique, une méthode de support à leur conception faciliterait leurs développements et implémentations.

Le domaine de la manipulation robotique aérienne est encore suffisamment récent rendant l'introduction de nouveaux sous-systèmes soit encore possible et même nécessaire. L'arrivée de l'apprentissage machine et de l'apprentissage profond vient contribuer au développement d'UAM plus autonomes, notamment au niveau de la prise de décision et de l'identification [4]. Ceci s'ajoute aux questions de recherches ouvertes communes au domaine de la robotique mobile, telles que les applications de visions par ordinateur, la localisation et la cartographie simultanée, ou encore l'interaction physique avec l'environnement, qui ne sont pas couvertes par cette thèse. Le développement de méthodes de support à la conception faciliterait l'implémentation et le design d'UAM et favoriserait leur déploiement à grande échelle, pour un usage courant.

1.2 Portée et objectifs de recherche

En plus des défis habituels liés à la conception de systèmes mécatroniques, comme leur multidisciplinarité et leur complexité, chaque sous-système d'un UAM est techniquement complexe à concevoir. Puisque leur processus de conception varie selon les tâches à accomplir, il a été choisi pour cette thèse de travailler avec des tâches de saisies d'objets. L'objectif principal de ce travail est donc de développer des outils et des méthodes de support à la conception de sous-systèmes d'un UAM accomplissant spécifiquement des tâches de saisies d'objets. Afin d'accomplir cet objectif, trois sous-objectifs (**SO**) sont définis permettant de cadrer et d'orienter le travail.

SO1 : Identifier et classifier les tâches pouvant être accomplies par un UAM.

Avant même de développer des méthodes de support à la conception pour les UAM, il est important de bien définir et de catégoriser tous les types de tâches pouvant être effectuées par ceux-ci, incluant la saisie d'objet. Cette catégorisation peut servir de guide tout au long du processus de conception afin d'orienter les choix et de réduire la complexité des systèmes, et ce faisant, vient faciliter la conception intégrée [9]. Les sous-systèmes intervenant dans un UAM ainsi que leurs interactions doivent être identifiés pour guider le processus de conception.

Parmi les sous-systèmes intervenant lors d'une tâche de saisie d'objet, deux ont été choisis pour répondre à l'objectif principal et ainsi développer des méthodes de support à la conception. Le premier, essentiel au fonctionnement d'un UAM, est le correcteur interne assurant sa stabilité et son positionnement (**SO2**). Le second est une méthode permettant la prise de décision et la sélection d'un UAM optimal parmi une flotte afin d'accomplir une tâche de

saisie dans un environnement encombré (**SO3**).

SO2 : Développer une méthode de support à la conception d'une loi de commande interne pour un UAM effectuant une tâche de saisie d'objet statique.

Un manipulateur effectuant une tâche de saisie statique où l'objet n'est pas en mouvement nécessite une commande en position précis. La méthode permet alors de supporter la conception détaillée d'une telle loi de commande afin d'atteindre les performances dynamiques désirées. Cette méthode de support à la conception pourrait alors simplifier l'obtention d'une loi de commande lors d'éventuelles modifications aux paramètres physiques de l'UAM durant le processus de conception.

SO3 : Développer un cadre supportant la conception d'un sous-système de prise de décision pour la planification de multiples tâches de saisies accomplies par une flotte d'UAM.

Puisque les UAM ont des capacités d'interactions supérieures aux UAV, des algorithmes de guidage sont nécessaires et doivent permettant d'accomplir ces tâches. Afin de profiter des différentes capacités d'interactions liées à différentes configurations d'UAM, un sous-système permettant la sélection d'une configuration optimale au sein d'une flotte disponible afin d'accomplir une tâche de saisie dans un environnement précis serait intéressant afin de mieux répartir le travail complet de la flotte. Comme cette planification dépend de l'application désirée, une méthode vient supporter le travail de conception pour différentes applications. Si une modification de la flotte ou de l'application survenait lors de la conception du système les méthodes proposées seraient en mesure de facilement faire face à ces changements.

La combinaison de ces trois sous-objectifs permet donc d'accomplir l'objectif principal et d'introduire des méthodes de support à la conception d'UAM adaptées à des tâches de saisie d'objets. Bien évidemment, d'autres sous-systèmes sont aussi impliqués dans l'accomplissement de ces tâches, tels que la vision par ordinateur ou la localisation, mais n'ont pas été traités dans cette thèse. Il est important de mentionner que ce travail ne présente pas une méthodologie permettant la conception complète d'un UAM, ni n'automatise les étapes de conceptions d'un tel système. Les outils et les méthodes de support à la conception qui sont présentés peuvent être utilisés dans différents cadres et leur fonctionnement est démontré à l'aide d'exemples en simulation.

1.3 Contributions

Par l'intermédiaire de trois articles, cette thèse contribue au développement d'outils et de méthodes supportant la conception des sous-systèmes d'UAM identifiés par les sous-objectifs de recherche. Le chapitre 4 présente dans un article de revue, une méthode supportant la concep-

tion d'une loi de commande à gains séquencés pour UAM munis d'un bras robotique à 2DDL. L'utilisation de la synthèse $\mathcal{H}_\infty$ supporte le problème de conception de loi de commande en ne demandant qu'une définition des objectifs dynamiques et des paramètres modélisant le système à commander au concepteur. L'obtention de la loi de commande est obtenue par optimisation, facilitant ainsi son calcul lors d'itérations durant le processus de conception de l'UAM, notamment lors de modification des paramètres physiques du système.

Le chapitre 5 développe, aussi dans un article de revue, un support à la conception d'un sous-système de sélection optimal d'UAM parmi une flotte pour une tâche de saisie dans un environnement encombré. Ce travail contribue au développement d'un sous-système de guidage pour UAM, qui sont peu communs dans la littérature. La conception de ce sous-système est supportée par la possibilité d'utiliser les méthodes présentées dans tout contexte impliquant des tâches de saisies d'objets statiques. Le concepteur peut définir la flotte utilisée et entraîner le prédicteur introduit par apprentissage machine. La méthode présentée peut aussi être utilisée durant la conception de la configuration d'un manipulateur en fonction d'une ou plusieurs tâches à accomplir.

Alors que les méthodes développées aux chapitres 4 et 5 supportent la conception détaillée de sous-systèmes, le chapitre 3 présente une taxonomie des tâches pouvant être accomplie par les UAM, contenue dans un article de conférence. Cette catégorisation utilise des blocs élémentaires pour décrire des missions complexes. Elle permet de définir les tâches de manières simples en fonction des capacités requises par l'UAM. Elle permet aussi de cadrer la tâche sélectionnée pour les deux autres travaux de cette thèse, soit la saisie d'objets statiques.

Le chapitre 6 discute, dans un contexte de conception, des relations existantes entre les différentes méthodes et outils développés dans cette thèse. Les méthodes introduites sont indépendantes du processus de conception utilisé et n'interviennent pas directement dans celui-ci. Elles vont plutôt supporter certaines étapes du processus de conception en orientant les concepteurs lors de la phase conceptuelle, en facilitant leur conception détaillée et leurs modifications lors d'itérations de design. Ce faisant, plus de temps pourra être consacré par les concepteurs aux étapes d'intégrations favorisant l'obtention d'un produit optimisé.

1.4 Structure de la thèse

Le chapitre 2 présente une revue de littérature portant sur les méthodes de support à la conception de sous-systèmes d'UAM dans un contexte de processus de conception intégrée. Il inclut aussi des pistes de solutions pour les problématiques identifiées qui servent de base aux travaux présentés dans les chapitres suivants. Le chapitre 3 propose une taxonomie des tâches

pouvant être accomplies par un UAM en plus d'identifier les sous-systèmes présents dans un UAM. Par la suite, le chapitre 4 présente une méthode de support à la conception d'une loi de commande interne pour UAM s'appuyant sur la synthèse $\mathcal{H}_\infty$ structurée. Le chapitre 5 détaille une méthode de sélection d'UAM au sein d'une flotte pour l'accomplissement optimal de tâches de saisies d'objets dans des environnements encombrés. Le chapitre 6 contient une discussion sur l'utilisation de ces méthodes, les contributions de recherche ainsi que les implémentations effectuées. Finalement, le chapitre 7 conclut sur l'ensemble du travail et présente les directions futures.

CHAPITRE 2 REVUE DE LITTÉRATURE

Ce chapitre présente une revue de littérature en lien avec les objectifs de recherche. Puisque chaque chapitre comporte sa propre revue de littérature en introduction des articles, le but n'est pas de répéter ici la même information. Les travaux présentés dans cette revue le sont d'un point de vue de support à la conception d'UAM en fonction des tâches à accomplir. La section 2.1 présente les UAM à travers une revue de la littérature d'un point de vue de conception et présente le processus de conception associé à ce type de systèmes. Les avancées techniques connexes sont alors laissées à l'introduction de chacun des chapitres. La section 2.2 présente les avancées en classification des capacités d'interactions des UAM dans le but d'un éventuel outil de support à la conception. Les sections 2.3 et 2.4 présentent une revue des outils de support à la conception de systèmes de guidage et de commande respectivement. Elles possèdent chacune la même structure, où les problématiques associées sont présentées au début et sont suivies d'une introduction à une piste de solution.

2.1 Définition et conception d'UAM

2.1.1 Configuration des UAM

Un UAM est un système robotique sans équipage composé d'une base aérienne à laquelle est ajoutée des capacités de manipulation ou d'interaction avec l'environnement [6]. La plateforme aérienne prend généralement la forme d'un multicoptère, d'un hélicoptère, d'un avion ou d'un ballon gonflable en modèles réduits [6], regroupés sous le terme UAV. Dans le cadre d'un UAM, la base aérienne peut être de formats assez variables, allant de drones possédant une masse inférieure à un demi-kilogramme [17] à un octocoptère ayant près d'un mètre d'envergure [18]. Les systèmes servant à l'interaction avec l'environnement sont aussi de configurations variées. Certains UAM sont équipés de pinces simples situées sous la base aérienne, [19–21] ou encore de câbles servant à tirer sur les objets (*slung-load*) [22–24]. Cependant, l'utilisation de bras robotiques à plusieurs DDL tend à être favorisée grâce à leur capacité de manipulation élevée [15]. Quelques exemples de configurations utilisant ces divers outils de manipulations ou d'interactions sont illustrés à la figure 2.1. Ceux-ci posent cependant de plus grands défis de conception et d'implémentation [15], notamment au niveau de la stabilisation, en comparaison aux autres types de manipulateurs. Le tableau 2.1 effectue une comparaison qualitative entre les différentes catégories d'UAM. L'utilisation de bras robotiques est fortement associée aux tâches de saisie autonome d'objet grâce à la portée des bras

robotiques, et à leur capacité de compenser pour l'erreur en position du drone directement. Cette revue de littérature, ainsi que le reste de la thèse, concerne exclusivement les UAM conçus à partir d'une base de type multicoptère équipée d'un bras robotique sériel à plusieurs DDL.

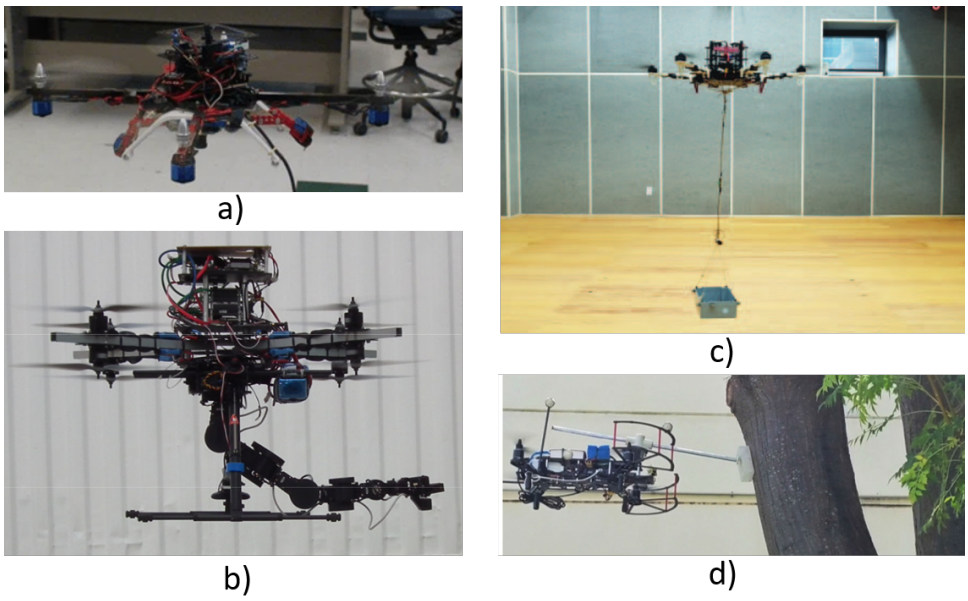

Figure 2.1 Différents types de manipulateurs. a) Pince simple. Tiré de [20]. b) Bras robotique à plusieurs DDL. Tiré de [18]. c) Charge suspendu par câble. Tiré de [24]. d) Tige pour application. Tiré de [12].

Les capacités d'interaction des UAM amènent le besoin de développer de nouvelles méthodes et outils pour des utilisations dans des contextes plus variés et répondant aux nouvelles problématiques spécifiques aux UAM en plus de celles affectant les UAV seuls. Un point fondamental à leur conception est la nécessité de stabiliser un système instable possédant, maintenant, un centre de gravité devenu dynamique à cause du bras robotique [5] par une loi

Tableau 2.1 Comparaison qualitative entre les différentes catégories communes de manipulateurs. Adapté de [15].

Type de manipulateur	Difficulté	Capacité de manipulation	Stabilité	Tendance en recherche
Pince	Faible	Faible	Élevé	Diminue
Cable	Moyenne	Moyenne	Moyenne	Diminue lentement
Bras robotique	Élevée	Élevée	Moyenne	Augmente rapidement

de commande adéquate. Au niveau de la conception mécanique, le choix du nombre de DDL du bras devient un choix de conception important. Ainsi, une configuration possédant un bras à 2DDL [25] peut permettre la saisie d'objet dans des conditions où ceux-ci sont facilement accessibles et sont dans des environnements peu encombrés tout en augmentant la charge utile (*payload*) possible grâce à un bras léger [26]. D'un point de vue mécanique, l'augmentation du nombre de DDL augmente le nombre d'actionneurs et de pièces de structure nécessaires ce qui diminue la charge utile disponible [12]. L'augmentation du nombre de DDL permet l'accomplissement de tâches nécessitant une capacité de mouvement du bras plus élevée que celle d'un bras planaire, comme l'ouverture de tiroir [27] ou l'inspection par contact [28]. À l'autre bout du spectre, des bras robotiques redondants à 7DDL permettent l'accomplissement de tâches dans des environnements encombrés [18]. La redondance des actionneurs par rapport à la pose de l'outil peut être utilisée pour minimiser l'effet du déplacement du bras sur la base [29]. Diverses configurations d'UAM possédant des bras robotiques sont illustrées à la figure 2.2. Il est évident que le choix de la configuration de l'UAM devient fortement lié à son utilisation désirée.

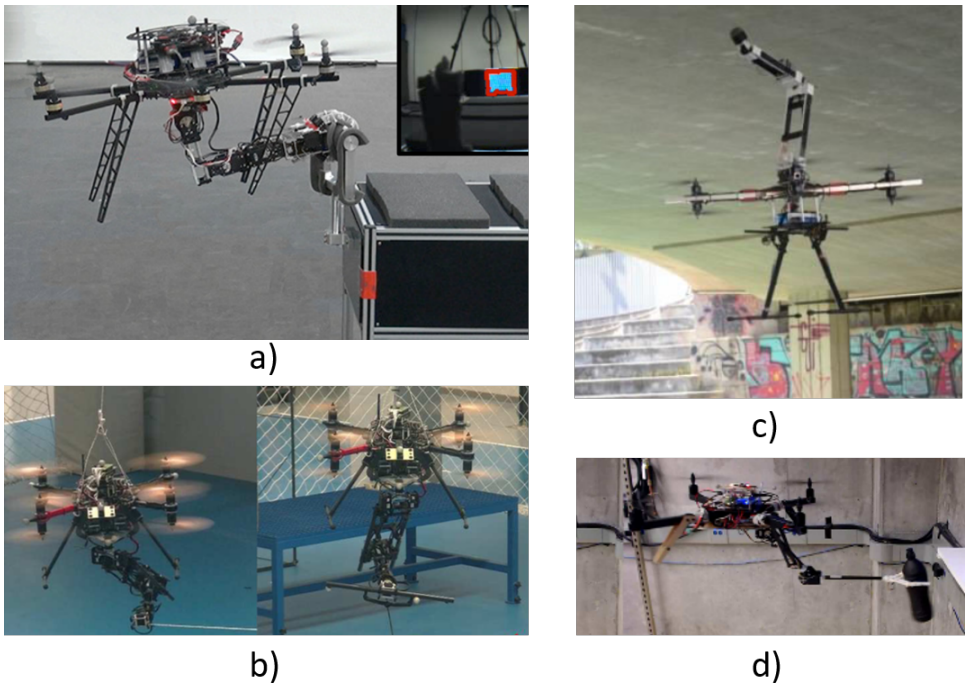

Figure 2.2 Multicoptères équipés de bras possédant des configurations différentes. a) Tiré de [27]. b) Tiré de [30]. c) Tiré de [28]. d) Tiré de [31].

2.1.2 Vue d'ensemble de la littérature concernant les UAM

Alors que les UAM sont de plus en plus employés dans l'accomplissement de tâches diverses [15], les méthodes de support à la conception pour UAM sont rares dans la littérature. Chaque laboratoire de recherche travaillant sur les UAM utilise une plateforme différente, et recommence les étapes de conception du début, notamment au niveau du développement de la structure, de l'implémentation d'un correcteur de vol et des différentes capacités de perception. Le tableau 2.2 présente les avancées et résultats des plus grands groupes de recherche dans le domaine travaillant spécifiquement sur les manipulateurs aériens équipés de bras robotiques à plusieurs DDL, selon le regroupement des groupes de recherche effectué par Ding et al. (2019) [15]. Les travaux publiés concernent majoritairement des stratégies de commande, avec ou sans vision, pour différentes configurations d'UAM. Très peu de travaux abordent d'autres aspects, tels que la planification de trajectoire, les capacités de décision ou le support à la conception. La plupart des laboratoires, et même parfois pour chaque projet à l'intérieur d'un même laboratoire, repartent d'une nouvelle plateforme pour étudier la dynamique et la commande. L'exploration des autres sous-systèmes nécessaires à un UAM autonome dans des contextes de tâches, tels que la navigation ou le guidage est alors peu présente dans la littérature, comme montré au tableau 2.2.

2.1.3 Processus de conception appliqué aux UAM et méthode de support à la conception

Les UAM sont des systèmes mécatroniques au sens où ils combinent la mécanique, l'électronique, la commande et le logiciel, afin d'obtenir un produit multidisciplinaire [7]. La conception d'UAM doit donc tenir compte des particularités de conceptions des systèmes mécatroniques, tels que les problèmes de communications dus à la multidisciplinarité, la complexité technique des systèmes, ainsi que la difficulté à obtenir une vue d'ensemble du système [53]. Le processus de conception des systèmes mécatroniques comporte typiquement les étapes de design conceptuel, détaillé et de production [54]. Il peut aussi inclure le design préliminaire avant le design détaillé. L'étape de design conceptuel permet la génération de plusieurs solutions potentielles, en incluant l'architecture du système [55], ainsi que la sélection d'un concept désigné répondant le mieux au cahier des charges. Le design préliminaire permet de sélectionner de grandes lignes directrices pour chacun des paramètres essentiels du système. L'étape de design détaillé sert alors à définir et concevoir chacun des composants du produit afin de satisfaire les requis et fonctions identifiés. Finalement, l'étape de production consiste en la sélection des procédés et méthodes utilisées pour la fabrication, s'il y a lieu.

Alors qu'habituellement ce processus est fait de manière séquentielle, c'est-à-dire en accom-

Tableau 2.2 Avancée et domaine d'application des travaux sur les UAM composés de multi-coptères équipés de bras robotiques

Configuration # de DDL	Avancée	Contrôle	Design	Vision	Autre
9	Design mécanique / commande en force [29]	■	■		
2×2	Cadre de travail pour ouverture de valve [32]	■			
-	Cadre de simulation [5]		■		
4	Commande en position par PID [33]	■			
2	Commande par prédiction de modèle [31]	■			
2×2	Commande en position pour ouverture de valve [34]	■			
4	Commande adaptative hybride [35]	■			
2×2	Classification des interactions/ IBVS/ modélisation du couplage [36]	■		■	■
n	Commande à couches multiples [37]	■			
3	Commande par impédance cartésienne [38, 39]	■			
5	IBVS relatif à la cible de saisie [40]	■		■	
5	IBVS coopératif [41]	■			
6	IBVS hybride avec la position [42]	■		■	
6	IBVS hybride avec la composition hiérarchique de la tâche [43]	■		■	■
5	IBVS hybride avec priorisation de la tâche [44]	■		■	■
5	Design mécanique et éléctrique [45]		■		
5	Commande par contact de force [28]	■			
2×5	Conception de bras inspiré par les bras humains / backstepping intégral / estimation de couple [46]	■	■		
7	Commande par backstepping [18]	■			
3	Commande par backstepping [47]	■			
6	Commande comportementale coordonnée [48]	■			
6	Commande par impédance avec module de cinématique inverse et de correction en position [30]	■			
3	Ouverture de tiroir pré-planifié [27]	■		■	■
2	Commande adaptative pour tâche de saisie [26]	■			■
2	Planification de trajectoire par RRT* et commande pour coopération [49]	■			■
2	Commande adaptative avec estimation de paramètre [25]	■			
2	IBVS avec commande adaptative [50]	■		■	
2	Planification de trajectoire par RRT* et primitive de mouvement [51]				■
2	Commande adaptatif glissant (sliding mode) avec estimateur de charge [52]	■			

plissant chaque étape pour chacun des domaines nécessaires au fonctionnement du système (la structure, la commande, l'électronique, etc.), la conception intégrée (*concurrent design*), où tous les domaines sont considérés simultanément lors des étapes de conception, permet d'obtenir un design optimisé [10]. Le modèle en V (*V-Model*) est un exemple de modèle choisi ici pour représenter la conception intégrée [56], et est illustré à la figure 2.3. À chacune des étapes de conceptions, on commence par considérer l'intégralité du système afin d'identifier les possibles problèmes d'intégrations et de conception, notamment dues à la multidisciplinarité. Par la suite, surviennent les étapes de conception spécifiques aux domaines, où chaque sous-système est conçu individuellement. Lors de cette étape, chacun des composants de l'UAM doit être analysé, modélisé et conçu par des experts dû à leur complexité technique. Par exemple, beaucoup d'efforts sont mis sur la commande de vol qui est essentielle à l'utilisation de l'UAM. Finalement, l'intégration permet d'intégrer les sous-systèmes conçus indépendamment. Le processus de conception possède habituellement plusieurs boucles de retour entre les différentes étapes (*redesign*) afin d'apporter les modifications requises à l'obtention d'un système optimisal.

Alors que le modèle en V est utilisé pour illustrer le processus de conception, les méthodes et outils développés dans cette thèse peuvent s'appliquer à toute méthodologie de conception choisie. Les outils des chapitres 4 et 5 viennent supporter la conception des sous-systèmes spécifiques d'un point de vue technique, en retirant le besoin d'avoir des experts des domaines. Ceci implique donc que la conception d'UAM utilise les sous-systèmes tels que présentés dans les chapitres mentionnés. Le chapitre 3 quant à lui présente une taxonomie qui supporte qualitativement la description des missions et tâches de l'UAM, mais n'agit pas sur le processus de conception lui-même.

Une solution potentielle est de simplifier la tâche de conception dans les étapes spécifiques au domaine, notamment au niveau du design détaillé, afin que l'équipe de conception puisse se concentrer sur l'intégration, ce qui devrait permettre l'élaboration d'un système optimisé dans son ensemble [10]. Le développement de certains sous-systèmes peut devenir contraignant si des modifications sont apportées durant le processus de conception, dû au temps requis à leur développement. Ceci a été identifié comme un défi potentiel lors de la conception de ce type de système [53]. Avoir des méthodes facilitant la conception de ces sous-systèmes plus complexes permettrait alors l'implémentation de cette solution.

Les besoins spécifiques de conception pour les robots autonomes comme les UAM font en sorte que d'éventuelles méthodes de support à la conception doivent considérer leur utilisation dans des environnements encombrés ou dynamiques [57], en plus des requis usuels associés aux robots comme la charge utile, l'autonomie et la manœuvrabilité. Ces outils doivent

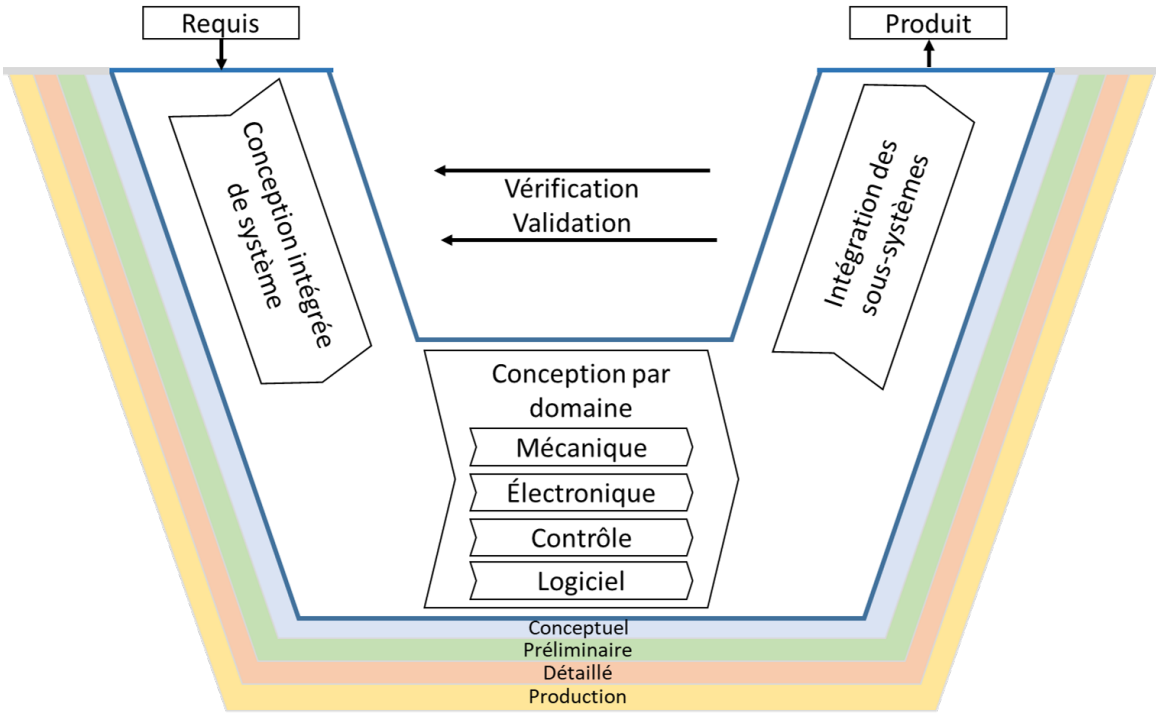

Figure 2.3 Processus de conception en V. Adapté de [56].

alors permettre la modélisation du monde physique. Dans le cas d'un UAM, ces environnements peuvent varier à cause de la variété de tâches pouvant être accomplies, à l'opposé, par exemple, d'un bras robotique dédié à une seule tâche en chaîne de montage [58]. Un autre besoin de conception est la nécessité de calculs précis pour les sous-systèmes essentiels, comme le correcteur, qui pourraient être supportés par des méthodes adaptées [56].

De toutes les manières possibles pour supporter la conception d'UAM, un outil de classification de tâches utile à l'étape de design conceptuel ainsi que les méthodes spécifiques au sous-système de commande et de guidage pour le design détaillé ont été choisis pour cette thèse. Ces méthodes seront appliquées spécifiquement dans le contexte de tâches de saisies d'objets.

2.2 Classification des capacités d'interaction comme outil de support à la conception

Les UAM permettent l'accomplissement de tâches et de missions très variées, dans divers contextes et environnements. Ils ont été utilisés dans des situations aussi diverses que la construction [59], le transport d'objets [26], l'ouverture de valves [34], l'inspection par

contact [28] et même l'écriture [60]. La plupart des missions complexes ont cependant une tâche commune, qui est aussi la tâche la plus étudiée dans la littérature, soit la saisie d'objets [18]. Elle est souvent employée comme étude de cas pour la vision par ordinateur, de la commande en position et de la conception mécanique des manipulateurs. De manière générale, les tâches sont habituellement choisies comme étude de cas, pour démontrer les diverses capacités introduites. Rarement, les aspects plus haut niveau des tâches comme la planification de la mission sont abordés, comme démontré dans le tableau 2.2.

L'accomplissement de tâches différentes nécessitera évidemment des capacités différentes. D'un point de vue de commande, la saisie et le transport de petits objets nécessiteront une correction en position précise de l'outil, robuste aux variations induites autant par la saisie que par les perturbations du multicoptère [26]. D'un autre côté, une tâche peut nécessiter une commande plus précise en force, par exemple pour appuyer sur un bouton pendant une période de temps [61]. Le même besoin s'applique pour les sous-systèmes haut-niveau tel que ceux de navigation et de guidage [3]. Les sous-systèmes d'un UAV sont généralement classifiés en trois catégories, le guidage, la navigation, et la commande (GNC- Guidance, Navigation and Control), comme illustré à la figure 2.4. Alors que la commande traite de la gestion des entrées du système pour obtenir un comportement dynamique désiré, la navigation quant à elle, s'occupe de mesurer, de suivre les états du système et de gérer le déplacement du système d'un point à l'autre [62]. Le guidage sert alors de pilote et d'intelligence et regroupe les algorithmes servant à la génération d'objectifs, de trajectoires et à la coordination pour un seul ou une équipe de UAM [63]. L'ensemble des systèmes GNC nécessaires pour l'accomplissement d'une tâche pourrait alors devenir un critère permettant une classification haut-niveau, c'est-à-dire que chaque tâche de cette classification nécessiterait un ensemble différent.

Des classifications ont déjà été développées dans d'autres contextes que les UAM. Les plus nombreuses, en demeurant dans le domaine de la robotique, sont les taxonomies pour les types de saisies adaptés à des mains humaines [64–67] et servent d'outils de support à la conception de systèmes [65]. Par exemple, en aidant à mieux comprendre les configurations de préhensions humaines, les roboticiens peuvent s'en servir comme guide et outil à la sélection et conception de systèmes de saisies, permettant de réduire la complexité des systèmes [65], et, de ce fait, facilitant le processus de conception [9]. Plus spécifiquement pour les UAM, leurs capacités à interagir de différentes manières avec leurs environnements augmentent la complexité de conception d'un système tentant d'effectuer ces tâches. C'est dans ce contexte qu'une taxonomie possédant un plus haut degré d'abstraction permettant de classifier les actions possibles devient utile [11]. Il devient alors un outil permettant de catégoriser les taches en fonctions des sous-systèmes nécessaires se généralisant aussi pour des missions

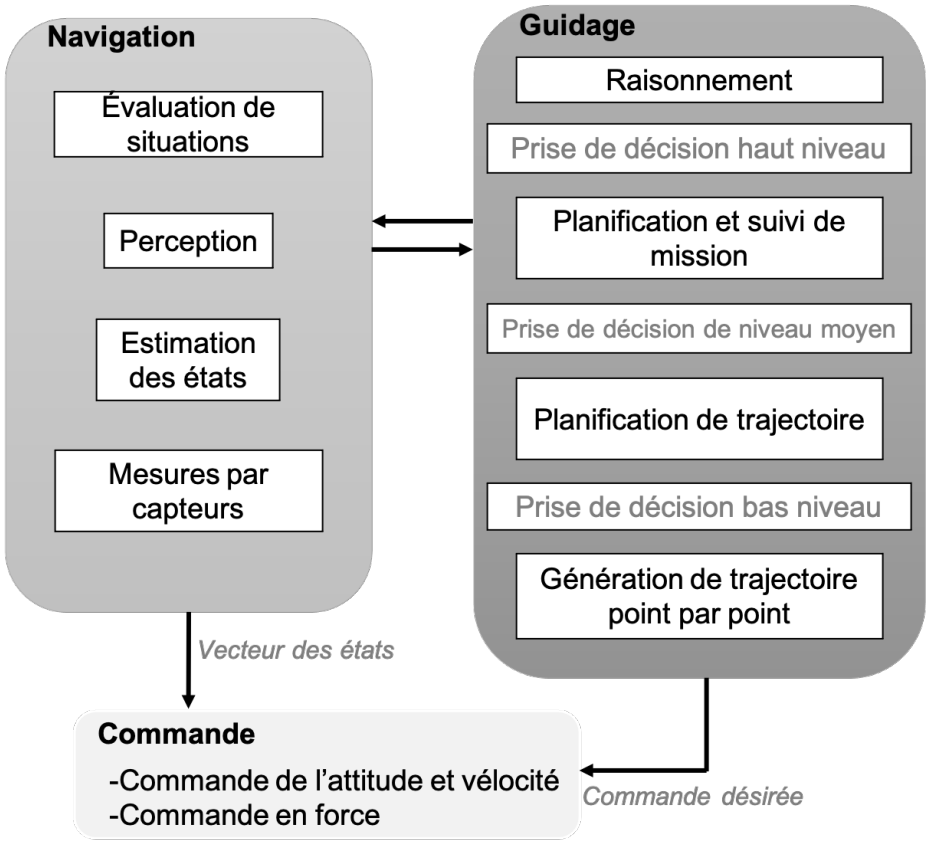

Figure 2.4 Relations entre guidage, navigation et commande. Adapté de [62]

regroupant de multiples tâches.

Certaines classifications ont déjà été accomplies pour les UAM, au niveau des différentes méthodes de commande en fonction de l'effort que l'UAM doit effectuer. Une première catégorisation définit trois types d'interactions, soit le couplage momentané, le couplage faible ou le couplage fort [36, 68]. Le premier type de couplage représente la saisie d'un objet, qui nécessite l'application d'une force momentanée et où la cible devient un poids supplémentaire statique par la suite. Le second représente l'application d'une force faible en vol par l'UAM, tandis que le dernier représente un couplage qui soutient le poids de l'UAM, comme une manœuvre de perchage. Une seconde classification des capacités des UAM regroupe les interactions en quatre classes, soit saisir, interagir, s'accorder et manipuler [15], sans critères d'application précis. Des critères plus spécifiques seraient nécessaires pour transformer une telle classification en véritable outil de support à la conception. Ces concepts servent à choisir les lois de commande à utiliser ainsi qu'à identifier les requis du manipulateur dans leur ensemble. De manière générale, une telle classification des tâches serait avantageuse autant durant le design conceptuel, permettant de simplifier la conception, que pour le design

détaillé, où elle pourrait être utilisée en planification de mission.

2.3 Support à la conception de sous-systèmes de guidage d'UAM associés aux tâches de saisies

Comme défini précédemment, le guidage peut être vu comme étant les capacités de pilotage autonome ou de raisonnement de l'UAM [63], et est nécessaire à l'accomplissement de tâches de manipulation avancée. La conception de ce type de sous-systèmes varie en fonction de la tâche et de l'utilisation à laquelle ils sont prévus [3]. Afin de faciliter leur conception lors de la conception détaillée, il faudrait proposer des méthodes de support à la conception flexibles et associés aux tâches pouvant être accomplies par des UAM.

Certains défis liés à la conception de sous-systèmes de guidage en général ont été identifiés [62]. Ces défis ont été cadrés pour la conception d'UAV, mais s'appliquent aussi aux UAM.

— Les systèmes de guidage nécessitent de l'information provenant de systèmes de perception, qui comportent eux-mêmes leurs problèmes et font encore l'objet de recherches actives
— L'implémentation embarquée de systèmes de guidage nécessite une puissance de calcul qui est souvent déjà utilisée par d'autres sous-systèmes plus essentiels, comme la perception, la navigation et la commande.
— Le manque de tests standardisés (*benchmark*) supportant le développement de nouvelles méthodes rend la comparaison et l'évaluation de tels sous-systèmes difficiles.

En réponse à ce dernier point, des mesures de références ont été proposées [69]. Celles-ci sont divisées en deux catégories : des tests pour les manipulateurs seuls, ainsi que des tests pour l'UAM dans son ensemble. Les tests pour le manipulateur seul proposent des mesures sur la précision de la trajectoire, du positionnement et de la répétabilité, de la capacité de charge utile, de la commande de force ainsi que de la détection de collision et le temps de réaction, chacun possédant différents critères proposés. Les mesures de performances concernant l'UAM reposent sur la mesure de différents critères lors d'une tâche simple de saisie d'objet, d'application de force de contact et de positionnement du multicoptère. Ce travail d'évaluation comparative standardisée [69] peut servir de support à la conception aux étapes conceptuelles et préliminaires, surtout d'un point de vue mécanique afin d'obtenir des lignes directrices sur les performances devant être atteintes dans différents contextes pour l'UAM.

Le nombre de publications concernant les systèmes de guidage haut-niveau pour UAM est faible, comme montré dans le tableau 2.2, et ce, dû notamment aux défis de conception

mentionnés. Malgré cela, le guidage est essentiel pour le développement d'UAM autonomes effectuant des missions et des tâches complexes. Les quelques sous-systèmes de guidage existants sont proposés pour des contextes très précis. La planification d'une mission dans le cas où une catastrophe amène le besoin d'une fermeture d'urgence de valves à l'aide d'un UAM a été présentée [14,32], où l'UAM doit détecter et identifier des valves selon leurs géométries et caractéristiques relativement faciles à reconnaître. Par la suite, des méthodes de commande se basant sur la configuration d'un UAM particulier sont utilisées pour accomplir la tâche. Un autre article propose aussi d'employer des UAM dans le cadre de missions de recherches et de sauvetages. De manière autonome, un UAM reconnaît des victimes humaines inconscientes et leur déploie un bracelet émetteur afin qu'elles puissent être suivies et retrouvées par les équipes d'urgences [16]. Ces travaux utilisent un seul UAM pour accomplir une tâche répétitive. Dans un contexte où les environnements et la description de la tâche changent de manière drastique, il peut devenir pertinent d'avoir une flotte d'UAM composée de plusieurs configurations différentes afin d'exploiter leurs forces respectives dans les différentes tâches et conditions d'utilisations.

Au niveau des sous-systèmes de guidage développés pour les UAV, ceux-ci servent surtout dans le cas où un UAV doit couvrir une grande surface dans le cadre d'une mission de recherche [70], ainsi que pour la planification et le suivi de trajectoire, en ayant des objectifs prédéfinis. Par exemple, la détection de marqueurs pré-calibrés combinée avec les équations de transformations de caméra servent à générer les cibles pour la planification de trajectoire [14]. Ce travail suppose l'accomplissement de tâches planifiées sur des objets déjà connus. Les autres systèmes de guidage proposés concernent la planification de trajectoires longues distances [3], dans divers environnements. Notamment, des capacités de guidage sont proposées dans le cas de missions de recherches autonomes par UAV lors de vol sans visuel, afin de générer leurs objectifs et par la suite obtenir des trajectoires [71]. Des UAV sont utilisés de manière collaborative afin d'accomplir des tâches de chargement et déchargement de caisses en contexte d'urgence. Un algorithme basé sur la gestion d'agents permet de gérer l'utilisation de cette flotte de manière optimale toute laissant une place à des décisions humaines [72]. Ce type d'algorithme suppose cependant que chaque UAV peut accomplir la même tâche simple de la même manière, ce qui n'est pas le cas dans une flotte d'UAM hétérogène. Ces sous-systèmes de guidage, bien qu'utiles dans la plupart des cas pour des missions autonomes avec un UAM, ne considèrent pas la planification de tâches en fonction de l'environnement, ni l'utilisation des différentes configurations d'UAM d'une flotte.

Plusieurs travaux ont été accomplis au niveau de la coordination de plusieurs UAV. Il est important ici de distinguer coordination et coopération où dans le premier, les UAV/UAM travaillent dans un même espace sans interaction entre eux (autre que d'éviter de se nuire),

tandis que dans le second, plusieurs unités peuvent travailler à accomplir la même tâche en s'entraidant [62]. Cette thèse n'aborde pas la coopération. La coordination de plusieurs UAV est surtout traitée comme un problème de commande optimale ou de commande distribuée. Par exemple, une équipe d'UAV accomplit un vol où un seul des UAV possède un plan de vol préétabli, et où le reste de l'équipe le suit à l'aide d'informations reçues de ce maître [73]. Une condition minimale pour l'observabilité de la position relative entre les agents robotiques a été démontrée et satisfaite à l'aide d'un ensemble de capteurs incluant un capteur de distance, un accéléromètre, un gyroscope et un magnétomètre [74]. Un algorithme permettant d'estimer la position relative à l'aide du même ensemble de capteurs a aussi été créé [75]. Des travaux ont aussi été effectués sur la planification de manœuvres d'évitement de collision dans la circonstance où deux UAV se retrouvent face à face et réussissent à se détecter visuellement. [76]. Finalement, un cadre décisionnel pour la coopération d'UAV avec des niveaux d'autonomie différents a été proposé [77]. Ces travaux, bien qu'adaptés à tout type de tâches, ne s'appliquent qu'aux déplacements et à la planification de trajectoires lors du vol. Cette étape est bien évidemment cruciale pour un UAM qui aurait une mission à accomplir dans un environnement large. Cependant, les capacités de manipulations accrues posent de nouveaux problèmes qui doivent être résolus à l'aide de méthodes de guidage adaptées aux tâches et contextes particuliers aux UAM. Les travaux de cette thèse s'intéressent plutôt à la commande de vol d'un UAM et la sélection et ne traitent pas de ces problématiques.

Une équipe d'UAM où chaque unité possède sa propre configuration nécessite une gestion haut-niveau pour l'optimisation de l'allocation des tâches et des missions. Dans le cas d'une flotte effectuant des tâches de saisie, un algorithme effectue une sélection optimale et autonome d'une configuration parmi la flotte disponible en fonction des conditions. Un contexte possible d'utilisation de ce sous-système pourrait être l'accomplissement de tâches de saisies notamment en entrepôt, soit en environnement connu, ou encore suite à des catastrophes, soit dans un environnement inconnu [4]. Plusieurs défis ont été identifiés afin d'obtenir des robots autonomes travaillant dans ces contextes, dont, notamment, l'obtention de méthodes de planifications de tâches haut-niveau [4]. Cependant, aucun travail portant sur les UAM n'emploie ce type de méthodes spécifiquement pour la sélection d'UAM optimal dans une flotte pour une tâche dans un environnement spécifique.

Quelques méthodes d'aide à la sélection ou de support à la décision pour la sélection de robots ont été développées dans le cas de manipulateurs fixes, notamment, la sélection d'un manipulateur parmi les différents modèles disponibles sur le marché. Une méthode permet d'évaluer un ensemble de produits en fonction des performances sur dix-huit critères comme la charge utile, la portée verticale, la masse, etc. Un processus analytique hiérarchique utilise ensuite des valeurs floues (*fuzzy*) pour produire l'évaluation à l'aide de descripteurs qualitatifs et

ensuite classer les choix [78]. Une autre méthode applique un processus similaire où l'évaluation repose sur un classement de l'importance des critères subjectifs et objectifs effectué par des experts [79]. Ces systèmes de décisions nécessitent une évaluation de plusieurs critères, souvent subjectifs, et permettent de comparer les performances pour une seule tâche [80]. Si la mission du manipulateur venait à changer, alors le processus doit être repris au début. Ces méthodes permettent un choix subjectif se basant sur les préférences et compromis des concepteurs, mais sont mal adaptées à une évaluation rapide pendant des missions avec des contextes et des tâches variés.

Un support à la conception de sous-systèmes de guidage permettant la sélection d'un UAM optimal au sein une flotte, en fonction de la scène et de la tâche, faciliterait la conception indépendamment de l'utilisation de la flotte. Actuellement, les méthodes de guidage pour UAM ne traitent pas de la planification de missions, sauf dans des cas très spécifiques comme l'ouverture de valves où la mission est assez constante à chaque répétition. Ce manque de travaux publiés provient de la spécificité de ces méthodes à leurs usages et des nombreux préalables nécessaire à leur développement (vision par ordinateur, navigation, cartographie et localisation, etc.). En plus de ces défis, les algorithmes de guidage doivent être légers d'un point de vue logiciel, afin d'être embarqué sur un micro-ordinateur déjà utilisé pour la navigation et la commande qui sont essentielles au fonctionnement de l'UAM [62]. L'apprentissage machine est alors une avenue potentielle pour le développement de méthodes de guidage [4], qui favorise notamment le développement d'algorithmes plutôt léger en taille nécessaire sur le système embarqué. Un cadre présentant une architecture de réseau et supportant sa conception par une méthode bien définie pourrait alors rendre supporter sa conception pour des contextes variés.

2.3.1 Apprentissage profond et guidage

L'apprentissage machine et plus spécifiquement l'apprentissage profond sont actuellement au cœur du développement d'une variété de sous-systèmes utilisés par les UAM et par les robots en général [4]. L'idée d'un algorithme servant d'intelligence est particulièrement appréciée de la communauté robotique et est actuellement étudiée dans plusieurs aspects comme la commande, la vision par ordinateur, la fusion de capteurs, la manipulation avancée, ainsi que la planification haut-niveau [4]. Cependant, très peu de travaux abordent la planification haut-niveau et, au mieux de nos connaissances, aucun ne l'aborde pour les UAM.

L'apprentissage machine (ML - *Machine Learning*) est un domaine issu de l'optimisation, des sciences informatiques et des sciences des données qui s'englobe dans le concept général d'intelligence artificielle. La base du ML consiste en l'entraînement d'un réseau afin qu'il

puisse prendre des décisions de manière autonome à partir des entrées fournies [81]. L'entraînement est alors conduit sous forme d'optimisation numérique. Le fonctionnement du ML repose sur un principe fondamental soit la qualité et la quantité des données utilisées lors de l'entraînement de l'algorithme [81]. Ces données doivent bien représenter les conditions d'utilisations réelles du réseau et être en nombre suffisamment élevé pour couvrir différents exemples et réussir l'entraînement. Il représente un des principaux problèmes à l'obtention d'algorithmes bien entraînés [81].

Les algorithmes de ML sont habituellement séparés entre les algorithmes supervisés et les algorithmes non-supervisés. Les algorithmes supervisés permettent d'associer chaque élément en entrée à une étiquette, soit la classe que l'algorithme doit apprendre à prédire dans le cas d'une classification, ou à une valeur numérique, dans le cas d'une régression. Les algorithmes non-supervisés, quant à eux, travaillent sans étiquette et vont plutôt chercher des groupements dans les données ou encore essayer d'extraire des caractéristiques (*features*) importantes [82]. Il est aussi possible d'effectuer de l'apprentissage semi-supervisé où la base de données contient une portion d'éléments avec étiquette et une autre sans étiquette. Ces types d'algorithmes deviennent pertinents considérant que les bases de données non-étiquetées sont nombreuses et que l'étiquetage est long et coûteux à produire [83].

Dans le cas d'algorithmes supervisés, l'entraînement de l'algorithme consiste à optimiser les paramètres intrinsèques du modèle mathématique choisi afin qu'il puisse prédire la bonne étiquette associée à une entrée donnée. Pour ce faire, une fonction de coût représentant la capacité du modèle à choisir la bonne étiquette est optimisée en utilisant une bonne part de la base de données, appelée l'ensemble d'entraînement. Une autre section de la base de données est conservée et non employée dans l'entraînement pour valider que le modèle est bien capable de généraliser son apprentissage sur des exemples n'ayant pas servi à l'entraînement, appelés l'ensemble de tests [81].

Une sous-catégorie du ML permettant de traiter des problèmes plus complexes est l'apprentissage profond (DL - *Deep Learning*). Le DL utilise des architectures d'algorithmes plus larges afin d'éviter un des défis majeurs avec le ML classique soit la sélection des caractéristiques adéquates à partir des données brutes. Une structure de réseau de neuronnes de DL contient plusieurs niveaux hiérarchiques pour apprendre des features simples un après les autres, afin de se construire une représentation d'une entrée brute. Ceci vient cependant avec quelques désavantages ; ils sont notamment plus difficiles à entraîner et nécessitent habituellement une plus grande base de données que pour des réseaux de ML classiques [81]. Les algorithmes de DL ont permis des avancées extrêmement rapides, notamment dans la détection sur images par des réseaux de neurones convolutionnels [84]. Ces réseaux et le DL en général se sont de-

puis propagés à de nombreuses applications utiles à la robotique, incluant l'apprentissage de dynamiques complexes, de lois de commande, de manipulations avancées, de reconnaissance d'objets avancée, d'interaction avec les humains et de fusion de capteurs. Une des avenues peu exploitées et favorables à l'utilisation de DL est la planification haut-niveau de tâches [4]. À l'aide d'une base de données d'exemples de tâches de saisies, incluant la description de la tâche et de l'environnement, et où chaque élément est identifié par la configuration optimale de l'UAM, il serait possible d'apprendre à choisir cette configuration optimale pour une tâche. La conception de cet algorithme de sélection serait alors supportée en proposant un algorithme de DL où un concepteur pourrait simplement modifier les étiquettes associées à la base de données et entraîner la structure de DL proposée. Il reste cependant à trouver une manière de représenter la tâche et l'environnement afin de l'utiliser dans un réseau de DL.

Afin de comprendre la suite, il est important d'illustrer le fonctionnement de base d'un réseau de neurones de DL. Nous présentons ici est un réseau dit pleinement connecté, aussi appelé perceptron à plusieurs niveaux (MLP - *MultiLayer Perceptron*), qui peut servir par exemple comme classificateur. La figure 2.5 présente la structure d'un tel type de réseau, où les tailles de l'entrée et des différentes couches peuvent varier selon les entrées utilisées et le nombre de neurones voulus dans le réseau. La première couche cachée, identifiée par le vecteur $\mathbf{h}^1$ est obtenue à l'aide d'une combinaison linéaire de l'entrée, représentée par le vecteur $\mathbf{x}$ suivi d'une fonction d'activation non-linéaire, nommée $F()$. La même transformation est alors répétée pour toutes les couches cachées du réseau où la taille des différentes matrices $\mathbf{W}$ et vecteurs $\mathbf{b}$ sont adaptés selon le nombre de neuronnes par couche. On obtient alors la sortie par combinaison linéaire de la dernière couche cachée, qui peut être soit une valeur continue dans le cas d'une régression, ou la prédiction d'une classe dans le cas d'un classificateur. L'équation 2.1 illustre ce réseau de neuronnes où $\mathbf{W}$ et $\mathbf{b}$ sont respectivement les poids et les biais, soit les variables à optimiser durant l'étape d'entraînement de l'algorithme. Ces variables sont optimisées afin d'extraire l'information nécessaire de l'entrée et des différentes couches subséquentes afin d'effectuer la prédiction ou la régression demandée.

$$\begin{aligned} \mathbf{h}^1 &= F(\mathbf{W}_1^\top \mathbf{x} + \mathbf{b}_1) \\ \mathbf{h}^2 &= F(\mathbf{W}_2^\top \mathbf{h}^1 + \mathbf{b}_2) \\ \mathbf{y} &= F(\mathbf{W}_3^\top \mathbf{h}^2 + \mathbf{b}_3) \end{aligned} \tag{2.1}$$

Le DL permet de manière générale d'entraîner un algorithme à identifier des motifs pouvant se répéter dans des données afin d'en obtenir, soit une classification, une identification ou

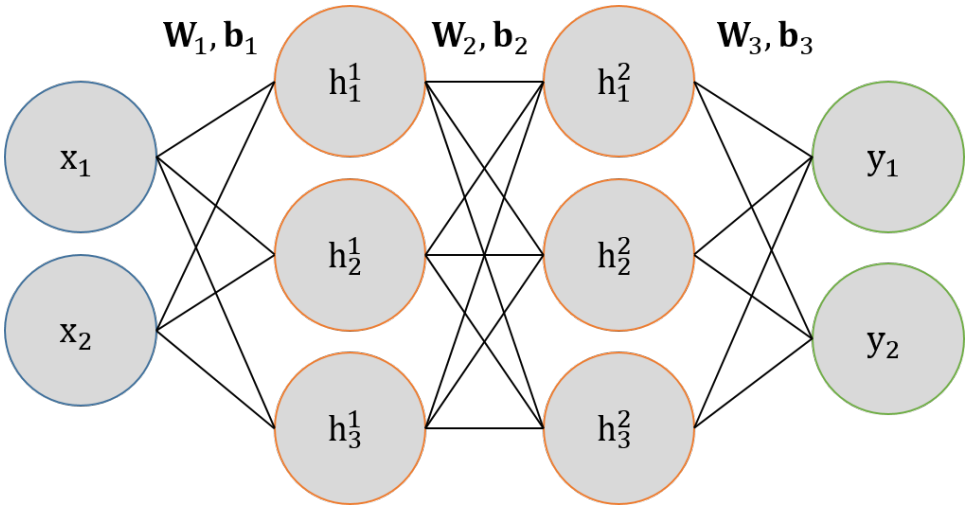

Figure 2.5 Structure de base d'un réseau de type MLP

autre sortie désirée. Il est possible de le faire pour des tâches dans lesquelles l'humain est excellent, comme reconnaître un chat dans une photo, ou encore dans des cas où l'humain aurait de la difficulté à identifier des motifs. Il peut supporter tous types d'entrée et de sortie, à condition d'identifier une structure les supportant.

2.3.2 Apprentissage profond avec nuages de points

Une des applications possibles du DL est la planification de tâches en fonction de la scène dans laquelle celles-ci se déroulent. Il faut donc obtenir une représentation de la scène environnante. Plusieurs représentations de ces environnements tridimensionnels sont possibles, avec des architectures de DL associés comme des photos [85–87], des représentations volumétriques (à base de voxel) [88, 89] et des nuages de points [90, 91]. Plusieurs robots commencent à être équipés de capteurs tels que des LIDAR ou des caméras de profondeurs, permettant d'obtenir des nuages de points de leur environnement [90], ce qui en fait des bons candidats, même si les utiliser avec le ML n'est pas trivial. De plus, la structure des données (une simple liste de coordonnées 3D) est un choix intéressant comparativement aux autres représentations qui nécessitent plus de mémoire, souvent limitée, dans des systèmes embarqués. L'utilisation de nuages de points pour le DL pose cependant quelques difficultés [90], nécessitant une architecture spécifique pour y répondre :

1. Ordre des points : Contrairement à une photo où chaque pixel possède une position importante par rapport aux autres, l'ordre des points dans un nuage de points est complètement arbitraire. Une simple convolution donnerait différents résultats selon

l'ordre des points. L'architecture doit donc pouvoir être invariante à ces permutations.

2. Interaction entre les points : Les points représentent une position dans un espace mesurable. C'est donc dire que des sous-ensembles de points peuvent correspondre à des structures locales qui doivent être détectées.
3. Invariance à la transformation : Les nuages de points doivent être invariants à la rotation et la translation dans l'espace.

PointNet [90] est un pionnier dans l'utilisation directe de nuages de points avec des algorithmes de DL. Il a été employé dans le cadre de reconnaissance d'objets à partir de leur modèle 3D sous forme de nuages de points, ainsi que dans le cadre de segmentation, où les différentes parties formant une scène doivent être identifiées. Sa capacité à tirer de l'information locale directement à partir de nuages de points en fait une excellente option pour des algorithmes devant analyser une représentation de scène. Afin de résoudre les problématiques identifiées aux nuages de points, PointNet propose les solutions suivantes pour chacun d'entre eux :

1. Ordre des points : Le réseau utilise une fonction symétrique sur les points. De manière pratique, ceci revient à appliquer le même réseau pleinement connecté sur chaque point et de combiner les sorties de ce réseau par une fonction de *maximum pooling*.
2. Interaction entre les points : Le réseau PointNet est sous-divisé en deux sections. La première permet l'apprentissage de caractéristiques sur le nuage de points par réseaux de neuronnes partagés pleinement connectés sur chaque point. La seconde section permet d'apprendre les interactions entre ces caractéristiques et la sortie en classification désirée, à l'aide d'un réseau pleinement connecté standard à plusieurs couches.
3. Invariance à la transformation : Dans la section s'occupant du nuage de points, une transformation linéaire sous forme de matrice de rotation est incluse dans les paramètres à apprendre du réseau. Ainsi le réseau peut devenir invariant aux transformations.

Afin de générer les caractéristiques trouvées dans le nuage de points et d'être invariant à l'ordre des points, PointNet utilise des MLP partagés. Ces réseaux fonctionnent comme les MLP présentés à la figure 2.5, sauf qu'au lieu d'avoir le nuage de points au complet en entrée, le réseau de neuronnes utilise les coordonnées x, y, z d'un point du nuage, et réutilise le même réseau avec les mêmes paramètres sur chacun des points. Cette structure est illustrée à la figure 2.6. Dans ce cas, il est possible d'obtenir une nouvelle représentation du nuage de points qui représente uniquement ses caractéristiques géométriques, et non plus une liste de

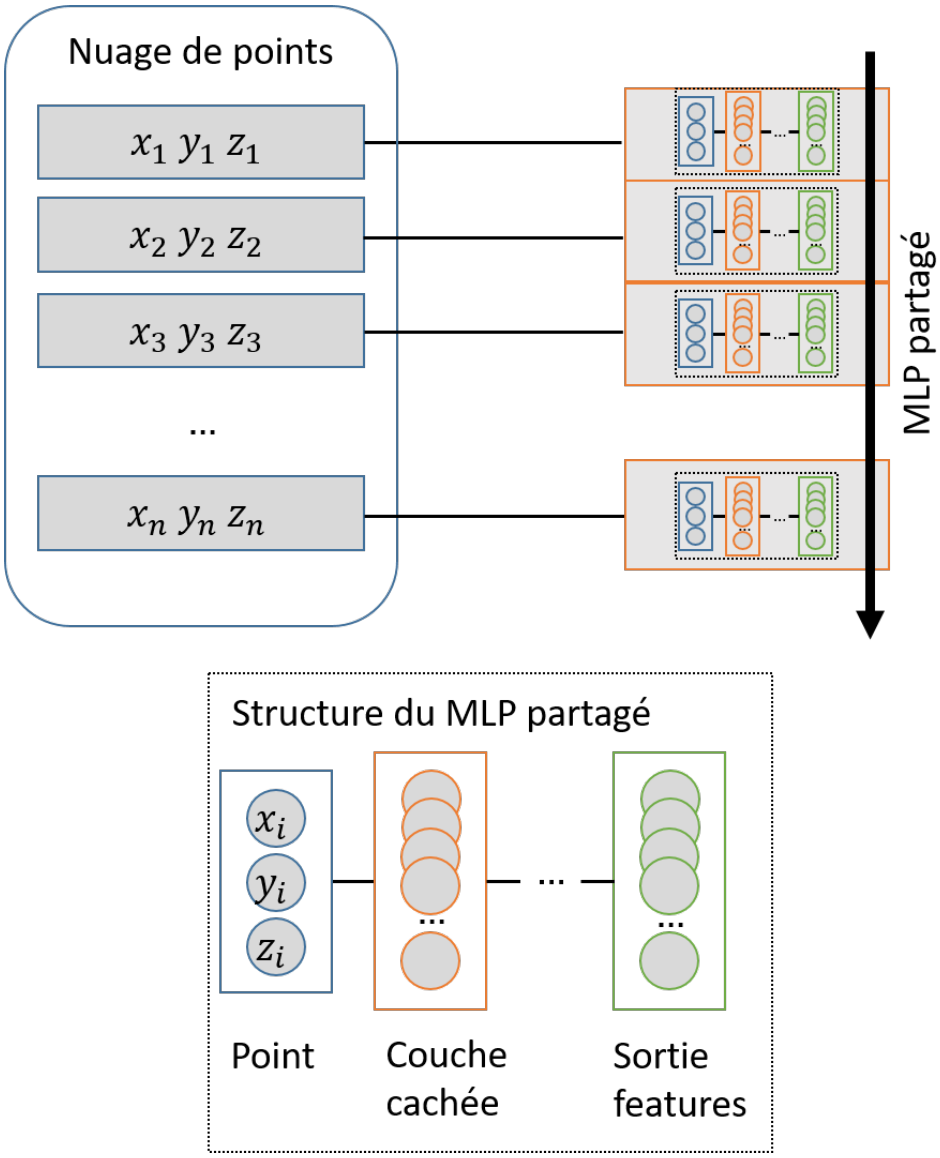

Figure 2.6 MLP partagé sur un nuage de points

points [90]. Une fois ces représentations obtenues pour chaque point, elles sont agrégées avec une fonction de mise en commun selon les scores maximums. Ceci fait que PointNet apprend à représenter un nuage de points à partir d'un sous-ensemble de points clés [90]. Par la suite, avec cette représentation, une seconde partie peut définir un classificateur, ou encore tout algorithme supervisé qui peut faire usage de cette représentation du nuage de points, comme montré à la figure 2.7.

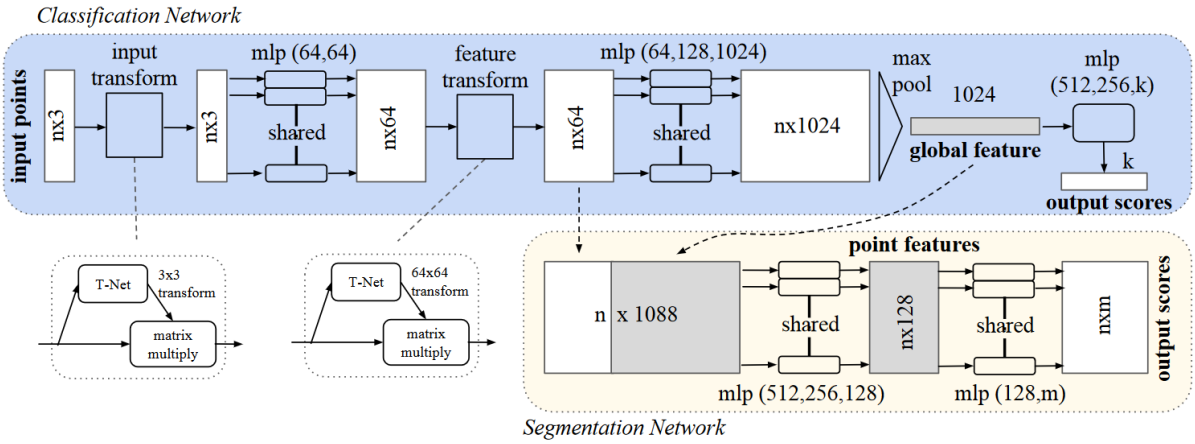

Figure 2.7 Structure du réseau de neurones PointNet. Tiré de [90]

2.4 Support à la conception de lois de commande d'UAM

Afin d'accomplir la tâche de saisie planifiée par les différents algorithmes de guidage, l'UAM doit pouvoir se déplacer dans l'espace de manière à se positionner correctement par rapport à la cible, selon les critères définis pour la tâche tout en minimisant l'erreur de positionnement du manipulateur. Les algorithmes de planification de trajectoires génèrent alors des commandes désirées pour les différentes variables contrôlables. Le correcteur permet d'accomplir ces commandes désirées en agissant sur les entrées du système. Il doit donc permettre de stabiliser le système, minimiser l'effet des perturbations en obtenant des performances dynamiques désirées, tout en respectant les difficultés particulières reliées à la commande d'un UAM par rapport à un multicoptère seul. Les différences identifiées [18] sont les suivantes :

— Le déplacement du centre de masse dû aux mouvements du bras robotique.
— La variation de la distribution de la masse qui amène des moments d'inertie variables.
— Les forces et les moments de réactions dynamiques appliqués sur l'UAV et provenant du mouvement du bras robotique

La littérature propose plusieurs lois de commande spécifiquement développées pour les UAM, mais très peu de méthodes supportant leur conception existent, et aucune n'a été appliquée aux UAM. De plus, des méthodes intégrées de support à la conception de lois de commande et de la structure de systèmes mécatroniques existent aussi, mais n'ont pas été appliquées aux UAM non plus. Par exemple, le design pour la commande (DFC - *Design for Control*) [92] permet de concevoir autant le correcteur que le design mécanique dans un même processus incrémental. Cette méthode se base sur une modélisation complète, mais simplifiée du système

dynamique contrôlé [93]. Elle a été appliquée à un quadricoptère asservi par la vision et équipé de PID (loi de commande proportionnelle, dérivée, intégrale) simples où l'objectif d'optimisation était un agrégat de performances dynamique et de vision, incluant l'erreur sur le positionnement et la vitesse [92]. Une méthode avec des objectifs et des applications similaires utilise des critères flous (*fuzzy criteria*) et une optimisation par essaims particulaires (PSO - *Particle Swarm Optimisation*) [94]. Dans ce dernier travail, des critères d'évaluation flous sont agrégés en un index de performance et par la suite optimisés selon les paramètres à l'aide d'un PSO. Dû au grand nombre de paramètres et de gains à traiter dans le cas de lois de commande plus complexes, comme celles nécessitées par des UAM, la méthode serait moins adaptée [92]. Aussi, les choix de métriques de performances peuvent varier d'un concepteur à l'autre. Le nombre d'itérations lors du processus de design peut aussi contraindre l'utilisation de ces méthodes.

Dans un cas où la conception de la structure et de la loi de commande est possible, il serait en effet plus optimal de l'effectuer simultanément. [95]. Ceci est cependant moins adapté à la conception d'UAM. Son utilisation pour des tâches variées pourrait nécessiter différentes stratégies de commande. De plus, pour la conception d'UAM dans les travaux identifiés au tableau 2.2, la plateforme est souvent un multicoptère commercial, c'est-à-dire que le multicoptère existe déjà et un bras robotique est ajusté selon les besoins. Le correcteur doit donc être conçu suivant ces contraintes de conception ainsi que celles associées aux tâches à accomplir. Par exemple, dans le cas spécifique d'une tâche de saisie d'objets, une commande en position précis est nécessaire [26]. De plus, à cause des difficultés inhérentes à la stabilisation d'UAM, une méthode permettant de supporter la conception de la loi de commande permettrait de simplifier cette étape du design détaillé.

Les stratégies de commandes modernes appliquées aux UAM ont été regroupées en deux catégories, en fonction de la méthode utilisée pour la modélisation et la conception du correcteur, soit les méthodes indépendantes et les méthodes unifiées [15]. Tout d'abord, les méthodes unifiées considèrent l'UAM comme étant un système unique où l'on commande toute entrée du système pour obtenir un contrôle de l'outil. Ces méthodes sont utilisées notamment dans le cas de commande en force comme la commande par impédance où l'on veut réguler les interactions avec les forces externes [38]. La commande adaptative est aussi appliquée afin de gérer l'ajout d'une masse inconnue à l'outil à l'aide d'une estimation des paramètres [25]. La complexité de modélisation du système entier amène des lois de commande adaptatives à mode glissant (*adaptive sliding mode*) [26] ou par impédance variable [96] afin de prévenir ces erreurs de modélisation, dans des tâches de saisie et de contact respectivement. Finalement, la méthodologie par approche récursive (*backstepping*) est possible grâce à la strucure des équations dynamique de l'UAM [97] et en garantit sa stabilité [18]. Ceci en fait donc une

méthode particulièrement bien adaptée, offrant une structure de loi de commande applicable à tout UAM.

D'un autre côté, les méthodes de modélisations dites indépendantes considèrent le multicoptère et le bras robotique comme deux systèmes distincts. Des stratégies par *backstepping* autant pour le multicoptère uniquement [47] que pour l'outil attaché sur le bras robotique [98] ont été employés. Finalement, la modélisation indépendante étant plus simple, elle permet l'utilisation de correcteurs comme les PID à plusieurs couches [37].

Tous ces correcteurs sont conçus à l'aide d'outil de simulation comme MATLAB/Simulink [99], par des experts de systèmes de commande, basés sur des concepts nécessitant un niveau de connaissance mathématique très spécialisé et rendant leur analyse complexe [13]. Leur application aux UAM est souvent considérée comme une étude de cas pour le développement de ces nouvelles lois de commande, sans égard à l'implémentation et au support à d'éventuels concepteurs. Dans ce cas, il serait intéressant d'avoir des outils permettant d'assister leur conception, en proposant une structure permettant de stabiliser le système et en offrant la possibilité d'automatiser une grande partie de la tâche de design en fonction des requis de conception.

2.4.1 Synthèse $\mathcal{H}_\infty$ structurée

La synthèse $\mathcal{H}_\infty$ structurée [100, 101] a été développée comme méthodologie pour la synthèse de loi de commande à structures fixes. Cette méthode permet au concepteur d'implémenter une structure prédéfinie ainsi que des requis de conception simplement formulés, tels que les temps de réponse, les marges de gain et de phase ou l'erreur en régime permanent, pour ensuite obtenir les gains de la loi de commande par optimisation non lisse. Cette méthode est limitée aux correcteurs linéaires [100] avec la possibilité de l'utiliser dans un cadre multi-modèle et donc avec des gains séquencés (*gain-scheduled*) [100]. Ceci peut être le cas pour un UAM, où les efforts appliqués sur le multicoptère sont influencés par les mouvements du bras robotique, rendant la stabilisation difficile du fait de ces interactions. Cette synthèse a déjà été employée pour des lois de commande d'avions militaires [102] où le système physique et les gains séquencés sont conçus simultanément. Si une structure permettant la stabilisation du système est connue, le processus de conception en est extrêmement simplifié. Il ne s'agit alors que d'identifier les requis dynamiques et les paramètres physiques du système à intégrer dans l'optimisation.

La méthode de synthèse de correcteur repose sur la définition d'un problème d'optimisation $\mathcal{H}_\infty$. On définit un système dynamique linéaire commandé comme à l'équation 2.2 et illustré à la figure 2.8. L'objectif de l'optimisation est d'obtenir un correcteur $\mathbf{K}(s)$ per-

mettant de minimiser la norme $\mathcal{H}_\infty$ du transfert en boucle fermée entre $\mathbf{w}$ et $\mathbf{z}$, identifié comme $\|\mathbf{T}_{\mathbf{w}->\mathbf{z}}(\mathbf{K})\|_\infty$, soit respectivement les entrées exogènes et les signaux régulés. La norme $\mathcal{H}_\infty$ est représentée par l'équation 2.3 et représente la valeur singulière maximale de ce transfert [103]. Dans le cas d'un système à entrée et sortie unique (SISO *Single Input Single Output*), ceci représente, à toute fin pratique, l'amplification maximale sur la plage de fréquences [101]. Dès lors, la synthèse est définie comme une tentative de minimiser cette norme tout en garantissant la stabilité interne du système [104].

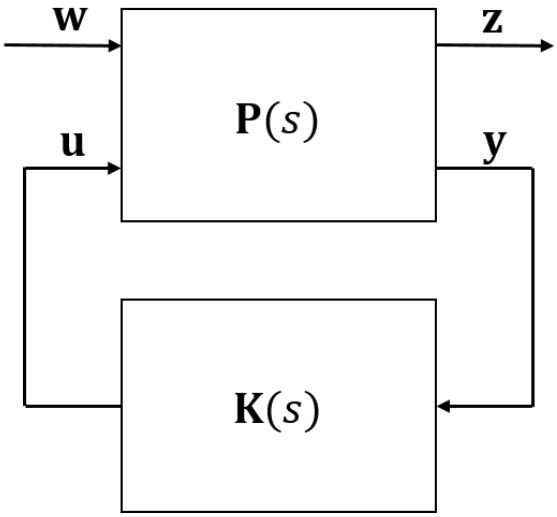

Figure 2.8 Modèle standard d'un système avec correcteur pour synthèse $\mathcal{H}_\infty$

$$\mathbf{P}(s): \begin{bmatrix} \dot{\mathbf{x}} \\ \mathbf{z} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{B_w} & \mathbf{B_u} \\ \mathbf{C} & \mathbf{D_w} & \mathbf{D_u} \\ \mathbf{E} & \mathbf{F_w} & \mathbf{F_u} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{w} \\ \mathbf{u} \end{bmatrix} \tag{2.2}$$

$$\|G(s)\|_\infty = \sup_{\omega \in \mathbb{R}^+} \bar{\sigma}\left[G(j\omega)\right] \tag{2.3}$$

$$\begin{aligned} \min \quad & \|\mathbf{T}_{\mathbf{w}->\mathbf{z}}(\mathbf{K})\|_\infty \\ \text{tel que} \quad & \mathbf{K}(s) \text{ stabilise } \mathbf{P}(s) \\ & \mathbf{K}(s) \in \kappa \end{aligned} \tag{2.4}$$

L'optimisation définie précédemment génère des correcteurs satisfaisant les contraintes, mais difficilement implémentables sur un système réel. Leur ordre élevé (égal à l'ordre du système

$\mathbf{P}(s)$) peut être notamment la principale limitation. La synthèse $\mathcal{H}_\infty$ structurée permet donc d'utiliser une structure préétablie en ajoutant cette contrainte au processus d'optimisation (κ) comme à l'équation 2.4 [104]. Au niveau du support à la conception, ceci implique que si une structure est capable de stabiliser et atteindre les performances voulues pour une configuration particulière d'UAM, alors le processus de conception de loi de commande pour la même configuration avec différentes valeurs de paramètres physiques, revient simplement à une optimisation numérique avec la structure préétablie.

La synthèse $\mathcal{H}_\infty$ structurée permet aussi d'optimiser les gains de manière automatique en transformant les requis dynamiques en fonctions compatibles avec l'optimisation [101]. Ceci est fait de manière transparente pour le concepteur qui n'a besoin que de saisir les requis. La transformation en critère $\mathcal{H}_\infty$ est automatique. Plusieurs requis peuvent être utilisés simultanément en les catégorisant selon des contraintes fortes ou faibles. Ceci amène un niveau d'abstraction accessible au concepteur qui n'a pas besoin d'être un expert en commande robuste pour obtenir un correcteur répondant à ses besoins.

Il est possible d'utiliser la synthèse $\mathcal{H}_\infty$ structurée de manière multi-modale, c'est-à-dire en réalisant la synthèse d'une loi de commande sur plusieurs modèles dynamiques simultanément. Le correcteur est donc conçu en effectuant un ajustement de courbes entre les modèles afin d'obtenir des gains séquencés selon des variables choisies [100].

La synthèse $\mathcal{H}_\infty$ structurée offre donc la possibilité de concevoir un correcteur par optimisation, en choisissant au préalable la structure de celui-ci et en ne nécessitant qu'une modélisation du système et des requis de conception. Cette méthode n'a pas encore été appliquée aux UAM où l'on peut séquencer les gains en fonction des mouvements du bras afin de compenser pour leurs effets sur la dynamique de vol.

2.5 Résumé des problématiques

Les UAM sont des systèmes mécatroniques qui posent des défis importants dès les premières étapes de conceptions, dû aux expertises variées requises, à leurs complexités techniques, et au processus de conception intégrée nécessaire à l'obtention d'un produit proche de l'optimum. Supporter la conception de certains sous-systèmes spécifiques d'un UAM durant les différentes étapes, permettrait alors aux concepteurs de réduire le temps consacré à ceux-ci, au profit d'une meilleure conception intégrée. Ces outils viendraient supporter le travail de concepteurs d'UAM dans des conditions d'utilisations précises. Une classification des tâches pouvant être accomplies par un UAM supporterait aussi la conception, en simplifiant la définition des missions accomplies par UAM dès l'étape du design conceptuel.

Au niveau du guidage, l'ajout de capacité de manipulation rend la conception de nouveaux algorithmes nécessaires. La grande variété de tâches pouvant être accomplies par les UAM amène un besoin de méthodes de support à la conception de ces sous-systèmes qui puissent être facilement adaptables et applicables aux contextes d'utilisation définis par les concepteurs. Bien que les questions de planifications de trajectoire et évitement d'obstacles ont été traitées pour les UAV, des sous-systèmes liés à l'accomplissement de tâches, à la génération d'objectifs de missions ainsi qu'au support à la décision, demeurent essentiels pour les UAM et sont toujours inexistants. Le manque de classification des tâches possibles et de mesures de référence applicables à ces cas rend aussi le développement d'outil d'aide à la conception difficile, sans base commune. Le DL peut amener une avenue de solution possible pour apprendre à prédire le choix de l'UAM optimal parmi une flotte à l'aide d'une représentation de la tâche et de sa scène locale, sous forme de nuage de points. La méthode à base de DL pourrait aussi être suffisamment flexible pour permettre le support à la conception de l'algorithme en fonction des besoins du concepteur et du cas d'utilisation.

Les outils de support à la conception intégrée de systèmes mécatroniques incluent la structure conjointement à la loi de commande, ce qui parfois n'est pas possible ; les UAM sont souvent conçus à partir de multicoptères commerciaux. Les lois de commande, qui bénéficient de la plus grande part de travaux accomplis, possèdent des structures complexes issues du domaine de la commande non-linéaire les rendant difficiles à analyser et à adapter. La synthèse $\mathcal{H}_\infty$ structurée permet une piste de solution pour le développement d'une méthode de support à la conception de loi de commande pour les UAM qui serait simple dans leur structure, sans nécessiter des experts en commande robuste ou non-linéaire pour le développement et l'implémentation. Cependant, il n'a jamais été utilisé avec des UAM et donc nécessite une structure spécifique qui tient compte des efforts induits par le déplacement du bras robotique sous le multicoptère.

CHAPITRE 3 ARTICLE 1 : TASK TAXONOMY FOR AUTONOMOUS UNMANNED AERIAL MANIPULATOR : A REVIEW

Charles Coulombe, Jean-François Gamache, Olivier Barron, Gabriel Descôteaux, David Saussié, Sofiane Achiche

Proceedings of the ASME 2020 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. Volume 9 : 40th Computers and Information in Engineering Conference (CIE). Virtual, Online. August 17–19, 2020.

Les coauteurs Jean-François Gamache, Olivier Barron et Gabriel Descôteaux ont contribué à l'analyse des vidéos, à la génération d'idées pour la taxonomie et à la révision de l'article. Les méthodologies, la revue de littérature, la formalisation de la taxonomie, l'écriture de l'article et la création des figures ont été accomplies par Charles Coulombe avec le support des directeurs de recherche Sofiane Achiche et David Saussié.

La contribution principale de ce chapitre est le développement d'une taxonomie des tâches pouvant être accomplies par les UAM et répond ainsi au sous-objectif **SO1**. Cette catégorisation s'inscrit dans le support à la conception d'UAM comme un outil pouvant classifier les tâches possibles, et fournissant des blocs élémentaires sous forme de tâches simples permettant de diviser des missions complexes. La classification a été effectuée à la suite d'une revue de littérature et d'une revue vidéo en différenciant les tâches selon les capacités cognitives nécessaires. Elle permet aussi de définir la tâche utilisée pour les deux prochains chapitres, soit la saisie d'objets statiques.

3.1 Abstract

The development of unmanned aerial manipulators (UAMs) allows a novel class of flying robots to carry out a wide variety of tasks in difficult environments due to their versatility and autonomy. However, the different tasks that can be carried out might call for different control strategies. To this end, one needs to categorize the possible tasks accomplishable by UAMs. This paper proposes a novel taxonomy, which is the result of a video information acquisition methodology combined with a review of research works in the literature. The different elements of the taxonomy are separated using a higher level of abstraction in a way that the general description of the tasks are considered and not its operational details. To illustrate the fact that algorithms must adapt to different tasks, a description of the usual UAM architecture is carried out. Four categories of criteria are used in the taxonomy

to differentiate all possible tasks. These categories are the interaction type, the actual task definition, the environment condition and the time sensitivity of the task. This taxonomy forms the basis for possible machine-learning-based task classifiers that could be used in autonomous UAMs control and mission planning. Multiple tasks defined in the taxonomy can be combined to accomplish complex missions.

3.2 Introduction

In recent years, the use of unmanned multirotor systems, especially quadcopter drones, has skyrocketed in multiple domains such as aerial photography, mapping, sensing and even racing. While certainly useful, one of their drawbacks is their low capabilities of interaction with their environments. To address this challenge, the use of unmanned multirotor systems equipped with manipulating devices has started to attract the attention of both academic research and commercial products. Those systems are named either Small-scale Rotorcraft Unmanned Robotic Systems [15], Unmanned Aerial Manipulator (UAM) [6], or simply Aerial Manipulators [27]. In this paper, the term UAM is used to refer to a flying unmanned multirotor base equipped with a device enabling it to interact with the environment, namely a manipulating device. One such UAM is presented in Fig. 3.1.

Figure 3.1 An unmanned aerial manipulator developed in our laboratory.

A task is defined as a precise objective that must be carried out by a UAM, such as picking or placing an object, perching, etc. It is composed of subtasks that define all the steps necessary to fulfill the task. Some subtasks are common to multiple tasks, such as flying or sensing the environment. Missions are defined as an objective that consists of multiple tasks joined together towards a greater goal. Figure 3.2 illustrates the relationship between the three terms, as used in this paper.

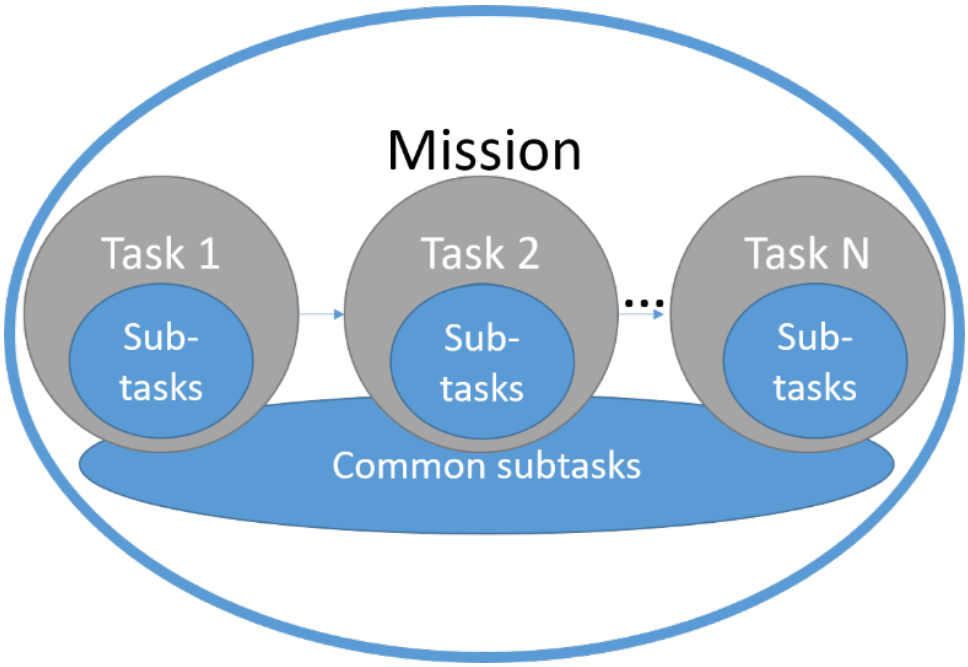

Figure 3.2 Relation between mission, tasks and subtasks.

UAMs are used in various difficult or hazardous environments and perform in a wide variety of conditions and domains, such as visual inspection [2], construction [105] or canopy sampling [106]. Recent trends in research move towards the autonomous completions of tasks by UAMs. This variety of domains and accomplishable tasks suggests that a taxonomy is needed to categorize them all. In its work on UAMs, [68] defines three possible types of interactions with the environments : momentary coupling, loose coupling and strong coupling (perching). While not defined or built as a taxonomy, the idea of classifying a task based on the interaction with the environment is in line with the work presented in this paper.

As mentioned in [11], taxonomies based on the effects of the actions on the environment are needed to guide the development of autonomous robots and to design them to accomplish a multitude of tasks. Such taxonomies are useful for classifying existing work and identifying future developments in the domain. While most of the existing classifications of tasks deal with the grasping aspect of a picking task, none classifies the type of tasks carried out using

a manipulating device. UAMs are able to deal with more varieties of tasks than a fixed-base robotic manipulator would, enlarging its capabilities beyond the usual picking and placing of slow-moving objects. A complete review of existing taxonomies is presented in Section 3.3 of this paper. The proposed taxonomy serves as a classification and identification tool of all possible tasks accomplishable by a UAM. The reasoning behind the need for a taxonomy is to allow one the ability to differentiate between tasks based on the needed UAM's cognitive capabilities. Cognitive capabilities are defined as the ensemble of algorithms for guidance, control or navigation, needed to accomplish a task. Each final element identified in the taxonomy therefore needs a different set of cognitive capabilities to accomplish the identified task. The proximity of different elements in the taxonomy implies that those elements necessitate similar cognitive capabilities.

The proposed task taxonomy could also be used in a completely autonomous UAM for task selection. This system would be able to take "mission-driven decisions", to obtain a "high adaptation to mission changes", and would be represented by a level 6 autonomous system on the autonomy level for unmanned rotorcraft system developed in [62]. A decision-making classifying algorithm would allow the aerial manipulator to choose between the tasks with respect to the context and environment using artificial intelligence to determine the most likely task from predefined categories. The proposed taxonomy proposes a systematic classification of tasks accomplishable by UAMs so that it can be used as a basis for the classes in such a system. Control strategies and navigation algorithms associated with each category of tasks could then be used to accomplish multiple tasks, without human intervention. For example, criteria could be set to recognize each element of the taxonomy and therefore enabling the correct navigation, guidance, control and interaction algorithms for the tasks at hand. Such system would rely on all recent advances in computer vision for unmanned multirotor systems [3] and artificial intelligence, especially deep learning for robotics [4] and unmanned aerial vehicles [107]. A similar project is in the work at an Alphabet company named "X, the Moonshot Factory" with their Everyday Robot Project [108], where a mobile robot equipped with a manipulator is trying to learn how to accomplish useful tasks in a typical desk-job environment, depending on the environment itself and human interactions.

The main objective of this paper is therefore to introduce a taxonomy of tasks relevant to UAMs. To develop it, a video information acquisition methodology was used, where multiple videos related to UAM were viewed and analyzed. While videos (YouTube, etc.) are not generally used in formal reporting, we chose to include them as many developments in the field are into their infancy stage and therefore researchers, laboratories as well as developers tend to post their experiments and early developments before even using formal publication channels. This methodology was used in conjunction with the literature on UAM and on

taxonomies to develop all possible tasks carried out by UAMs. To explain the effects of task classification on the inner workings of a UAM, its architecture and the interdependencies between subsystems are presented. This work considers some constraints. Only UAMs defined as an unmanned flying multirotor robot equipped with manipulating devices are covered in this taxonomy. It will also focus solely on tasks carried out by a single unit UAM, with a single robotic arm and will not deal with multiple robots or swarms.

3.3 Taxonomy Review and UAM Architecture

3.3.1 UAM Architecture and Subsystem Interactions

Design choices for subsystems used in an autonomous UAM are greatly influenced by the tasks to be accomplished. Figure 3.3 shows the subsystems usually present in an autonomous UAM, but variations may exist depending on design choices, such as in the choice of different sensors, or navigation capabilities. This diagram was adapted to UAMs from [3] in order to show the interactions between all subsystems. Through different sensors, the UAM is able to obtain measurements of its dynamic states and information from its environment. This information is processed using several algorithms to obtain a representation of the immediate environment of the UAM and to estimate its pose and corresponding derivatives. This is usually done in conjunction with information from the guidance subsystem that deal with generating trajectories and planning missions. Using the state estimations, environment mapping and mission planning information, the control algorithm is able to calculate the motor commands necessary to obtain the desired trajectory, arm movement and environment interaction. All of these relationships are represented in Fig. 3.3 in which, P/V/A means position/velocity/acceleration.

Since the control strategies, including navigation and guidance algorithms might change depending on the task, it is important to have a classification that distinguishes and identifies when different algorithms are required. If a UAM's task is to push on a surface, the control must be able to evaluate the generated force by the system and compensate for variations. While the position must also be tracked with precision to apply the force at the right point, it is not crucial to do it rapidly. A controller such as [109] would be appropriate for this application. Meanwhile for picking up a moving object, feedback on applied force is mostly useless, while rapid and precise tracking and following of the objective is essential to the success of a task. It would then require a completely different controller such as the one in [110]. While the use of specialized tools might necessitate a different control strategy per tool, this work does not distinguish between them. The proposed taxonomy considers a generic

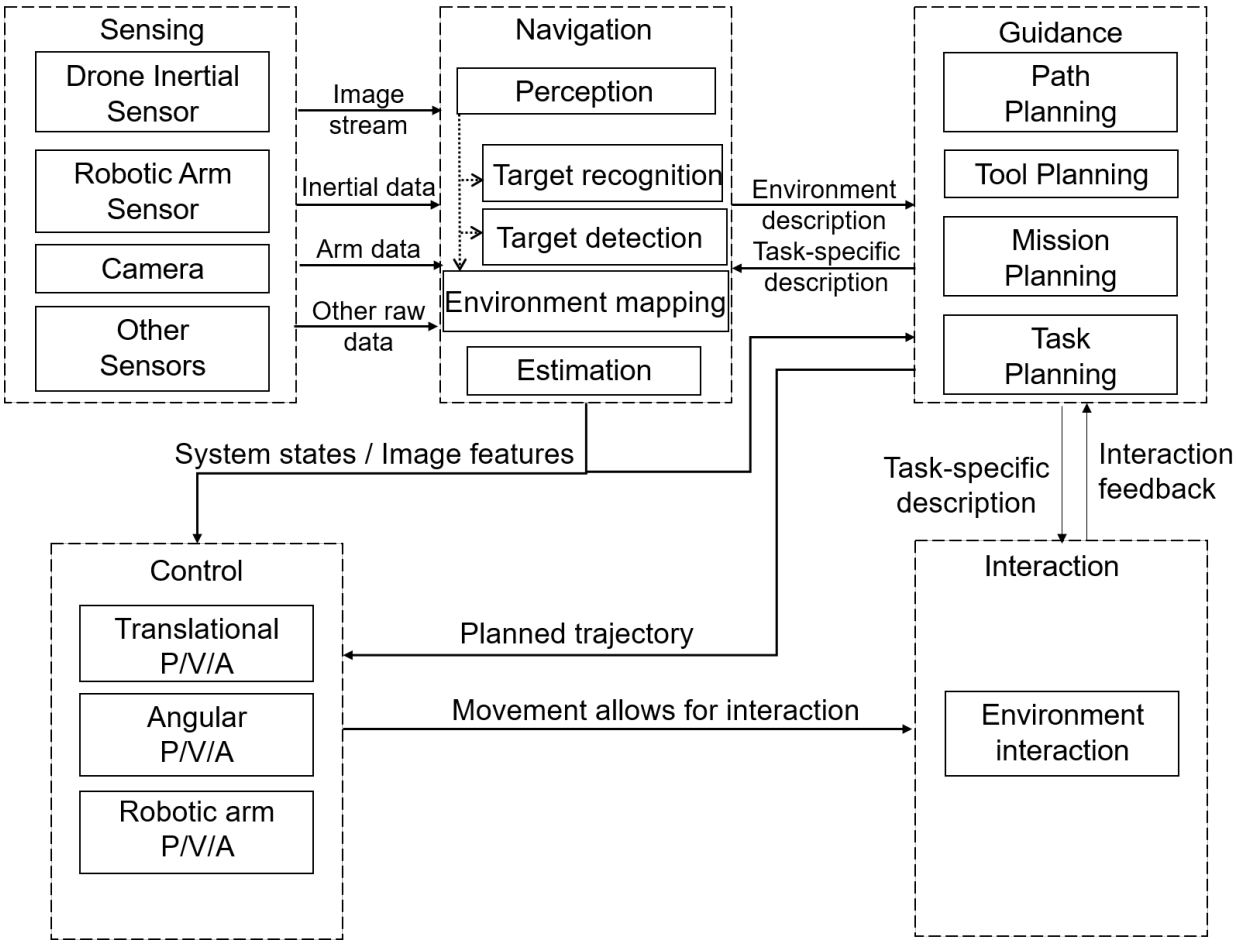

Figure 3.3 Schematic showing all interactions between the different subsystems in a usual UAM architecture.

robotic arm with standard grippers as the physical capability of the system. Specialized tool implementation details would have to be developed in a case-by-case manner. Tasks accomplished with specialized tools are still being covered by the proposed taxonomy. In other words, this taxonomy differentiates between cognitive capabilities, but not physical ones.

Considering how all subsystems interact in a UAM, the guidance algorithms play a crucial role. Guidance regroups all algorithms used for path planning, grasp planning, obstacle avoidance, and mission planning. They interact with the rest of the UAM to generate the desired trajectories and behaviors. The command and information obtained from these subsystems allow the UAM to interact autonomously with its environment. These algorithms are extremely dependent on the task or missions that is currently carried out.

3.3.2 Taxonomy Literature Review

Existing taxonomies in the literature cover different aspects of robotic manipulation. Multiple taxonomies on grasping tasks were already proposed. For example, the GRASP taxonomy introduced a hierarchical vision of grasping were each grasping action can be classified in one of the 33 defined types of grasps, ranging from power grasping to precision grasping [65]. A grasp taxonomy based on the relation between hand and object, looking at temporal anchor points was presented in [66]. Finally, another grasp taxonomy where tasks were described in simple words describing everyday actions by humans was introduced in [67].

A framework for classifying fine robotic grasping and assembly was presented in [111] and did not consider the geometry and physical characteristics of the grasped object. Other types of taxonomy for robotics include a taxonomy for task allocation in multi-robot systems [112], a taxonomy for dance in social robots [113], and a taxonomy for communicating with small multirotor systems using hand gestures [114], where each of these taxonomies deal with specific tasks. It was argued in [11] that a taxonomy based on the effects of the actions of the environment was needed with the end results being a taxonomy for compliant manipulation task [11], in which tasks are described independently of robot kinematics. Other criteria used in robotic taxonomies are the following : number of robots (single robot vs multiple), complexity of missions (single task vs multiple task) [112] and haptic feedback [115]. All those taxonomies from the literature cover specific aspects of a robotic task such as grasping and do not allow to categorize a multitude of tasks at a higher abstraction level without considering the configuration of the robot. This is mostly because currently used robot manipulators are extremely specific in their task description, meaning that a single robot is usually used and programed to perform a single task, or a small subset of similar tasks.

Considering the literature on taxonomies of tasks, one is proposed to classify all tasks achievable by a UAM, while staying at a higher level of abstraction that does not consider the geometry of the object, the type of the grasp needed, nor the manipulator used. Consequently, physical capabilities are not considered, while cognitive capabilities are used to differentiate between all tasks. While the UAM configurations and manipulating devices choice may influence its ability to accomplish a task in particular, it does not intervene in our proposed classification.

3.3.3 Video Information Acquisition Methodology

To develop the taxonomy and consider all plausible tasks possible for an aerial manipulator, a video information acquisition methodology is used. Since the topic of aerial manipulation

yields results that are easily viewable in video form, it was determined to be a promising avenue and was confirmed by major research laboratories working on the topic publishing their results in such form. The systematic video information acquisition methodology was then carried out by watching the first hundred videos on YouTube using the keywords “aerial manipulation”, “aerial manipulator”, “drones with robotic arms” and “aerial robotics”, as of November 2019. Each video was classified by their domain of origin, either academic research, commercial product or hobbyist solution. The fulfilled task and the manipulating device used were recorded. After removing duplicates and non-relevant videos (swarms or non-UAM related), 62 were left. Following this step, all results were regrouped in similar categories for the fulfilled task and the UAMs configuration. While doing this first categorization, a higher-level of abstraction regarding the tasks was kept in mind, in essence with the idea of the proposed taxonomy. Results of the video information acquisition are summarized in Figs. 3.4 and 3.5, where the tasks and configurations used are categorized within similar class, counted and separated by their domain of origin.

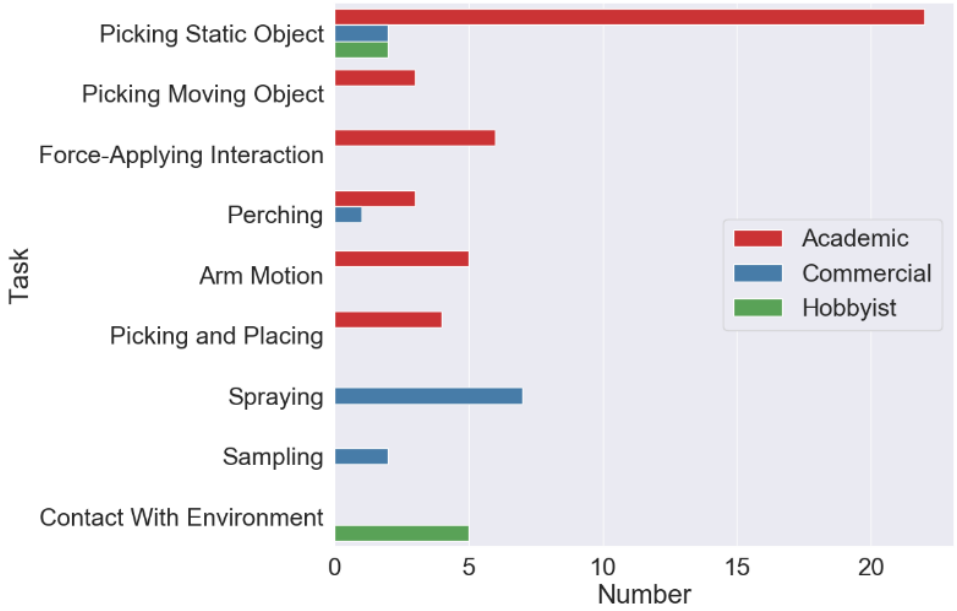

Figure 3.4 Distribution of tasks in video information acquisition methodology by domain.

Figure 3.4 clearly shows that academic research focus on picking static objects using a UAM. While some works were carried out on other tasks such as picking a moving object, perching, force-applying interactions, simple arm motions or placing, most still focused on this simple but essential task. Commercial products mostly focus on the use of UAM for spraying of liquid (paint, insecticide, water, etc.). The task categories chosen here are a preliminary version of the ones developed in our taxonomy. Figure 3.5 shows that the most common manipulating

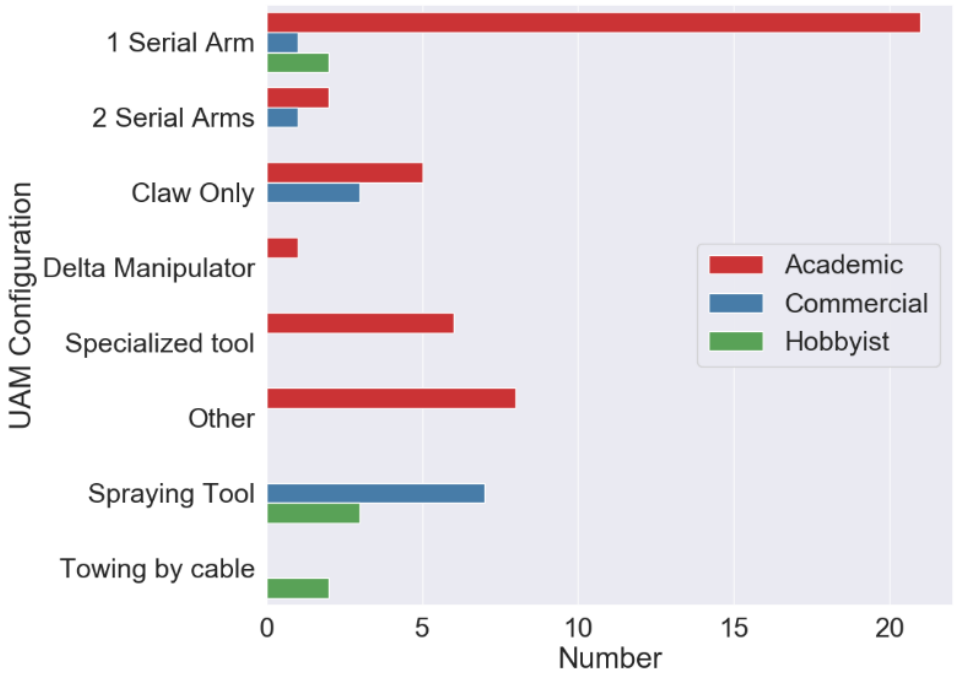

Figure 3.5 Distribution of manipulator configuration in video information acquisition methodology by domain.

device used is a single serial robotic arm equipped with a gripper, followed by the use of a specific tool designed for the task at hand, such as perching tool or sampling tool. The results of the video information acquisition were then used in preparation for the elaboration of the taxonomy presented in Section 3.4.

3.4 Taxonomy and Examples

Using the results of the video information acquisition methodology presented in Section 3.3 and recent literature on UAMs [6, 15], we developed the taxonomy presented in Fig 3.6. Notably, the work of all involved research groups was considered [15] to produce the taxonomy and consider all possible tasks.

All the right-end side cases are considered as different tasks in this taxonomy. Figure 3.6 shows all 20 possible tasks identified. Each vertical column is a separation by a certain criterion. The first column lists the type of interaction, the second lists the tasks end results on a high abstraction level, while the last column gathers environmental constraints applicable to the tasks. Another criterion, namely time sensitivity, is also present for each of the 20 tasks, but was not shown in Fig. 3.6 for ease of visualization consideration. This effectively implies that all 20 tasks have time sensitive and non-time sensitive versions of themselves.

The first column in Fig. 3.6 separates for types of interactions. They are defined as punctual object interaction, continuous object interaction and no physical interaction. As mentioned in [68], these three categories are the only possible types of interactions between a UAM and its environment. In the first, a UAM interacts punctually with an object. This is similar to the concept of a momentary coupling, identified by [68]. It is usually done in the form of picking (which includes grasping), placing, or sampling. A special task is the delivery of a direct impact using the manipulation device.

For continuous object interaction, the UAM applies a force or a moment for a certain amount of time on a surface or an object. This category is divided between direct application, where forces are applied by physical contact with the manipulator, and indirect application, where forces are applied to the object using a cable or a rod. It is also divided between static application and moving application of forces. A special case of continuous object interaction is the perching task, where the aerial manipulator uses its manipulation device to attach itself to a surface and stop its motors. This category could correspond to the loose and strong coupling identified by [68].

The last category considers the movement of the manipulation device without any interaction with the exterior environment. Some tasks require a movement of the manipulator or even the entire UAM but without applying any forces or moments of any kind on the environment. Such tasks could involve spraying in agricultural context or moving a light attached to the arm in a dark environment.

Those three categories are then subdivided between task definitions, represented by the second column in Fig. 3.6, by considering the results of the tasks. Finally, the last column differentiates between environmental conditions that could necessitate a change in cognitive abilities of the UAM to accomplish the task, effectively bringing the total of different tasks identified by the taxonomy at 20. These conditions change the set of algorithms needed in the different subsystems identified in Fig. 3.3 and therefore were identified as different tasks. Evidently, all tasks presented in Fig. 3.6 are carried out while the UAM is airborne. This implies that all tasks include subtasks such as liftoff, movement, environment scanning and mapping, target identification and landing. Since they are accomplished in any mission, they are not considered in the taxonomy. These are the common subtasks identified in Fig. 3.2. The next paragraphs explain the different tasks present in the second column of Fig. 3.6.

Picking : This task consists of picking up an object using the attached manipulation device. The UAM identifies the target, and plans the manipulator trajectory and grasping point to grasp the object successfully. No difference is made between the different types of grasps, as the GRASP taxonomy would [65]. If desired, these classifications could be used conjointly

with Fig. 3.6. The environmental conditions distinguish between the statuses of the object to pick. In the first case, the target is static. As Fig. 3.4 shows and as found in the literature, most research is usually targeted towards the picking of a non-moving object, as exemplified by [116], where a vision-based algorithm for the center of mass extraction and grasp planning is presented or [50], where a fish-eyed lens is used for depth and grasp planning. In the second case, the object is moving in 2D, i.e., the target is moving in a plane usually parallel to the ground, or in 3D. This challenging task was explored by [110] where a UAM catches a cylinder on a mobile robot. While no work concerning picking of 3D moving object using a single UAM was found in the literature, this task seems the logical next step and is already explored for example with three quadcopters catching and throwing a ball in the air using a net [117].

Placing : The placing task consists of depositing an already grasped object at a certain position and orientation. Using the task description, the UAM must navigate and orient the payload to fit the desired drop conditions. We distinguish between the different drop conditions. First, no desired conditions simply result in dropping the object at random. A desired position is a point in the environment of the UAM that will serve as a drop point, while a desired orientation indicates that the object must be oriented correctly, with respect to both itself and its environment. When a desired position or orientation is needed, a distinction is made on the placing condition. A fixed placing condition means that the position or orientation is directly specified in the task or mission description, while a relative placing condition is defined by previous fulfilled tasks. This last kind of placing task is used in construction or assembly projects, where the next piece to be placed depends on what has already been assembled or constructed. A fixed desired position and orientation task example would be the peg-in-hole assembly, defined in [118]. A swarm of quadcopter drones with grippers was used in an assembly project, to assemble a tower of blocks [6] or simple assemblage [59, 105], and illustrate the relative drop point with desired position and orientation task. While these projects dealt with swarms of quadcopters, a unique UAM could accomplish the same tasks (simply taking more time), so therefore it was included in this taxonomy. The last task in this class is a throwing task, where the object is ejected with a certain velocity from the manipulator. While no work was found using a UAM, a project such as [119] where a fixed robotic arm learns to throw objects could also be carried out with an aerial manipulator.

Sampling : Sampling consists of collecting part of an object, usually with the help of a specialized tool. An example of sampling is found in the tree canopy sampling using a UAM [106]. Canopy sampling is an example of a destructive sampling where the UAM must cut out part of the object to take a sample, while water sampling in a river would be an example

of non-destructive sampling. In both cases, the UAM must identify the relevant object for the task using its sensor and plan the execution of its sampling task with the tool, while considering the environment and exterior conditions.

Direct impact : This is a special type of punctual interaction where a direct shock being received or delivered by a UAM, similar to the impact of a hammer. While no work was found on this task, it can conceivably be imagined as a UAM using an impact device to carry out reparations in hard-to-reach environments, such as wind power plants or overhead electric transmission lines.

Direct contact / Indirect contact : As mentioned earlier in this section, these tasks are about applying forces or moments to a target in the environment for a certain period of time. The UAM identifies the point of interest in its environment and must control its movements to deliver the adequate amount of force to the target, using different methods and measures. The type of task, second column in Fig. 3.6, differentiates between a direct and an indirect contact. The UAM applies the force directly to the object using its robotic arm in the former, and could apply a force indirectly in the latter, for example by pulling a cable linked to the object as in the case of suspended payload [120]. The environmental conditions separate if the force needs to be applied at a static point or in motion. Examples of direct contact, static force applications could include the pose of a sensor [119], while a moving trajectory could consist of drawing on a board or contact-based inspection [121].

Perching : A special case of continuous object interaction is the perching task, where a UAM uses its robotic arm to attach itself firmly to an object while shutting down its rotor-motor subsystems. Perching was studied a lot on UAMs equipped with special perching apparatuses [21, 122], but a general manipulator with a robotic arm can conceivably accomplish this task as well, such as [123]. Cognitive capabilities for a UAM changes depending on the environment conditions of perching, whereas the perching surface is static or moving.

Arm trajectory / Arm endpoint : Some tasks only require a movement of the robotic arm (or the drone) without having any physical interaction with the environment. Two task definitions were identified. In the first one, the complete arm must follow a trajectory while, in the other, the arm must only reach a certain endpoint. The environmental constraints in the case of motion-only tasks would be the need for environmental feedback. As an example, if the task is to pour a liquid in a cup, the need for precise visual feedback for relative position of the UAM to the cup is clear. However, when using a spraying implement in an agricultural field, there is no need for precise visual feedback and a simple GPS position is sufficient. Evidently as mentioned earlier, time sensitivity is another criterion for task classification, not shown in Fig. 3.6. A time-sensitive complete trajectory example is found in photography,

where a camera moved by a manipulator needs to control its entire trajectory as to keep shooting steady videos, as with the commercial system DJI Inspire 2 [124]. On the other hand, to measure damage on a system with a laser, only the endpoint position is required and it is not time sensitive. As seen in Fig. 3.4, little academic research is conducted on this type of task because of its relative simplicity. Moving and controlling a UAM without physically interacting with the environment is a problem that is essentially solved from a control point of view [15].

All the previously explained tasks are the building blocks of mission planning and can describe any complex missions. The control, sensing, navigation and guidance subsystems as defined by Fig. 3.3 must adapt to the tasks. We exemplify the use of the classification with a drink-serving mission. It would require two tasks defined in the taxonomy, both picking a static object and an arm trajectory with environmental feedback. The guidance algorithms would have to be chosen in order to be able to grasp the bottle while keeping its orientation. Another subsystem would have to plan how to move the robotic arm, and the bottle gripped by it, in order to pour over the glass. Non-task dependent subsystems include global navigation, sensing, flying control, etc. Using this taxonomy, multiple missions could be defined as needed in different domains, such as construction, manipulation, reconnaissance, etc.

As mentioned, this taxonomy considers solely tasks accomplishable by a single UAM with a single robotic arm. While the taxonomy presents all possible tasks for this specific configuration, new tasks and possibilities open when dealing with new configurations. For example, the use of two UAMs or swarms allows for collaboration between the robots. New tasks could become feasible such as collaboration in the moving of objects or complex assembly tasks. For these tasks, new algorithms of mission planning must be used. Therefore, new tasks and categories in the taxonomy should be added with respect to the reasoning of this taxonomy that each endpoint is a different set of cognitive capabilities of a UAM.

3.5 Conclusion

A taxonomy is presented that classifies the different tasks that can be accomplished by a single UAM equipped with a manipulating device. The classification is completely independent of details such as geometry of the target or physical capabilities of the system, keeping only a higher level of abstraction and separating for cognitive capabilities. Since this taxonomy does not consider physical capabilities, it can be used in conjunction with more specific ones such as grasping classifications. The different tasks are grouped into three categories of interaction, punctual object interaction, continuous object interaction and no environment interaction. They are then separated by their task description and environmental conditions. Most tasks

are already explored in the literature and given as examples. Such a taxonomy is useful for future work to identify and classify new advances in autonomous UAM development. Future work should first focus on the association of all developed tasks with existing algorithms and cognitive capabilities of a UAM. For the multiple selected tasks, a UAM could then use the taxonomy as the base to a logic system that could detect which task the UAM must carry out, based on the signal and information in its environments, or human interaction. This logic system could certainly be supported by artificial intelligence, where the categories of the classifier are those defined by this taxonomy. Secondly, the taxonomy could be broadened to take into account all UAMs that have not been considered in this work, including swarms and multi-arm systems.

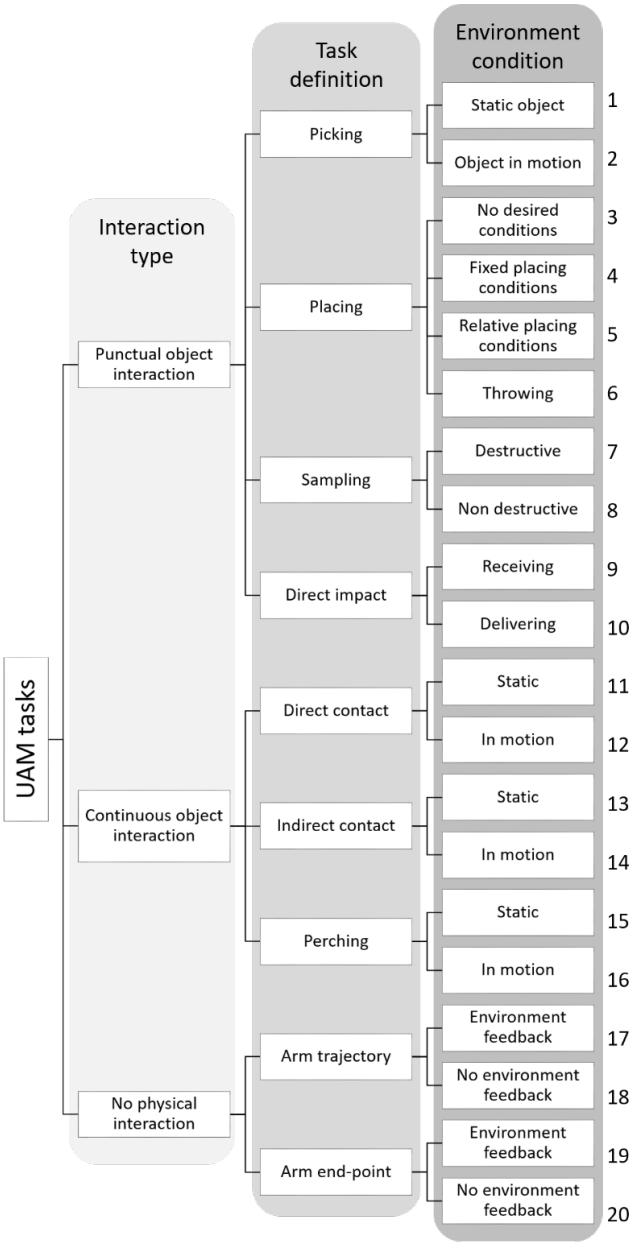

Figure 3.6 Taxonomy of tasks accomplishable by a single UAM.

CHAPITRE 4 ARTICLE 2 : MODELING AND GAIN-SCHEDULED CONTROL OF AN AERIAL MANIPULATOR

Coulombe, C., Saussié, D. & Achiche, S. Modeling and gain-scheduled control of an aerial manipulator. International Journal of Dynamics and Control (2021). https ://doi.org/10.1007/s40435-021-00807-2

Ce chapitre présente une méthode supportant la conception d'une loi de commande de vol à gains séquencés pour UAM équipé d'un bras robotique à 2DDL, et répondant au sous-objectif **SO2**. Une structure de loi de commande est introduite pour la configuration de l'UAM et les gains sont obtenus par optimisation $\mathcal{H}_\infty$. Cette méthode repose sur une modélisation dynamique de l'UAM développée dans cet article dans un format facilitant son analyse et son utilisation avec toutes configurations d'UAM. La structure de séquence de gains a été démontrée comme minimale pour pouvoir stabiliser le système. L'obtention de la loi de commande par optimisation vient réduire son problème de conception à la définition des requis dynamiques et des paramètres physiques modélisant le système à commander. Cette méthode vient simplifier l'ajustement de la loi de commande lors d'itérations de l'UAM durant le processus de conception, par exemple si les paramètres physiques sont modifiés. L'obtention d'une nouvelle loi de commande ne nécessite alors qu'une nouvelle optimisation.

4.1 Abstract

Aerial manipulators (AMs) are flying robotic systems composed of a multicopter equipped with one or more robotic arms. The dynamic modeling of AMs is complex due to the highly coupled interactions between the multicopter and the robotic arms. In this paper, the dynamics and kinematics of a quadcopter are combined with the Newton-Euler recursive method to obtain the forces applied by the robotic arm on the multicopter. The complete dynamic model is then linearized and expressed in terms of the articulation angles of the robotic arm for controller synthesis purposes. A novel gain-scheduled controller based on structured $\mathcal{H}_\infty$ synthesis is proposed using the joint angles of the robotic arm as scheduling variables. The controller is tuned according to multiple performance objectives on a family of parameterized linearized models. Simulations on the non-linear model with the gain-scheduled controller demonstrate its ability to compensate for robotic arm movements, while keeping the desired dynamic characteristics regardless of arm position. The advantage of gain scheduling is demonstrated by a comparison with a non-scheduled version of the controller. Finally, the controller is shown to be able to compensate for a 10% uncertainty on the nominal tuning

values of the physical parameters.

Keywords : Aerial manipulator ; Kinematic and dynamic modeling ; Structured $\mathcal{H}_\infty$ synthesis ; Gain-scheduling

4.2 Article Highlights

— Dynamics of aerial manipulators is obtained by combining the dynamics of a quadcopter with the forces and moments generated by a moving robotic arm fixed underneath it. Equations are presented in a simulation-ready format. Using the complete nonlinear dynamics of the aerial manipulator, linearized models are obtained for multiple positions of the manipulator. This family of linearized models is necessary for the multi-model synthesis.

— A linear gain-scheduled controller is shown to be able to stabilize an aerial manipulator equipped with a two-degree-of-freedom arm when using quadratic scheduling functions with the arm joint angles as gain-scheduled variables. In a single structured $\mathcal{H}_\infty$ synthesis, the gain-scheduled controller is tuned to obtain the desired performances over a grid of linear models.

— The controller is shown in simulations to stabilize the system with the required performances. Further simulations show the clear advantages of the gain-scheduling aspect of the controller by comparing it to a non-scheduled version of itself. Finally, the closed-loop system is shown to be able to compensate for and variations on the nominal physical parameters values.

4.3 Introduction

Multicopter drones are small flying robots that can accomplish different missions due to their maneuverability, and ability to access remote environments. Examples of these capabilities include coastal photography [1], structural inspection [125] or even competitive racing [126]. Since multicopters do not possess interaction capabilities, their uses are limited to movement, measurement and observation tasks. Multicopters, often quadcopters, equipped with serial robotic arm(s), named aerial manipulators (AM), are raising both academic and industrial interests for their ability to accomplish complex tasks in hard-to-reach or dangerous environments [127]. As for multicopters, AMs are unstable systems and they must be equipped with a flight control system able to stabilize the system and counteract the added arm effects [12].

Several dynamic models of quadcopters have been proposed, each, with various degrees of fidelity [128–130]. Usually, most of the models assume a symmetrical structure of the quad-

copter where both axes of symmetry cross at a fixed center of mass (CoM) [97, 131]. A model dealing with dynamic changes to its center of mass was presented in [132]. Quadcopter dynamics are well known and were published in books such as [133].

Modeling the dynamic interactions between the arm and the drone is a tedious task to carry out and is needed to obtain the complete dynamics of an AM. This task can be treated by different methods, such as developing the complete kinematics of the system to compute the dynamics, or by considering the arm as an external perturbation acting on the quadcopter. Commonly used methods for dynamic modeling are Euler-Lagrange [26] and Newton-Euler [17, 134] approaches. The latter one was shown to be more adapted to controller design, analysis and simulations while the former one was used for studying the inertia variations [68]. For example, the work presented in [135] modeled the AM as a drone with added forces and moments generated by the robotic arm. In [30], the system was modeled as a 6+(n+1)-DoF system, where n is the number of links of the robotic arm. A drone equipped with two 2-DoF arms, each with RP joints, was modeled in [17]. A complete mathematical model of an AM equipped with a 2-DoF arm using the kinematics of the system to develop its dynamics was presented in [134]. However, a recurrent problem in many publications is an over-simplification of the modeling of the drone. The coordinate system transformation equations are often overly simplified or the used hypotheses are too constraining. Often, models obtained using small Euler angles hypotheses are used out of these hypothesis boundaries as in acrobatic maneuvers. In summary, dynamic models are often presented for a single, specific case, or are too simplified, overly constrained or simply in an unsuitable form for controller synthesis.

Different controllers were proposed for the stabilization and control of quadcopters and multicopters. Simple controllers such as PIDs and LQRs (linear quadratic regulators) were first proposed for quadcopters in [136]. These linear controllers are usually simple to design and to tune, but may be more sensitive to modeling uncertainties or to unplanned disturbances. More complex controllers such as backstepping controllers [97, 131] or an adaptive sliding mode controller [137] were also applied to a quadcopter for their added stability with regards to perturbations and robustness to system uncertainties.

Controllers that can deal with the interactions between the quadcopter and the robotic arm are needed in order to stabilize AMs. Simple controllers used for quadcopter alone are not able to stabilize an AM and therefore new controllers are needed. Those found in literature often possess complex structures in order to reduce the effects of interactions and dynamics that could have been missed during the modeling of the system. For example, adaptive sliding mode control is used in [138–141], for its ability to self-adapt to the situation at hand.

This control strategy accounts for the uncertainties in the system modeling. Backstepping control is used in [18, 47, 98], allowed by the cascade structure of the quadcopter dynamics. Backstepping control is often used for its robustness to uncertainties in the system [47]. Impedance control is used in [30, 142], as to counter external forces both from disturbances and contacts with the environment. In [134] the authors use a disturbance observer based controller that was shown to perform fewer calculations than more complex controllers, while keeping performance almost on par. In summary, most controllers in the literature are tuned directly on a non-linear model and are themselves complex in their structures, making them difficult to analyze and design.

A structured $\mathcal{H}_\infty$ synthesis framework [100, 104, 143] has been presented and used for the tuning of complex controllers with multiple design requirements. The framework allows to define a fixed controller structure, and tune it using $\mathcal{H}_\infty$-type requirements. This framework is also applicable for gain-scheduled controllers, which have the advantage of being able to adapt to different conditions of use by varying the gains whenever a change occurs in specified scheduling variables [144]. A linear gain-scheduled controller is simpler in its implementation when compared to backstepping and adaptive control strategies usually used for the control of AMs. Gain-scheduling was used in an adaptive controller [35], but this work mentioned the lack of an “explicit method for gain scheduling controller synthesis". Other gain-scheduling methods previously developed include an eigenstructure assignment method presented in [145, 146] on a controller with a structure similar to the one used in this paper, where the method was applied on a ballistic missile pitch-axis controller. This approach was later extended to fault-tolerant control for multicopters where the controller is scheduled with the level of detected faults [147, 148].

The main contributions of this paper are (1) the development of a generalised dynamic model for any AM configuration that is necessary for the controller synthesis and (2) the development of a novel gain-scheduled controller tuned within the $\mathcal{H}_\infty$ synthesis framework. Since the controller synthesis is based on accurate modeling of the AM with all the interactions betweem the quadcopter and the arm, a complete dynamic model is presented here as a preliminary step for the controller design. The dynamic model is then linearized around multiple equilibrium points. The proposed controller structure is a state feedback with integral action, where the tuning of the scheduled gain is obtained using an $\mathcal{H}_\infty$ synthesis framework on a grid of linearized models. The gains are scheduled with respect to the robotic arm joint angles. The structured $\mathcal{H}_\infty$ synthesis framework allows for multiple design goals and constraints when tuning the controller. This allows the introduced method to be used with any AM configuration. An AM composed of a quadcopter equipped with a 2-DoF robotic arm is used for the controller synthesis. A 2-DoF arm was chosen since it allows the end

effector to have 6-DoF when coupled with the quadcopter base.

Section 4.4 presents the detailed kinematic and dynamic modeling of the AM. Section 4.5 presents the equilibrium conditions and linearization of the system, while Section 4.6 introduces the gain-scheduled controller and its tuning using the structured $\mathcal{H}_\infty$ synthesis framework. Section 4.7 presents simulation results and performances of the controller in nominal and non-nominal cases.

4.4 Kinematic and Dynamic Modeling of an Aerial Manipulator

Accurate kinematic and dynamic modeling of the system is necessary for the development of the proposed gain-scheduled controller. The model is therefore presented in a suitable form for both controller synthesis and numerical simulations. The model is developed for a quadcopter equipped with a 2-DoF arm but could be applied to any AM configuration of a multicopter with a n-DoF robotic arm.

In this paper, a physical vector $\mathbf{x}$ expressed in a frame $\mathcal{F}_0$ is denoted $\mathbf{x}^0$, while ${}^0\dot{\mathbf{x}}^1$ is the time derivative of vector $\mathbf{x}$ with respect to the frame $\mathcal{F}_0$ and expressed in frame $\mathcal{F}_1$. The rotation matrix $\mathbf{R}_{1/0}$ between $\mathcal{F}_0$ and $\mathcal{F}_1$ belongs to the special orthogonal group SO(3), and thus one has $\mathbf{R}_{0/1} = \mathbf{R}_{1/0}^{-1} = \mathbf{R}_{1/0}^\top$. To simplify the notation, the short-hands $\mathrm{s}_x := \sin x$ and $\mathrm{c}_x := \cos x$ are used. Finally, the $\times$ operator denotes the vector cross product.

4.4.1 Kinematics and Dynamics of the Quadcopter

The kinematic and dynamic equations of the quadcopter are first introduced. Let $\mathcal{F}_i$ denote an inertial reference frame centered at a fixed local reference point and oriented according to the North-East-Down (NED) convention, and let $\mathcal{F}_b$ denote a body-fixed frame attached to the quadcopter, centered at its center of mass (CoM) and oriented with the $\mathbf{x}_b$ and $\mathbf{y}_b$ axis along the quadcopter arms as shown in Fig. 4.1. The 6-DoF rigid body and flat-Earth equations of a quadcopter form a set of twelve, coupled, scalar equations representing the kinematics and dynamics of a non-deformable rigid body with constant mass m and inertia matrix $\mathbf{I}_b$ [133]. For a quadcopter, the inertia matrix expressed in $\mathcal{F}_b$ is diagonal for symmetry reasons and denoted $\mathbf{I}_b = \mathrm{diag}(I_{xx}, I_{yy}, I_{zz})$. The robotic manipulator is treated as external forces and moments acting on the quadcopter, while also possessing its own dynamics that is affected by the quadcopter. The arm equations are presented separately in subsection 4.4.2.

The position vector of the CoM in $\mathcal{F}_i$ is denoted $\mathbf{p}^i = [p_N \; p_E \; p_D]^\top$, and the orientation of $\mathcal{F}_b$ relative to $\mathcal{F}_i$ is described by three Euler angles $\mathbf{\Phi} = [\phi \; \theta \; \psi]^\top$, respectively roll, pitch and

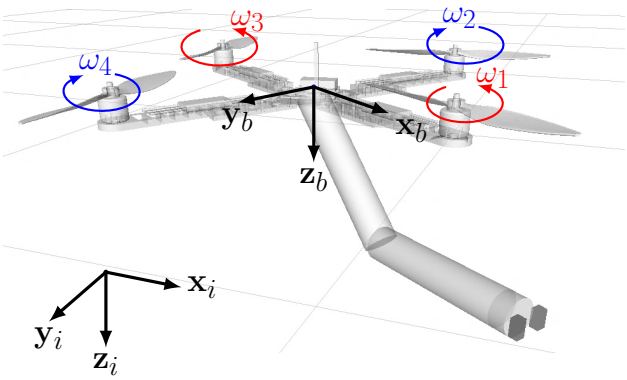

Figure 4.1 Inertial frame, body frame and drone configuration

yaw. The resulting rotation matrix $\mathbf{R}_{b/i}$ is thus parameterized as :

$$\mathbf{R}_{b/i}(\mathbf{\Phi}) = \begin{bmatrix} \mathrm{c}_\theta \mathrm{c}_\psi & \mathrm{c}_\theta \mathrm{s}_\psi & -\mathrm{s}_\theta \\ \mathrm{s}_\phi \mathrm{s}_\theta \mathrm{c}_\psi - \mathrm{c}_\phi \mathrm{s}_\psi & \mathrm{s}_\phi \mathrm{s}_\theta \mathrm{s}_\psi + \mathrm{c}_\phi \mathrm{c}_\psi & \mathrm{s}_\phi \mathrm{c}_\theta \\ \mathrm{c}_\phi \mathrm{s}_\theta \mathrm{c}_\psi + \mathrm{s}_\phi \mathrm{s}_\psi & \mathrm{c}_\phi \mathrm{s}_\theta \mathrm{s}_\psi - \mathrm{s}_\phi \mathrm{c}_\psi & \mathrm{c}_\phi \mathrm{c}_\theta \end{bmatrix} \tag{4.1}$$

The vector $\mathbf{v}^b = [u\ v\ w]^\top$ denotes the inertial linear velocity of the CoM expressed in $\mathcal{F}_b$ and finally, $\boldsymbol{\omega}^b_{b/i} = [p\ q\ r]^\top$ is the angular velocity of $\mathcal{F}_b$ with respect to $\mathcal{F}_i$.

Using the previously mentioned assumptions and notations, the kinematic and dynamic equations of a rigid body affected by any forces and moments are then :

$$^i\dot{\mathbf{p}}^i = \mathbf{R}_{i/b}\mathbf{v}^b \tag{4.2}$$

$$\boldsymbol{\omega}^b_{b/i} = \begin{bmatrix} 1 & 0 & -\mathrm{s}_\theta \\ 0 & \mathrm{c}_\phi & \mathrm{s}_\phi \mathrm{c}_\theta \\ 0 & -\mathrm{s}_\phi & \mathrm{c}_\phi \mathrm{c}_\theta \end{bmatrix} \dot{\mathbf{\Phi}} = \mathbf{H}^{-1}\left(\mathbf{\Phi}\right)\dot{\mathbf{\Phi}} \tag{4.3}$$

$$^b\dot{\mathbf{v}}^b = \frac{1}{m}\mathbf{f}^b - \boldsymbol{\omega}^b_{b/i} \times \mathbf{v}^b \tag{4.4}$$

$$^b\dot{\boldsymbol{\omega}}^b_{b/i} = \mathbf{I}_b^{-1}\left(\mathbf{m}^b - \boldsymbol{\omega}^b_{b/i} \times \mathbf{I}^b\boldsymbol{\omega}^b_{b/i}\right) \tag{4.5}$$

where the forces $\mathbf{f}^b$ and the moments $\mathbf{m}^b$ in Eqs. (4.4) and (4.5) are expressed in $\mathcal{F}_b$ for the sake of simplicity.

The forces can be broken down into three main sources :

$$\mathbf{f}^b = \mathbf{f}^b_g + \mathbf{f}^b_t + \mathbf{f}^b_{arm} \tag{4.6}$$

with $\mathbf{f}_g^b$ the weight of the quadcopter, $\mathbf{f}_t^b$ the thrust force generated by the rotors, and $\mathbf{f}_{arm}^b$ the forces due to the robotic arm weight and dynamics (developed in subsection 4.4.2). The vector $\mathbf{f}_g^b$ is given by :

$$\mathbf{f}_g^b = \mathbf{R}_{b/i} \begin{bmatrix} 0 \\ 0 \\ mg_0 \end{bmatrix}^i \tag{4.7}$$

with g_0 the gravitational acceleration. Each propeller generates a thrust force $T_j = k_t \omega_j^2$ along the $\mathbf{z}_b$-axis where k_t denotes the propeller thrust coefficient and ω_j the rotational speed of motor j for $j \in 1, 2, 3, 4$. The expression of the thrust force $\mathbf{f}_t^b$ is then :

$$\mathbf{f}_t^b = \begin{bmatrix} 0 \\ 0 \\ -\sum_{j=1}^{4} T_j \end{bmatrix}^b \tag{4.8}$$

The drone is subjected to moments from four main sources :

$$\mathbf{m}^b = \mathbf{m}_t^b + \mathbf{m}_r^b + \mathbf{m}_g^b + \mathbf{m}_{arm}^b \tag{4.9}$$

with $\mathbf{m}_t^b$ the moments generated by the lift forces, $\mathbf{m}_r^b$ the moments induced by the rotating motors, $\mathbf{m}_g^b$ the gyroscopic moments, and $\mathbf{m}_{arm}^b$ the moments generated by the robotic arm weight and movements (developed in subsection 4.4.2). The vector $\mathbf{m}_t^b$ is given by :

$$\mathbf{m}_t^b = \begin{bmatrix} dk_t \left(\omega_2^2 - \omega_4^2\right) \\ dk_t \left(\omega_1^2 - \omega_3^2\right) \\ 0 \end{bmatrix}^b \tag{4.10}$$

with d the distance between the CoM and each motor. Each propeller generates a reaction moment $M_j = \text{sign}(\omega_i) k_d \omega_j^2$ along the $\mathbf{z}_b$-axis where k_d is the propeller drag factor. The expression of the total reaction moment $\mathbf{m}_r^b$ is then :

$$\mathbf{m}_r^b = \begin{bmatrix} 0 \\ 0 \\ k_d \left(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2\right) \end{bmatrix}^b \tag{4.11}$$

Finally, the gyroscopic moments $\mathbf{m}_g^b$ are :

$$\mathbf{m}_g^b = \begin{bmatrix} 0 \\ 0 \\ J_m\left(-\omega_1 + \omega_2 - \omega_3 + \omega_4\right) \end{bmatrix}^b \times \boldsymbol{\omega}_{b/i}^b \tag{4.12}$$

where J_m is the combined motor and rotor inertia for a single motor.

4.4.2 Dynamic Modeling of a Robotic Arm on a Floating Base

The robotic manipulator is modeled as a 2-DoF serial robotic arm mounted under a floating base, i.e., the quadcopter previously modeled. It is worth noting that the same method could be used to model a n-DoF manipulator. The chosen manipulator remains in the $\mathbf{x}_b$-$\mathbf{z}_b$ plane, i.e., the links are rotating around the $\mathbf{y}_b$-axis of the body-fixed frame, as shown in Fig. 4.2.

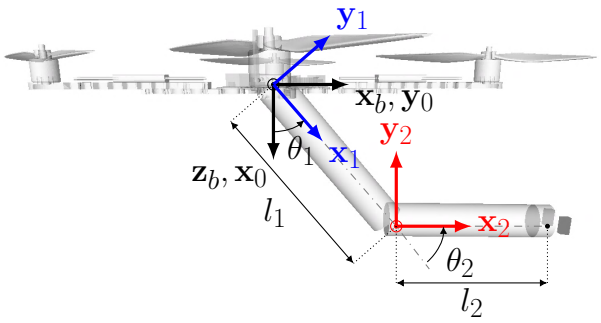

Figure 4.2 Robotic arm frames

Referring to Fig. 4.2, frame $\mathcal{F}_0$ is a body-fixed frame centered at the CoM of the quadcopter, and whose $\mathbf{x}_0$, $\mathbf{y}_0$, and $\mathbf{z}_0$-axis coincide with the $\mathbf{z}_b$, $\mathbf{x}_b$, and $\mathbf{y}_b$-axis of frame $\mathcal{F}_b$, respectively. Frame $\mathcal{F}_1$ is centered at the CoM and attached to link 1 ($\mathbf{x}_1$-axis along the link) ; $\mathcal{F}_1$ is obtained by rotating $\mathcal{F}_0$ through an angle θ_1 around the $\mathbf{z}_0$-axis. Frame 2 is centered at the junction of the first and second links and attached to link 2 ($\mathbf{x}_2$-axis along the link) ; $\mathcal{F}_2$ is obtained by rotating $\mathcal{F}_1$ through an angle θ_2 around the $\mathbf{z}_1$-axis. The rotation matrices

between the different frames linked to the manipulator are listed below :

$$\mathbf{R}_{b/0} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \tag{4.13}$$

$$\mathbf{R}_{0/1} = \begin{bmatrix} \mathrm{c}_{\theta_1} & -\mathrm{s}_{\theta_1} & 0 \\ \mathrm{s}_{\theta_1} & \mathrm{c}_{\theta_1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.14}$$

$$\mathbf{R}_{1/2} = \begin{bmatrix} \mathrm{c}_{\theta_2} & -\mathrm{s}_{\theta_2} & 0 \\ \mathrm{s}_{\theta_2} & \mathrm{c}_{\theta_2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.15}$$

The classic Newton-Euler recursive method is used to derive the dynamic equations of the manipulator [149, Ch. 6]. Although the method treats both rotational and prismatic joints, only the equations for our particular case are presented for the sake of simplicity (Alg. 1) The associated notations are gathered in Tab. 4.1 while the numerical values of the parameters are available in Tab. 4.3. For the two-link planar manipulator considered in this paper and assuming that the CoM of each link is at mid-length, one has :

$$\mathbf{p}_1 = \mathbf{0}, \mathbf{p}_2 = l_1\mathbf{x}_1, \mathbf{p}_{C_1} = \frac{l_1}{2}\mathbf{x}_1, \mathbf{p}_{C_2} = \frac{l_2}{2}\mathbf{x}_2. \tag{4.16}$$

Moreover we recall that for any vector $\mathbf{x}_i$, $\mathbf{x}_i^k$ means the coordinate vector of $\mathbf{x}_i$ in $\mathcal{F}_k$. For instance, in Alg. 1, one has $\mathbf{z}_k^k = [0\ 0\ 1]^\top$ and $\boldsymbol{\omega}_{k-1}^k = \mathbf{R}_{k/k-1}\boldsymbol{\omega}_{k-1}^{k-1}$.

For a fixed base, one has $\boldsymbol{\omega}_0 = \dot{\boldsymbol{\omega}}_0 = \mathbf{v}_0 = \dot{\mathbf{v}}_0 = \mathbf{0}$. But since the arm is mounted under the quadcopter, the velocities and accelerations of the base of the arm (Frame $\mathcal{F}_0$) are non-null. They are therefore set as those of the quadcopter, i.e. :

$$\omega_0^0 = \mathbf{R}_{0/b}\boldsymbol{\omega}_{b/i}^b = \begin{bmatrix} r & p & q \end{bmatrix}^\top \tag{4.17}$$

$$\mathbf{v}_0^0 = \mathbf{R}_{0/b}\mathbf{v}^b = \begin{bmatrix} w & u & v \end{bmatrix}^\top \tag{4.18}$$

$$\dot{\boldsymbol{\omega}}_0^0 = \mathbf{R}_{0/b}{}^b\dot{\boldsymbol{\omega}}_{b/i}^b \tag{4.19}$$

$$\dot{\mathbf{v}}_0^0 = \mathbf{R}_{0/b}\left({}^b\dot{\mathbf{v}}^b + \boldsymbol{\omega}_{b/i}^b \times \mathbf{v}^b - \mathbf{R}_{b/i}\begin{bmatrix} 0 & 0 & g_0 \end{bmatrix}^\top\right) \tag{4.20}$$

These velocities and accelerations are transmitted from one link to another through the outward iterations and result in an additional torque that couples the dynamics of the quadcopter and the manipulator. Moreover, to include the effect of the gravity on the arm links,

Algorithm 1: Iterative Newton-Euler method

- Initialization : $\boldsymbol{\omega}_0, \dot{\boldsymbol{\omega}}_0, \mathbf{v}_0, \dot{\mathbf{v}}_0$
- Outward iterations : $k = 1 \rightarrow 2$

$$\begin{aligned}
\boldsymbol{\omega}_k^k &= \boldsymbol{\omega}_{k-1}^k + \dot{\theta}_k \mathbf{z}_k^k \\
\dot{\boldsymbol{\omega}}_k^k &= \dot{\boldsymbol{\omega}}_{k-1}^k + \ddot{\theta}_k \mathbf{z}_k^k + \boldsymbol{\omega}_{k-1}^k \times \dot{\theta}_k \mathbf{z}_k^k \\
\dot{\mathbf{v}}_k^k &= \dot{\mathbf{v}}_{k-1}^k + \dot{\boldsymbol{\omega}}_{k-1}^k \times \mathbf{p}_k^k + \boldsymbol{\omega}_{k-1}^k \times (\boldsymbol{\omega}_{k-1}^k \times \mathbf{p}_k^k) \\
\dot{\mathbf{v}}_{C_k}^k &= \dot{\mathbf{v}}_k^k + \dot{\boldsymbol{\omega}}_k^k \times \mathbf{p}_{C_k}^k + \boldsymbol{\omega}_k^k \times (\boldsymbol{\omega}_k^k \times \mathbf{p}_{C,k}^k) \\
\mathbf{F}_k^k &= m_k \dot{\mathbf{v}}_{C_k}^k \\
\mathbf{N}_k^k &= \mathbf{I}_{C_k}^k \dot{\boldsymbol{\omega}}_k^k + \boldsymbol{\omega}_k^k \times \mathbf{I}_{C_k}^k \boldsymbol{\omega}_k^k
\end{aligned}$$

- Inward iterations : $k = 2 \rightarrow 1$

$$\begin{aligned}
\mathbf{f}_k^k &= \mathbf{F}_k^k + \mathbf{f}_{k+1}^k \\
\mathbf{n}_k^k &= \mathbf{N}_k^k + \mathbf{n}_{k+1}^k + \mathbf{p}_{C_k}^k \times \mathbf{F}_k^k + \mathbf{p}_{k+1}^k \times \mathbf{f}_{k+1}^k \\
\tau_k &= {\mathbf{n}_k^k}^\top \mathbf{z}_k^k
\end{aligned}$$

the gravity vector in the opposite direction is also added to the linear acceleration $\dot{\mathbf{v}}_0^0$. Finally, the dynamic equations obtained with Alg. 1 combined with initial conditions in Eqs. (4.17) to (4.20) can be expressed as a single vector dynamic equation :

$$\mathbf{M}(\boldsymbol{\Theta})\ddot{\boldsymbol{\Theta}} + \mathbf{C}(\boldsymbol{\Theta}, \dot{\boldsymbol{\Theta}})\dot{\boldsymbol{\Theta}} + \mathbf{g}(\boldsymbol{\Theta}) = \boldsymbol{\tau} + \mathbf{d}_0 \tag{4.21}$$

with $\boldsymbol{\Theta} = [\theta_1 \ \theta_2]^\top$, $\mathbf{M}(\boldsymbol{\Theta})$ the inertia matrix of the manipulator, $\mathbf{C}(\boldsymbol{\Theta}, \dot{\boldsymbol{\Theta}})$ the matrix of Coriolis and centrifugal terms, $\mathbf{g}(\boldsymbol{\Theta})$ the vector of gravity terms, $\boldsymbol{\tau} = [\tau_1 \ \tau_2]^\top$ the actuator torque vector, and $\mathbf{d}_0$ the coupling torque resulting from Eqs. (4.17) to (4.20) without the gravity effect. The vector $\mathbf{d}_0$, depending on $\boldsymbol{\Theta}$ and its derivatives, as well as the states of the quadcopter, can be interpreted as a disturbance to be rejected.

Finally, the forces and moments exerted by the manipulator on the quadcopter, resp. $\mathbf{f}_{arm}$ and $\mathbf{m}_{arm}$, can be calculated. From Alg. 1, the forces and moments applied by link 0 (the quadcopter) on link 1 are $\mathbf{f}_1$ and $\mathbf{n}_1$, respectively. By using Newton's third law, one obtains the forces and moments exerted by the manipulator :

$$\mathbf{f}_{arm}^b = -\mathbf{R}_{b/0}\mathbf{R}_{0/1}\mathbf{f}_1^1 \tag{4.22}$$

$$\mathbf{n}_{arm}^b = -\mathbf{R}_{b/0}\mathbf{R}_{0/1}\mathbf{n}_1^1 \tag{4.23}$$

Tableau 4.1 Notations for the Iterative Newton-Euler method

Notation	Meaning
$\mathbf{R}_{k/k-1}$	Rotation matrix from $\mathcal{F}_{k-1}$ to $\mathcal{F}_k$
$\boldsymbol{\omega}_k$	Angular velocity of link k
$\dot{\boldsymbol{\omega}}_k$	Angular acceleration of link k
$\mathbf{p}_k$	position of $\mathcal{F}_k$ relative to $\mathcal{F}_{k-1}$
$\mathbf{p}_{C_k}$	position of CoM of link k in $\mathcal{F}_k$
$\mathbf{v}_k$	linear velocity of origin of $\mathcal{F}_k$ w.r.t. $\mathcal{F}_{k-1}$
$\dot{\mathbf{v}}_k$	linear acceleration of origin of $\mathcal{F}_k$ w.r.t. $\mathcal{F}_{k-1}$
$\mathbf{v}_{C_k}$	linear velocity of CoM of link k
$\dot{\mathbf{v}}_{C_k}$	linear acceleration of CoM of link k
m_k	mass of link k
l_k	length of link k
$\mathbf{I}_{C_k}$	inertia matrix of link k at CoM
$\mathbf{F}_k$	total force exerted on link k at CoM
$\mathbf{N}_k$	total torque exerted on link k at CoM
$\mathbf{f}_k$	force exerted on link k by link $k-1$
$\mathbf{n}_k$	torque exerted on link k by link $k-1$
τ_k	torque exerted at joint of link k by actuator

4.4.3 Motor Dynamics

The motor and propeller dynamics can also be considered in the simulation model of the system :

$$\begin{aligned} V_{a_j} &= k_e \omega_j + R_a i_{a_j} \\ J_m \dot{\omega}_j &= k_m i_{a_j} - k_d \omega_j^2 \end{aligned} \tag{4.24}$$

with ω_j the rotational speed of the j-th motor, $j \in [1, 2, 3, 4]$, V_{a_j} the input voltage, k_e the back-EMF constant of the motor, R_a the armature resistance, J_m the rotor and propeller inertia, k_m the torque constant, and i_{a_j} the armature current. The viscous damping and the armature impedance of the motors are considered negligible.

4.4.4 State equation

The modeling of the AM is now complete for both numerical simulations and controller design without being overly simplified. Equations (4.2) to (4.5) and Eq. (4.21) are rewritten as a state equation :

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{4.25}$$

where state and input vectors, resp. $\mathbf{x} \in \mathbb{R}^{16}$ and $\mathbf{u} \in \mathbb{R}^6$, are defined as :

$$\mathbf{x} = \begin{bmatrix} \mathbf{p}^{i\top} & \mathbf{\Phi}^\top & \mathbf{v}^{b\top} & \boldsymbol{\omega}_{b/i}^{b\top} & \mathbf{\Theta}^\top & \dot{\mathbf{\Theta}}^\top \end{bmatrix}^\top \tag{4.26}$$

$$\mathbf{u} = \begin{bmatrix} \omega_1 & \omega_2 & \omega_3 & \omega_4 & \tau_1 & \tau_2 \end{bmatrix}^\top \tag{4.27}$$

The motor dynamics is not included in the state space model because it is assumed to be sufficiently fast, and therefore negligible in the control law synthesis process that will be led in the next two sections.

4.5 Equilibrium and Linearization of the Dynamic Model

Since the controller is to be scheduled with respect to the robotic arm joint angles θ_1 and θ_2, a family of linear models of the drone at different equilibrium points must be obtained. These linear models will be used in the multi-model synthesis of the scheduled controller.

4.5.1 Equilibrium Definition

The quadcopter equilibrium condition is chosen as hovering flight, while the manipulator is in any fixed position $\mathbf{\Theta}_e = [\theta_{1,e}\ \theta_{2,e}]^\top$. At equilibrium, one has $\dot{\mathbf{x}} \equiv \mathbf{0}$, and the system to be solved is then :

$$\mathbf{0} = \mathbf{f}(\mathbf{x}_e, \mathbf{u}_e) \tag{4.28}$$

Since any position $\mathbf{p}$ or heading ψ can be chosen for the hovering flight equilibrium condition, all the state variables but the joint angles in Eq. (4.26) are set to zero without loss of generality, i.e. :

$$\mathbf{x}_e = \begin{bmatrix} \mathbf{0}_{1\times 12} & \mathbf{\Theta}_e^\top & \mathbf{0}_{1\times 2} \end{bmatrix}^\top \tag{4.29}$$

From Eq. (4.28), the system to be solved boils down to :

$$\begin{aligned} k_t\left(\omega_{1,e}^2 + \omega_{2,e}^2 + \omega_{3,e}^2 + \omega_{4,e}^2\right) &= (m + m_1 + m_2)g_0 \\ dk_t\left(\omega_{2,e}^2 - \omega_{4,e}^2\right) &= 0 \\ dk_t\left(\omega_{1,e}^2 - \omega_{3,e}^2\right) &= \tau_{1,e} \\ k_d\left(\omega_{1,e}^2 - \omega_{2,e}^2 + \omega_{3,e}^2 - \omega_{4,e}^2\right) &= 0 \end{aligned} \tag{4.30}$$

where the actuator torques at equilibrium are given by :

$$\tau_{1,e} = \left(\frac{m_1}{2} + m_2\right) g_0 l_1 \sin\theta_{1,e} + \frac{m_2 g_0 l_2}{2} \sin(\theta_{1,e} + \theta_{2,e})$$
$$\tau_{2,e} = \frac{m_2 g_0 l_2}{2} \sin(\theta_{1,e} + \theta_{2,e})$$

By solving (4.30), the rotational speeds of each motor can be expressed as functions of $\theta_{1,e}$ and $\theta_{2,e}$:

$$\omega_{1,e}^2 = \frac{m_T g_0}{4k_t} + \frac{\tau_{1,e}}{2dk_t} \tag{4.31}$$
$$\omega_{2,e}^2 = \omega_{4,e}^2 = \frac{m_T g_0}{4k_t} \tag{4.32}$$
$$\omega_{3,e}^2 = \frac{m_T g_0}{4k_t} - \frac{\tau_{1,e}}{2dk_t} \tag{4.33}$$

with $m_T = m + m_1 + m_2$ the AM total mass. From Eqs. (4.31) to (4.33), the effect of the manipulator at equilibrium can be seen as a mass added to the quadcopter and a supplementary moment around the $\mathbf{y}_b$-axis. Motors 2 and 4 always have equal equilibrium speed, and motors 1 and 3 speed up or slow down by an equal value to compensate for the robotic arm. For instance, this imposes the following design condition $m_T g_0 d \geq 2\tau_{1,e}$. Finally, using the motor dynamics as in Eq. (4.24), it is now possible to calculate the equilibrium voltage needed for each motor j :

$$V_{a_j,e} = k_e \omega_{j,e} + \frac{R_a k_d}{k_m} \omega_{j,e}^2 \tag{4.34}$$

4.5.2 Linearization

The model of the AM is linearized around the equilibrium condition $(\mathbf{x}_e, \mathbf{u}_e)$ presented in the previous subsection using standard linearization techniques. This yields the family of linearized systems parameterized with the manipulator configuration $\boldsymbol{\Theta}_e$:

$$\dot{\boldsymbol{\delta}\mathbf{x}} = \mathbf{A}\left(\boldsymbol{\Theta}_e\right) \boldsymbol{\delta}\mathbf{x} + \mathbf{B}\left(\boldsymbol{\Theta}_e\right) \boldsymbol{\delta}\mathbf{u} \tag{4.35}$$

where $\boldsymbol{\delta}\mathbf{x} = \mathbf{x} - \mathbf{x}_e$ and $\boldsymbol{\delta}\mathbf{u} = \mathbf{u} - \mathbf{u}_e$ denote the deviations of the state and input vectors from the equilibrium $(\mathbf{x}_e, \mathbf{u}_e)$. Depending on the joint angle values $\boldsymbol{\Theta}_e$, the matrix elements of $\mathbf{A}$ and $\mathbf{B}$ will change, and thus their dynamic characteristics, such as eigenvalues. Instead of designing a controller robust to any possible value of $\boldsymbol{\Theta}_e$, it is preferably sought to schedule the controller with $\boldsymbol{\Theta}_e$ to better adapt and maintain a consistent level of performance.

As classically done with multicopter dynamics, the four motor control inputs $\delta\omega_j$ are re-

mapped into a vertical force δF_z and three moments $\delta M_{\{x,y,z\}}$ with the ulterior motive of decoupling the dynamics into four modes. The linearized transformation is given by :

$$\begin{bmatrix} \delta F_z \\ \delta M_x \\ \delta M_y \\ \delta M_z \end{bmatrix} = \begin{bmatrix} -k_t & -k_t & -k_t & -k_t \\ 0 & dk_t & 0 & -dk_t \\ dk_t & 0 & dk_t & 0 \\ k_d & -k_d & k_d & -k_d \end{bmatrix} \begin{bmatrix} 2\omega_{1,e}\delta\omega_1 \\ 2\omega_{2,e}\delta\omega_2 \\ 2\omega_{3,e}\delta\omega_3 \\ 2\omega_{4,e}\delta\omega_4 \end{bmatrix} \tag{4.36}$$

Four subsystems can then be extracted from (4.35) and (4.36) for controller design purpose. These subsystems correspond to the vertical (p_D), lateral (ϕ), longitudinal (θ), and directional (ψ) modes with their respective state and input vectors being :

$$\begin{aligned} \boldsymbol{\delta}\mathbf{x}_{p_D} &= \begin{bmatrix} \delta w & \delta p_D \end{bmatrix}^\top & \delta u_{p_D} &= \delta F_z \\ \boldsymbol{\delta}\mathbf{x}_{\phi} &= \begin{bmatrix} \delta p & \delta \phi \end{bmatrix}^\top & \delta u_{\phi} &= \delta M_x \\ \boldsymbol{\delta}\mathbf{x}_{\theta} &= \begin{bmatrix} \delta q & \delta \theta \end{bmatrix}^\top & \delta u_{\theta} &= \delta M_y \\ \boldsymbol{\delta}\mathbf{x}_{\psi} &= \begin{bmatrix} \delta r & \delta \psi \end{bmatrix}^\top & \delta u_{\psi} &= \delta M_z \end{aligned}$$

Each subsystem is a second order LTI system with its own $\mathbf{A}_m$ and $\mathbf{B}_m$ matrices, $m = \{p_D, \phi, \theta, \psi\}$, still depending on $\boldsymbol{\Theta}_e$. This changes the overall control problem from designing one controller for a single MIMO system to designing four controllers for each subsystem. Two additional inputs, namely the torques applied to each arm joint, are considered as disturbances in the context of controller design. As opposed to a lone quadcopter, where each of these modes are entirely uncoupled, the modes here are coupled due to the presence of the manipulator. For the purpose of controller design, the four flying modes are still treated as uncoupled.

4.6 Control Strategy

In this section, the AM controller structure, synthesis framework and algorithm initialization starting points are presented.

4.6.1 Manipulator controller structure

The controller used for the manipulator is a joint space inverse dynamics controller [150], where the mass, Coriolis and centrifugal, and gravity matrices are supposed to be measurable

at any time, i.e. :

$$\boldsymbol{\tau} = \mathbf{M}(\boldsymbol{\Theta}) \left[\ddot{\boldsymbol{\Theta}}_c + \mathbf{K}_d(\dot{\boldsymbol{\Theta}}_c - \dot{\boldsymbol{\Theta}}) + \mathbf{K}_p(\boldsymbol{\Theta}_c - \boldsymbol{\Theta}) \right] + \mathbf{C}(\dot{\boldsymbol{\Theta}}, \boldsymbol{\Theta})\dot{\boldsymbol{\Theta}} + \mathbf{g}(\boldsymbol{\Theta}) \tag{4.37}$$

where $\mathbf{K}_p \in \mathbb{R}^{2\times2}$ and $\mathbf{K}_d \in \mathbb{R}^{2\times2}$ are the PD controller diagonal matrices tuned to obtain a satisfying dynamic behavior of the arm, and $\boldsymbol{\Theta}_c, \dot{\boldsymbol{\Theta}}_c, \ddot{\boldsymbol{\Theta}}_c$ are the commanded angular positions, velocities and accelerations. This controller has been implemented as given by Eq. (4.37) and will not be detailed here since it is not the focus of this paper.

4.6.2 Controller structure

A state feedback controller with integral action on the regulated variables is chosen to stabilize the quadcopter. The advantages of this controller include a simple structure and implementation, while still reaching the desired performances. The controller is focused on stabilizing the quadcopter around hovering flight and is scheduled with θ_1 and θ_2 to deal with the arm motions. The quadcopter controller consists of two nested loops. The inner loop controls the Euler angles $\boldsymbol{\Phi}$ and the altitude $h = -p_D$; the integral actions ensure a zero steady-state error and allows to reject the perturbations due to the manipulator, or other factors, i.e, uncertainties and external disturbances. The inner loop equations are :

$$\begin{aligned}
\delta u_{p_D} &= -\mathbf{K}_{p_D}(\boldsymbol{\Theta})\boldsymbol{\delta}\mathbf{x}_{p_D} + K_{i,p_D}(\boldsymbol{\Theta})\textstyle\int(\delta p_{D,c} - \delta p_D) \\
\delta u_{\phi} &= -\mathbf{K}_{\phi}(\boldsymbol{\Theta})\boldsymbol{\delta}\mathbf{x}_{\phi} + K_{i,\phi}(\boldsymbol{\Theta}) \textstyle\int(\delta\phi_c - \delta\phi) \\
\delta u_{\theta} &= -\mathbf{K}_{\theta}(\boldsymbol{\Theta})\boldsymbol{\delta}\mathbf{x}_{\theta} + K_{i,\theta}(\boldsymbol{\Theta}) \textstyle\int(\delta\theta_c - \delta\theta) \\
\delta u_{\psi} &= -\mathbf{K}_{\psi}(\boldsymbol{\Theta})\boldsymbol{\delta}\mathbf{x}_{\psi} + K_{i,\psi}(\boldsymbol{\Theta}) \textstyle\int(\delta\psi_c - \delta\psi)
\end{aligned} \tag{4.38}$$

where $\mathbf{K}_m \in \mathbb{R}^{1\times2}$ and $K_{i,m} \in \mathbb{R}$, $m \in \{p_D, \phi, \theta, \psi\}$, are respectively matrices and scalars whose elements are scheduled with respect to the joint angles $\boldsymbol{\Theta}$. Figure 4.3 presents the structure of the inner loop of the controller, in which $\boldsymbol{\delta}\mathbf{x}_m$ is the state vector of each subsystem, δu_m the input, δr_m the regulated variable that goes in the integral action, and δr_c the corresponding commanded value.

The outer loop is an output feedback loop, controlling the position (p_N, p_E) in the horizontal plane as :

$$\begin{aligned}
\delta x_u &= K_{x,2}(\delta p_{N,c} - \delta p_N) - K_{x,1}\delta u \\
\delta y_u &= K_{y,2}(\delta p_{E,c} - \delta p_E) - K_{y,1}\delta v
\end{aligned} \tag{4.39}$$

This outer loop is built around the inner loops controlling ϕ and θ. The commands $(\delta x_u, \delta y_u)$

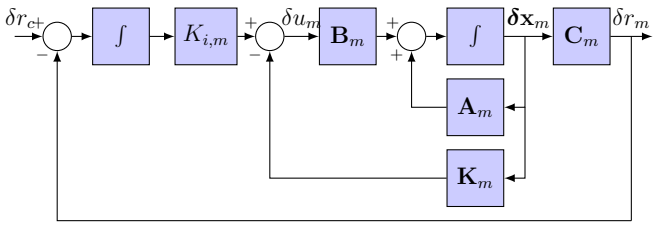

Figure 4.3 Inner loop controller structure

are translated into demands in$(\delta\phi_c, \delta\theta_c)$ through a rotation of angle ψ. The gains are not scheduled since the manipulator do not affect the quadcopter translation dynamics in the linear models.

4.6.3 Structured $\mathcal{H}_\infty$ synthesis

The gain-scheduled controller (4.38) is tuned within the structured $\mathcal{H}_\infty$ synthesis framework [104] with the MATLAB Robust Control Toolbox function `systune` [100]. The capabilities of the `systune` function allow to perform multi-model synthesis with a predefined controller architecture and self-scheduling of the controller gains, i.e., the scheduling of the gains is chosen *a priori*.

In order to tune the gain-scheduled controller, a grid of 5×5 operating points of the scheduling variables θ_1 and θ_2 is created in the range of $\left[-\frac{\pi}{4}, \frac{\pi}{4}\right] \times \left[-\frac{\pi}{4}, \frac{\pi}{4}\right]$ rad. This leads to a family of 25 linearized models obtained with Eq. (4.35). Finally, the `systune` function accepts a wide variety of design requirements of different natures ; they are gathered in Tab. 4.2.

Tableau 4.2 Design requirements

	Requirement	**Objective**
1	Tracking (ϕ, θ)	2% settling time $\approx 2\,\mathrm{s}$
		steady-state error $< 0.01\%$
	Tracking (ψ, p_D)	2% settling time $\approx 4\,\mathrm{s}$
		steady-state error $< 0.01\%$
2	Robustness margins	gain margin > 5 dB
		phase margin $> 25°$
3	Closed-loop poles	damping ratio $\zeta > 0.4$

In addition to the grid of linear models and the performance requirements, the synthesis algorithm also requires initial conditions for the controller gains to start the optimization, i.e., a set of gains that stabilizes the closed-loop system for a nominal case and approaches the desired performances. These values are calculated using an eigenstructure assignment method

[151], so the closed-loop poles are chosen as to approximately reach the desired temporal characteristics. These initialization gains are calculated for the case $(\theta_{1,e}, \theta_{2,e}) = (0, 0)$, which corresponds to a vertical arm underneath the drone and is the central operating point in the grid.

4.6.4 Gain-scheduling surfaces

Each gain that must be scheduled is modeled as a tuning surface function of the robotic arm joint angles θ_1 and θ_2. Different surface shapes were tested, including linear, coupled linear, and quadratic. Only quadratic surfaces were able to stabilize the AM over the grid and to obtain the desired tuning objectives, therefore making it the minimal degree usable for this specific configuration of AM. A general formulation for each gain is given by

$$K = K_0 + K_1\theta_1 + K_2\theta_2 + K_3\theta_1\theta_2 + K_4\theta_1^2 + K_5\theta_2^2 \tag{4.40}$$

where the coefficients K_j, $j \in [0..5]$ of the surfaces are different for each gain, meaning that for the controller developed in this paper, 12 different surfaces are obtained.

Using the gains obtained by eigenstructure assignment as initial values of the gains K_0, the structured $\mathcal{H}_\infty$ synthesis tunes the values of the 72 coefficients (12 surfaces each having 6 coefficients) by simultaneously trying to reach the desired tuning goals presented 4.2 for the 25 linearized models. With the gains obtained, the time responses of the controlled linear models are identical for each operating point, confirming that the structured $\mathcal{H}_\infty$ synthesis is able to tune surfaces in a way to obtain the same closed-loop poles at each scheduling points on the grid of robotic arm joint variables. The outer loop (Eq. 4.39) is simply tuned using the previously mentioned eigenstructure assignment method. Now that the scheduled controller is satisfactorily tuned, it can be implemented on the non-linear system for further simulations.

4.7 Simulations of the Closed-Loop Aerial Manipulator

The previously introduced controller (Eqs. (4.38) and (4.39)) is now applied to the non-linear simulation model of the AM using the equations presented in Section 4.4.

4.7.1 Model and controller parameters

The quadcopter used in this paper is a modified AscTec Pelican where the physical parameter values were taken from [133]. The robotic arm used is a 3D printed arm, developed in our

laboratory, with the parameters taken from a CAD model. Finally, the motor parameters were chosen as to be realistic for the system at hand. The values of each parameters are given in Tab. 4.3. The use of accurate values for all parameters is necessary in order to tune the controller as to attain the desired performances, but as shown later, a certain variance between the tuning values and the actual values can be tolerated.

Tableau 4.3 Aerial manipulator parameter values

Symbol	Quantity	Value
m	Drone mass	$1.27\,\mathrm{kg}$
I_{xx}	Drone inertia ($\mathbf{x}$)	$0.043\,\mathrm{kg}\cdot\mathrm{m}^2$
I_{yy}	Drone inertia ($\mathbf{y}$)	$0.043\,\mathrm{kg}\cdot\mathrm{m}^2$
I_{zz}	Drone inertia ($\mathbf{z}$)	$0.070\,\mathrm{kg}\cdot\mathrm{m}^2$
d	Rotor distance	$0.211\,\mathrm{m}$
k_d	Drag parameter	7.5×10^{-7}
k_t	Lift parameter	$3.0\times10{-}5$
m_1	Mass, link 1	$0.08\,\mathrm{kg}$
m_2	Mass, link 2	$0.01\,\mathrm{kg}$
l_1	Length, link 1	$0.125\,\mathrm{m}$
l_2	Length, link 2	$0.522\,\mathrm{m}$
$I_{zz,1}$	Inertia, link 1 (around $\mathbf{z}_1$)	$1.04\times10^{-4}\,\mathrm{kg}\cdot\mathrm{m}^2$
$I_{zz,2}$	Inertia, link 2 (around $\mathbf{z}_2$)	$2.27\times10^{-4}\,\mathrm{kg}\cdot\mathrm{m}^2$
k_e	Back EMF	7.8×10^{-3}
R_a	Resistance	$0.50\,\Omega$
J_m	Motor and propeller inertia	$7.8\times10^{-3}\,\mathrm{kg}\cdot\mathrm{m}^2$
k_m	Torque constant	3.7×10^{-3}

Table 4.4 contains the values of the gain coefficients obtained with the structured $\mathcal{H}_\infty$ synthesis. While the magnitude of the values of K_0 may be surprising, they relate the error on an angle or position to motor rotational speed acoording to the inverse of the mapping presented in Eq. (4.36). This results in high value of the gains, but the commands still remain within the motor limitations. As for the gains of the outer loop (Eq. (4.39)), they are given in Tab. 4.5.

Tableau 4.4 Gain-scheduled controller coefficient values

Gain	Value					
Variable	K_0	K_1	K_2	K_3	K_4	K_5
$\mathbf{K}_{p_D}(\delta w)$	29087	−74	−13	294	−713	−108
$\mathbf{K}_{p_D}(\delta p_d)$	141750	−89	9	471	−3310	−524
K_{i,p_D}	180210	129	718	−39	−3313	−2102
$\mathbf{K}_{\phi}(\delta p)$	3709	−6	0	−43	−6	93
$\mathbf{K}_{\phi}(\delta \phi)$	20315	−31	0	−233	−48	455
$K_{i,\phi}$	31657	2	5	−310	−44	976
$\mathbf{K}_{\theta}(\delta q)$	3714	2	1	0	−59	−1
$\mathbf{K}_{\theta}(\delta \theta)$	20273	10	4	−1	−294	−7
$K_{i,\theta}$	31990	65	28	−2	−814	−12
$\mathbf{K}_{\psi}(\delta r)$	49440	−19	100	13	2779	8
$\mathbf{K}_{\psi}(\delta \psi)$	266710	7	4	0	−308	−3
$K_{i,\psi}$	270160	−55	−47	−322	131	−1

Tableau 4.5 Outer loop gain values

Gain	Value	Gain	Value
$K_{x,1}$	−0.0793	$K_{x,2}$	−0.018
$K_{y,1}$	0.08	$K_{y,2}$	0.019

4.7.2 Applying the controller on the nominal non-linear system

The behavior of the proposed controller on the non-linear dynamics of the AM is studied to confirm the ability of the controller to reach the desired performances. The following scenario has been chosen to show the response of each mode and the effect of the manipulator motion on the entire system : at $t = 0\,\text{s}$, a commanded horizontal position $p_{N,c} = 0.5\,\text{m}$ and $p_{E,c} = 0\,\text{m}$, followed at $t = 10\,\text{s}$ by a demand in altitude $h = -p_{D,c} = 0.5\,\text{m}$. As for the manipulator motion, we consider two cases :

— Case 1 : $\theta_1 = \theta_2 = \frac{\pi}{4}\sin(0.25t)$

— Case 2 : $\theta_1 = \theta_2 = \frac{\pi}{4}\sin(t)$

The joint angle maximum amplitudes are tested with a slow motion (Case 1) and a faster one (Case 2). In this scenario, the physical parameters of the quadcopter and the manipulator are the nominal values presented in Table 4.3.

Figure 4.4 shows the temporal responses of the quadcopter CoM position in the inertial frame

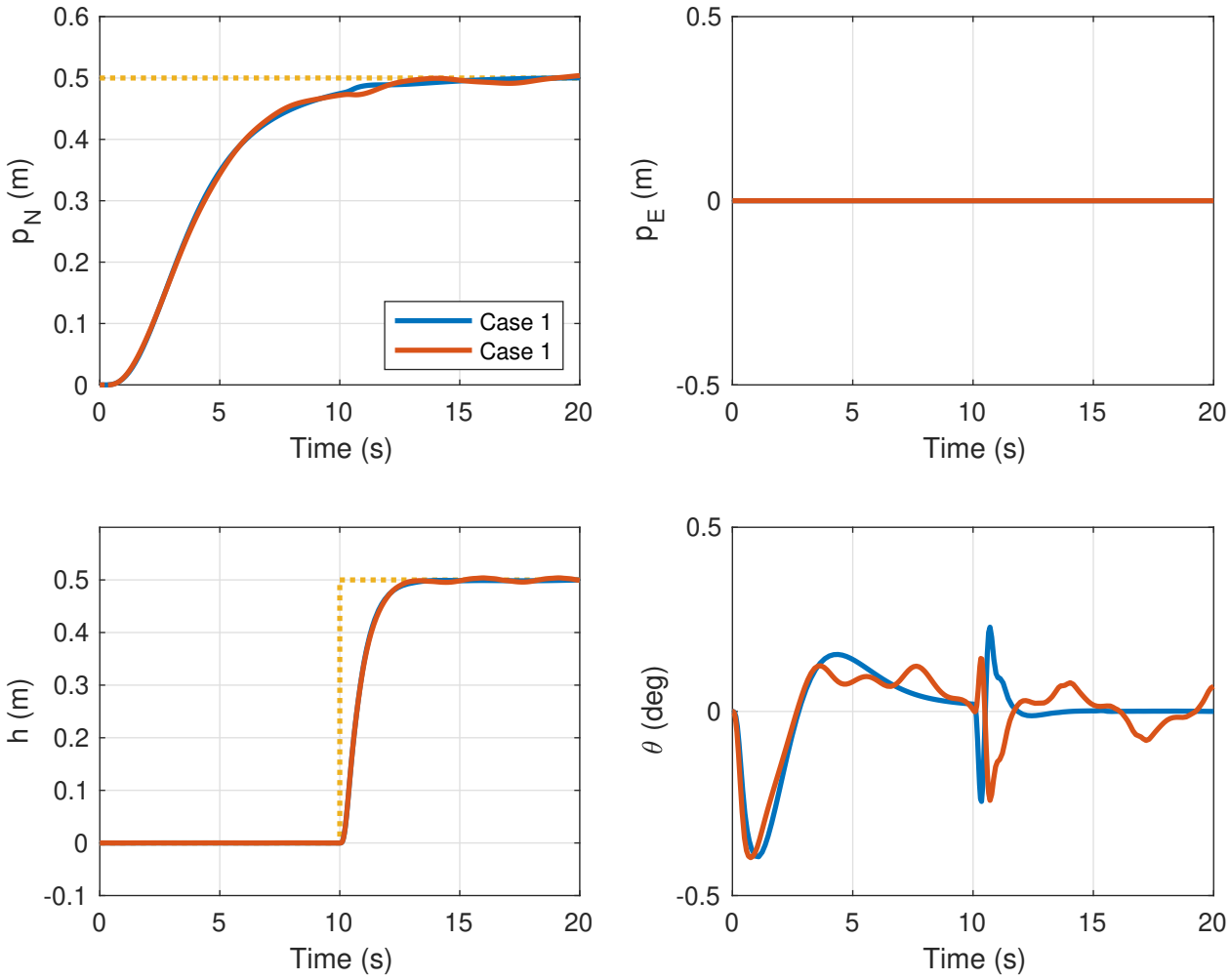

Figure 4.4 Quadcopter CoM position in frame $\mathcal{F}_i$ and pitch angle θ (Nominal parameters)

$\mathcal{F}_i$. In both cases, the quadcopter reaches its steady state position in approximately 7 s in p_N, and is able to keep it within an error of 0.01 m of the steady state value, while having the arm always in movement. The p_E time response is not affected by the movement of the arm as the arm motion remains in the vertical plane. Before $t = 10$ s, the altitude h is practically not affected and remains constant, while after $t = 10$ s, the steady state is reached in 2 s and kept notwithstanding the disturbances of the moving arm. While the movement of the arm creates a force in $\mathbf{z}_i$, it is negligible when compared to the quadcopter mass, which is the biggest influence in these modes. The vertical mode is notably faster than the longitudinal and lateral modes as designed during the controller synthesis. If the Euler angles ϕ and ψ are unaffected, the pitch angle θ remains close to 0, meaning that during the whole motion the quadcopter is almost horizontal. The oscillations of the manipulator can be perceived in the

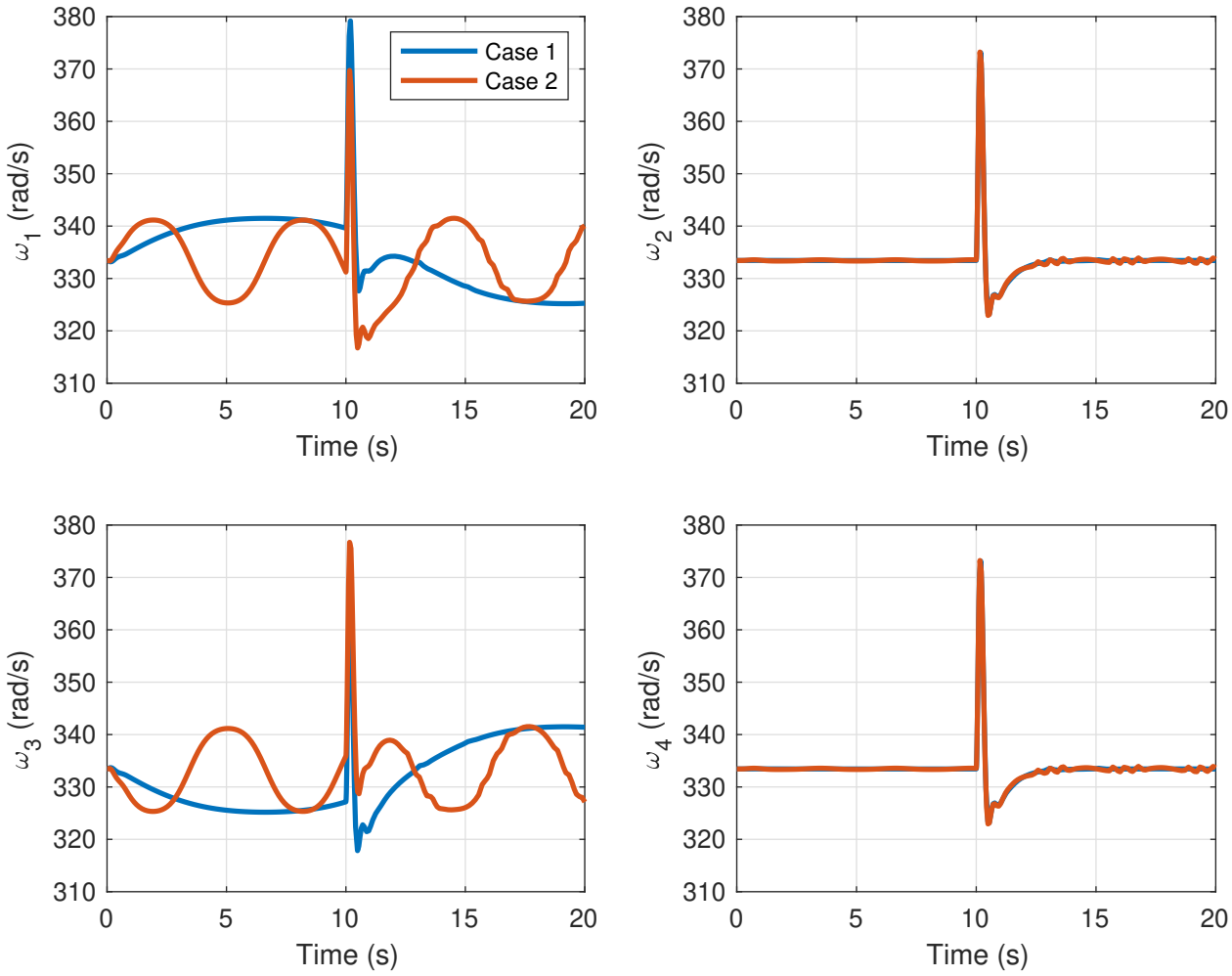

Figure 4.5 Motor rotational speeds (Nominal parameters)

responses in p_N, p_D and θ but they are clearly visible on the rotational speeds of the motors (Fig. 4.5). In both cases, the arm oscillations of 0.25 and 1 rad/s are present on motors 1 and 3 with the same frequency, while the motors 2 and 4 are unaffected. All motors experience the same spike at 10 s due to the change of altitude.

4.7.3 Comparison between non-scheduled and scheduled controllers

To show the effectiveness of the gain-scheduled controller, its performances are compared to a robust version of the controller whose gains are no longer scheduled with $\boldsymbol{\Theta}$. This non-scheduled controller is still tuned within the structured $\mathcal{H}_\infty$ framework around the nominal case $(\theta_{1,e}, \theta_{2,e}) = (0, 0)$. Moreover the motor rotational speeds are no longer adjusted with the arm position. As the non-scheduled controller could not stabilize the system under the pre-

vious scenario, the following one is considered. The position in the horizontal plane (p_N, p_E) must remain at the origin while the first joint angle θ_1 moves from 0 to $\pi/4$ at $t = 0\,\text{s}$. At $t = 8\,\text{s}$, the second joint angle θ_2 moves from 0 to $\pi/4$ while at $t = 10\,\text{s}$, the AM must reach an altitude of 0.5 s. Figure 4.6 exhibits the differences in the behaviours between the scheduled and the non-scheduled controllers. In the first seconds, the AM with the scheduled controller remains stationary, while the other moves away from the origin in p_N as the first joint reaches $\pi/4$. Both AMs manage to reach the altitude of 0.5 m simultaneoulsy, but because of the movement of the second joint, the AM with the non-scheduled controller still takes time to return to the origin. This clearly shows the advantages of the gain-scheduled controller over its non-scheduled version.

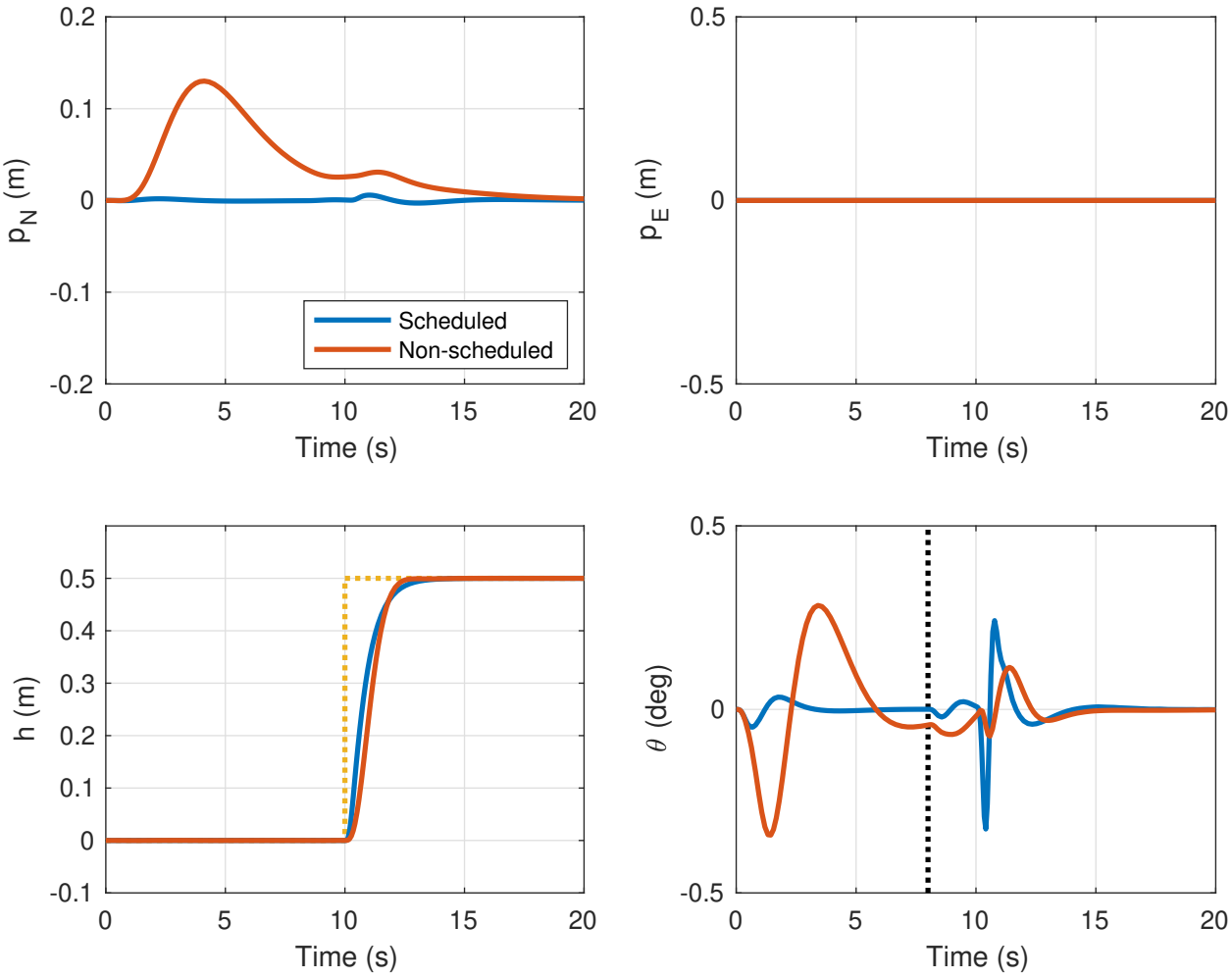

Figure 4.6 Time responses of x^i and z^i position for the proposed controller and a nominal non-scheduled controller

4.7.4 Monte Carlo simulations with physical parameters uncertainties

While the controller is tuned using a nominal set of physical parameter values, these values are rarely equal to those of a real-life system. In order to assess the controller's ability to deal with these uncertainties, the AM nominal parameters are disturbed within a range of $\pm 10\%$ around their nominal values (Tab. 4.3). A Monte Carlo simulation is used ($N = 1000$) with uniform random sampling in the intervals. Figure 4.7 illustrates the results of the Monte Carlo simulation for the same commands as those presented in Section 4.7.2. The time responses in p_N and θ show a certain variance around the nominal case, but the stability and the performance are still adequate. As for the altitude h , it remains largely unaffected by the variance of the parameters. Thus, the scheduled controller exhibits a natural robustness to uncertainties of 10% on the nominal parameters, which is enough to compensate for the difference between nominal values used in tuning and the actual values in the future experimental testbed.

4.8 Conclusion

This paper introduced a novel gain-scheduled state feedback controller for a quadcopter equipped with a 2-DoF arm, tuned using the structured $\mathcal{H}_\infty$ synthesis framework. The dynamic model of the system was obtained by combining the dynamics of a quadcopter with the arm effects on the quadcopter using Newton-Euler recursive method. The equations were linearized at different operating points of the robotic arm for controller synthesis purposes. The proposed controller is a two loops state feedback with integral action on the Euler angles and the altitude. Each gain of the inner loop is gain-scheduled using quadratic surfaces with respect to the robotic arm joint angles. It is shown in simulation that the closed-loop nonlinear system reaches the desired temporal performances for the nominal system, while being robust to uncertainties in the values of the physical parameters. The main advantage of the proposed controller is its simplicity while maintaining adequate dynamic performances.

Future works will explore other possible scheduling surfaces, and scheduling with respect to the mass of a grasped object if known in advance. Experimental testing will allow for validation of the simulation results. The controller will be implemented in a static object picking tasks context.

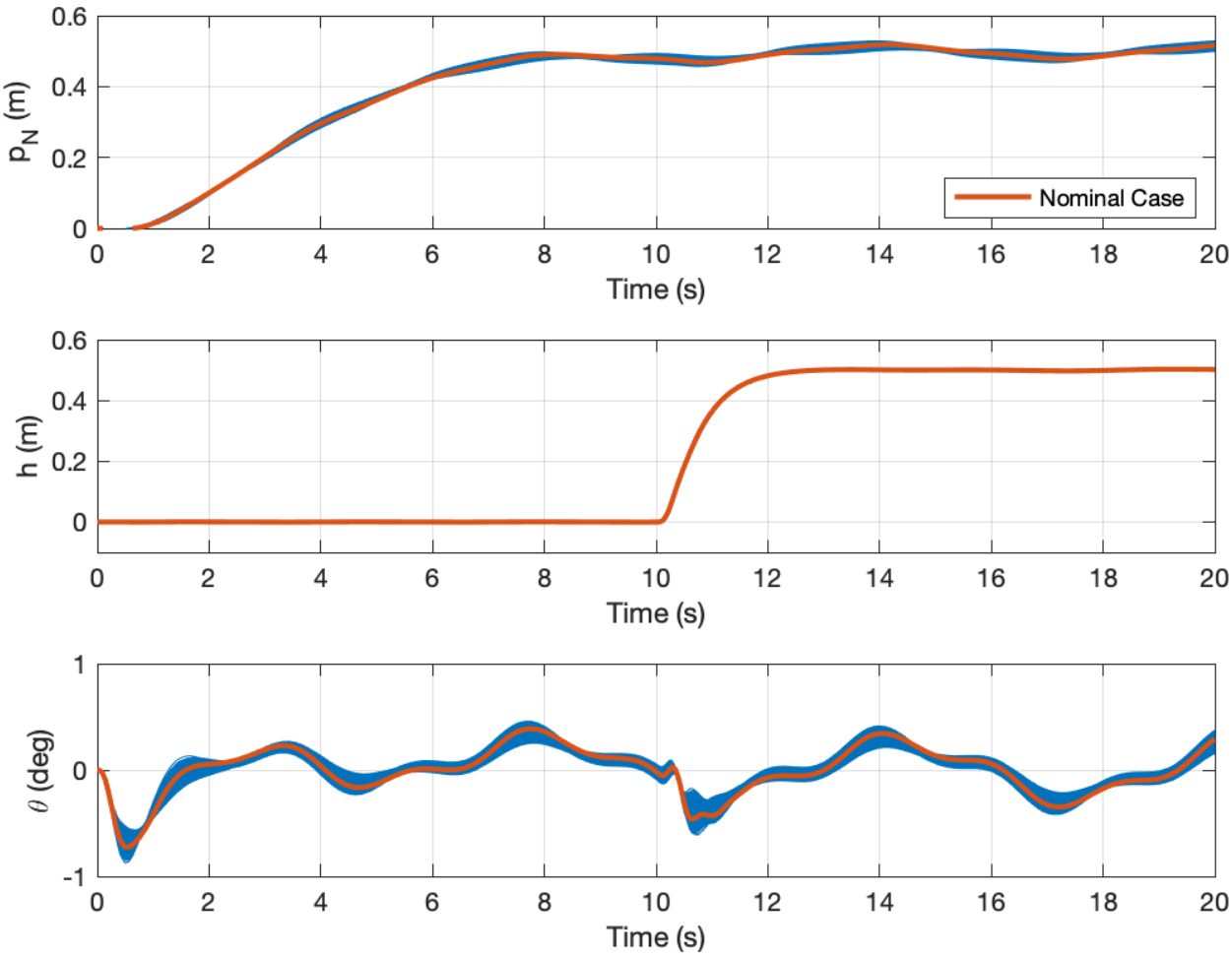

Figure 4.7 Monte Carlo simulations for 10% uncertainties on nominal parameters

CHAPITRE 5 ARTICLE 3 : SELECTION OF UNMANNED AERIAL MANIPULATOR CONFIGURATIONS FOR PICKING TASKS IN CLUTTERED ENVIRONMENTS USING POINT CLOUD BASED DEEP LEARNING

Charles Coulombe, David Saussié, Sofiane Achiche

Soumis à Journal of Intelligent & Robotic Systems, Mai 2021

Ce chapitre répond au sous-objectif **SO3** en introduisant une méthode supportant la conception d'un sous-système effectuant la sélection optimale d'un UAM parmi une flotte associé à une tâche dans un environnement encombré. Cette méthode se base sur l'entraînement d'un réseau de neurones par apprentissage profond à l'aide d'une base de données créée pour cette application précise. Pour obtenir cette base de données, une méthode à base de planification de trajectoire est développée. Le prédicteur utilise une représentation de l'environnement de type nuage de point qui est traitée directement par le réseau de neurones. Finalement, un concepteur utilisant cette méthode ne doit choisir que la flotte utilisée et le critère d'optimisation désiré. Il ne reste alors que d'entraîner le prédicteur à nouveau avec la base de données étiquetées selon la nouvelle flotte.

5.1 Abstract

Unmanned aerial manipulators (UAMs) are flying robots that are able to interact with their environments using an attached manipulator. They are most commonly developed to accomplish object picking tasks, and the planning of these tasks are beginning to figure prominently in the literature. Different UAM configurations have different task solving capabilities, meaning that in a fleet of various UAMs, a selection must be made in order to choose one that is better suited for a specific picking task. Moreover, in a situation where multiple picking tasks should be accomplished simultaneously, such a selection method could be used to efficiently dispatch UAMs to their tasks. A deep learning based method for selecting an optimal UAM from a fleet to perform a static picking task in a cluttered environment is introduced. This method is based on a description of the task and the scene in which it is accomplished, respectively in the form of an orientation quaternion and a point cloud. This information is used as input for a deep convolution neural network classifier that predicts the optimal UAM capable of carrying out the task. To train this classifier, a dataset of 20,000 items is created using randomly generated scenes and objectives, which are processed by a path planning

based method to obtain the reference classes. The introduced deep learning based method is shown to be able to select an adequate UAM configuration for the task at hand with a 71.2% classification accuracy. An experimental testbed confirms the suitability of the method by using 3D scans of real scenes in the trained network and obtains a classification accuracy similar to that of the test part of the simulated dataset. The introduced method could be used in a fleet dispatch system for multiple simultaneous picking tasks, or even in a UAM design context.

Keywords : Deep Learning ; Aerial Manipulator ; Point Cloud ; Selection ; Picking Tasks

5.2 Introduction

In the last few years, quadcopters and other multicopters have become more common whether in academic research, commercial applications, or simply as a hobby. Since this rise in popularity, researchers have been working to increase their interaction capabilities, identified as one of their major weaknesses [6]. Unmanned aerial manipulators (UAMs) were developed as a direct solution to this weakness. They are a class of robots consisting of a manipulation device attached to an unmanned aerial vehicle (UAV). Most commonly, they are made of an unmanned multicopter equipped with one or more robotic arms that can have different configurations, notably by changing the number of degrees of freedom (DoFs) [18, 26, 152]. UAMs offer the possibility to accomplish a variety of tasks such as structural visual inspection [125], object picking [50] or even complex assembly [105]. They are also useful for their potential use in different hazardous environments [127]. Tasks that can be performed by UAMs were categorized in a task taxonomy [153], where object picking tasks were identified as the most commonly found in literature, and therefore are the focus of this work.

Recent domain reviews [6, 15] have highlighted the fact that a large portion of the published studies on UAMs focuses either on stability control [35, 37] or on trajectory planning for different tasks, such as drawer opening [27] or cooperation between multi-UAMs [154]. Limited works detail path or trajectory planning strategies specific to picking tasks with UAMs. One such research avenue uses multiple points to guide a UAM in a scene devoid of obstacles [116]. This approach is based on the configuration of the planar manipulator and tends to keep the arm leveled as to keep the camera looking at the objective. Similarly, an object is picked up using Image-Based Visual Servoing (IBVS), in which the camera is equipped with a fish-eye lens [50]. Path planning is solved conjointly with dynamic controls in an obstacle-free environment. A weight matrix is used in the inverse kinematics in order not to over stretch the arm, and the movement of the arm is limited to the vicinity of the horizontal plane [50]. The task of picking straight-standing objects, as defined in these previous works, limits the use

of the complete range of motion of the UAM. A trajectory planning strategy for grasping a moving target on the ground from the air is to use the multicopter to fit the trajectory of the target, while relative inverse kinematics is used to adjust the relative pose of the target to the manipulator [110]. In this case, this effectively decouples the kinematics of the drone and the arm, while allowing the arm to remain mostly horizontal. Another work uses a larger range of motion, in which a trajectory is planned for two UAMs cooperating together to move a single large target, by considering the combined dynamics and kinematics of both UAMs and target [154]. These aforementioned studies accomplish their picking tasks in obstacle-free environments and do not require the use of the complete arm's range of motion and the combined kinematics of the system, as it would be necessary in cluttered scenes, and often done with fixed-base robotic arm [155]. In these environments, a UAM must deal with obstacles at close range and avoid collisions. Depending on the obstacle positions relative to the task target, specific UAM configurations might be better suited than others.

As of today's literature and as reported in [15], few works also consider high-level problematic related to UAM guidance subsystems, including mission planning and decision making, as defined in [62]. The focus is on integrating vision for picking task in obstacle-free environments [50], or stabilizing the drone during picking, [68], which are essentially lower-level subsystems. However, to the best of the authors' knowledge, no significant research has been conducted on UAMs performing picking tasks in cluttered environments, and therefore neither on UAM selection for these tasks. High-level task planning has been identified as one of the challenges for robotics in general, especially for its use in robotic bin and shelf picking, as well as in robotic disaster mitigation and recovery [4], both cases in which UAMs could be valuable assets. UAVs are already being used following disasters to assess and obtain critical information about the situation at hand [70]. It would therefore be interesting to see UAMs being used in such contexts where they would not be limited to monitoring, but could also take action. Recent developments regarding the use of UAMs in this context include mission planning for emergency valve closure [32] and in search-and-rescue operations where a UAM automatically deploys a signal-emitting wristband on unconscious victims [16]. These papers used a single UAM unit to accomplish the task, but in a context where the task and the environment vary, multiple UAMs with different configurations could be used in coordination to optimally exploit the advantages associated with each configuration. Although UAMs are beginning to be used in more complex missions, there is a notable lack of development in the literature on mission planning methodology specifically to UAMs. Optimal UAM selection based on the task at hand and its local environment could be used in the mentioned context above. Machine learning, and in particular deep learning, has shown a strong potential impact in solving this kind of high-level task planning problems [4].

While machine learning has been used for parameter tuning, adaptive control or real-time trajectory planning in UAVs [156], deep learning is mostly used for vision applications in autonomous UAM and robotic manipulations. Examples include recognition and pose estimation of common objects in images [86], grasp pose selection [85], or even landing pad detection [88]. These methods tend to find a specific information in images or other data representations. An interesting problem is to obtain the context of the scene from its representation. To process scenes and cluttered environments, different data formats are used : images [85–87], voxels or volumetric representations [88, 89], or point clouds [90]. Some deep learning methods are specifically tailored to use these forms of representations. For example, LIDAR measurements are transformed to a volumetric density mapping in order to detect a landing pad [88]. A scene segmentation method based on proactive analysis and robot experimentation, in which robots were allowed to physically push objects, results in a point cloud representation [157].

A recent breakthrough in deep learning concerns network architectures specifically developed to directly use point clouds as inputs, with PointNet and PointNet++ being pioneers in this domain [90, 91]. The advantages of this type of inputs are their ease of use for 3D representation as opposed to images, and possible embedded implementations due to the low memory requirement to store point clouds [90] as opposed to more voluminous representations such as voxels [158]. Following PointNet, other variants for classifying and segmenting directly on point clouds were proposed in [159–162], with the goals of capturing more accurately information from local geometric features [160, 162]. Finally, a point cloud was also used to represent a scene and to detect whether it is traversable for an autonomous car [163]. The use of point clouds in deep learning is a recent development that easily lends itself in robotics. As a matter of fact, robots and UAMs are now often equipped with a LIDAR, a depth sensor or a stereo camera capable of producing point clouds. Current existing networks are used for object recognition based on point cloud representations [90, 91], and while they are certainly useful as is, they could be used for other purposes, such as full scene analysis for use in mission planning and dispatching of UAMs.

The objective of this paper is to develop a systematic method to select the most appropriate UAM to perform a picking task in a cluttered environment. This would address the need for a high-level guidance method as previously identified, and could be used in any situation where multiple picking tasks must be accomplished in various environments. The scenario considered in this paper is the following. A fleet containing a fixed number of units is defined, and for any object picking task, there is a unit within the fleet capable of performing it. Even though the fleet as a whole is general enough to perform any picking task, each UAM unit has its own specific configuration that does not make it suitable for all tasks due to the

presence of obstacles or the definition of the task itself. Furthermore, the tasks considered in this paper are limited to static picking tasks, since they have been identified as the most common ones [153]. To summarize, the UAM must reach a non-moving object and use its manipulator to pick it up according to a chosen pose. The question is : how to optimally and autonomously assign a drone of the fleet to a task ?

To answer that question, a deep learning classifier to select the optimal UAM is proposed. This classifier outputs a decision using information representing the task and the scene in which it is executed. Two classifier architectures are tested, one based on a modified PointNet architecture [90], and the other, based on a modified PointNet++ architecture [91]. First, PointNet is chosen as it is the pioneer in the field [161] and one must start with the simplest architecture in order to obtain a proof of concept. Second, PointNet++ is a natural continuation that can provide better capture of local features [91]. To train the classifiers, a large dataset is required and created using a path planning based method. This method can solve the UAM selection problem, but it is slow, computationally heavy, and therefore difficult to implement on embedded microcomputers ; this is the reason why deep learning classifiers are used.

The remainder of the paper is organized as follows. Section 5.3 presents a path planning based method capable of selecting the best suited configuration associated with a scene and a task. Section 5.4 presents the dataset generation, using the method in Section 5.3 to obtain the label associated with each generated data point. Section 5.5 presents the deep learning classifier architectures used, their generalization results and a use with real-life scenes. A discussion compares the results while offering insight into the limitations and how these methods could be applied or used in other contexts in Section 5.6.

5.3 Path planning based selection method

This section is dedicated to the presentation of the drone fleet, as well as the descriptions of the scenes and tasks to be performed. The path planning based selection method is is then detailed and used in the generation process of the database.

5.3.1 Definition of the UAM fleet

The available UAM configurations must be defined beforehand in order to be used in the path planning based selection method. In this work, four different manipulator configurations are available in the fleet, but any number of UAMs could be used as long as the fleet contains at least two different UAM configurations.

The selected configurations are derived from existing UAMs found in the literature, with minor modifications to their physical parameters so that they have the same reach, for comparison purposes. All four configurations are built from the same quadcopter flying base with a robotic arm attached below (Fig. 5.1). As for the attached robotic arms, they have a variable number of rotational joints : a 2-DoF arm [31] (Fig. 5.1a), a 3-DoF arm [38] (Fig. 5.1b), a 4-DoF arm [35] (Fig. 5.1c) and a 6-DoF arm [37] (Fig. 5.1d). The arm configurations are presented in Tab. 5.3 by their Denavit-Hartenberg parameters as defined in [150]. The same simple two-finger end-effector is attached to each arm as shown in Fig. 5.1. From a configuration point of view, each arm is a more complex iteration of the previous one, i.e., the 3-DoF arm can fully mimic the kinematics of the 2-DoF arm, etc. This fleet was chosen because the 2-DoF and 3-DoF arms allow for lighter UAMs and therefore longer flight time, while the 4-DoF and 6-DoF arms allow to reach more complex targets in cluttered environments.

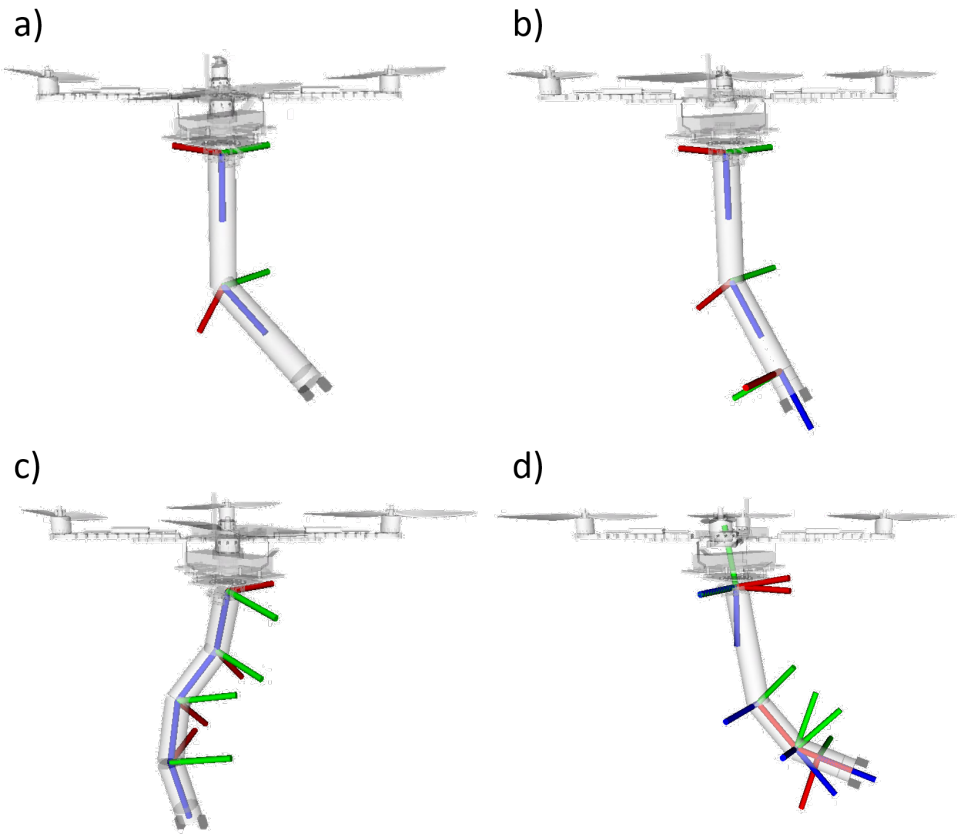

Figure 5.1 UAM configurations where the arm joints are shown as coordinate systems as used in the URDF modeling

First, two reference frames are defined, namely the North-East-Down (NED) inertial frame $\mathcal{F}_i$, and the body-attached frame $\mathcal{F}_b$ as shown in Fig. 5.2. The kinematics of the UAM base,

i.e., the quadcopter kinematics, is modeled as a floating solid body with three translations in $\mathcal{F}_i$ and only one rotation around the axis $\mathbf{z}^i$ (yaw angle ψ), which assumes that the base remains horizontal. To this end, the roll and pitch motions that induce the horizontal translation of the quadcopter are supposed to be negligible. This implies that the roll ϕ and pitch θ angles, as well as the translation speeds, will remain very low, and thus, that the horizontality assumption will be almost satisfied. This is perfectly reasonable, as in the case of position control, the input commands of the controller are often the desired positions expressed in the inertial frame $\mathcal{F}_i$ and the desired yaw Euler angle ψ. Also, for picking tasks in cluttered environments, trading speed for precision both during the movement and picking phases is judicious in order to minimize failures.

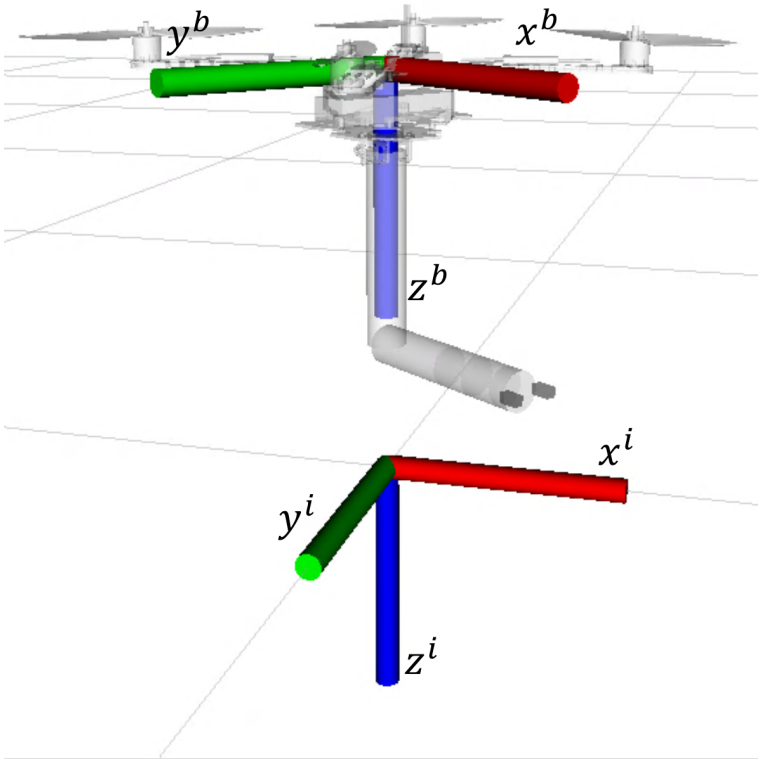

Figure 5.2 Reference frames of the system, the drone attached body frame is shown on the UAM, while the tool must match the desired orientation

The position and orientation of $\mathcal{F}^b$ relative to $\mathcal{F}^i$ is then given by the homogeneous trans-

formation matrix :

$$\mathbf{T}_{i/b} = \begin{bmatrix} \mathrm{c}_\psi & -\mathrm{s}_\psi & 0 & t_x \\ \mathrm{s}_\psi & \mathrm{c}_\psi & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.1}$$

where $\mathbf{t}_{i/b} = [t_x\ t_y\ t_z]^\top$ is the position of the origin of $\mathcal{F}_b$ relative to $\mathcal{F}_i$, and the short-hands $\mathrm{s}_x = \sin x$ and $\mathrm{c}_x = \cos x$ are used for brevity. The position of a point in $\mathcal{F}_i$ denoted by $\mathbf{p}^i = [x_i\ y_i\ z_i\ 1]^\top$ and its position in $\mathcal{F}_b$ denoted by $\mathbf{p}^b = [x_b\ y_b\ z_b\ 1]^\top$ are linked by the relation :

$$\mathbf{p}^i = \mathbf{T}_{i/b}\mathbf{p}^b \tag{5.2}$$

In the path planning framework, the quadcopter is effectively modeled as a 4-DoF serial robot ; three translations (prismatic joints) and one rotation around $\mathbf{z}_i$ (revolute joint) are applied sequentially at the CoM of the quadcopter.

The kinematics of the manipulator is expressed by the transformation matrices obtained from the Denavit-Hartenberg parameters associated with each UAM as presented in Tab. 5.3 and as defined in [150]. A static transformation is necessary between the frame $\mathcal{F}_b$ and the initial frame $\mathcal{F}_0$ of the arm used for the Denavit-Hartenberg modeling : it considers the offset between the quadcopter CoM and the attach point of the arm, i.e, the origin of $\mathcal{F}_0$, as well as the rotation between $\mathcal{F}_b$ and $\mathcal{F}_0$. This transformation is different for each configurations, depending on the orientation of the initial revolute joint of the arm.

Once the UAM configurations and kinematics are defined, they are modeled as a Unified Robot Description Format (URDF) file for use with the motion planning framework, namely MoveIt 1.0 [164]. The MoveIt framework, built upon Robot Operating System (ROS), is used for its ability to solve path planning problem while including complex scene representation, obstacles and self-collision avoidance. For collision avoidance purpose, simple geometric solids such as spheres, boxes and cylinders are used to approximate the UAM geometry and to simplify the calculations [165].

5.3.2 Definition of scenes and tasks

With models of the UAM fleet available, scenes and task objectives must be added to the framework to define the path planning problem. To represent the scene, a point cloud of the local environment is transformed into a voxel representation using the MoveIt Point Cloud Octomap Updater plugin with an Octomap resolution of 0.05 m. This step is necessary because MoveIt does not support point cloud directly. The capture and pre-processing of the

point cloud is outside the scope of this method and the point cloud is considered an accurate representation of the 3D scene in which the task is carried out, and obtained beforehand.

Static picking tasks are modeled by a target position and orientation, respectively representing the picking point on the target object and the orientation that the end-effector must reach in order to successfully grasp the object. Without loss of generality, the target position is always chosen as the origin of the point cloud. A simple translation can be applied to the point cloud to align the target with the origin. It is assumed that the UAM is already in the near vicinity of the target and is able to move freely around the scene, since only local planning is considered. If needed, a combination of multiple scenes could be used to represent more complex tasks or even waypoint navigation as discussed in Section 5.6.

The target orientation is represented by a normalized quaternion, as it allows a representation of the orientation in the $\mathbb{R}^3$ space without any singularity, such as the gimbal lock inherent in Euler angle representations. Moreover, they are easily normalized for use in the deep learning method. Picking orientation and target acquirement could be automatically obtained from a grasp detection algorithm such as [86], but is beyond the scope of this work. Note that no actual picking steps are considered, and the task is considered accomplished by simply reaching the gripping conditions. The target is not represented by a physical object, since payloads for UAMs are usually small, especially in cluttered environment ; it could nevertheless be added if needed.

5.3.3 Path planning and selection

With the scene and task represented, a path planning algorithm, RRTConnect [166], is used in conjunction with a numerical inverse kinematic solver in order to reach the desired pose, while avoiding obstacles and self-collision as demonstrated in Fig. 5.3. The planning algorithm takes a maximum of 5 s calculation time to obtain a path to the desired goal within numerical tolerances of 0.0001 m and 0.0001 rad on position and orientation, respectively. These tolerances are only used for the path planning algorithm and not for the actual movement.

For each UAM configuration, path planning is attempted for the same scene and task description, and a simple binary variable is recorded representing success or failure. Among all configurations capable of reaching the target (if any), the one with the lowest number of arm DoFs is chosen to carry out the task. This is the optimal criterion used in the UAM selection method. Indeed, a UAM with fewer arm DoFs is generally lighter, due to the lower number of motors and parts, which allows for lower power consumption and, consequently, longer flight time. Lower UAM weight, due to a smaller manipulator, also allows for a higher maximum payload using the same motors [18].

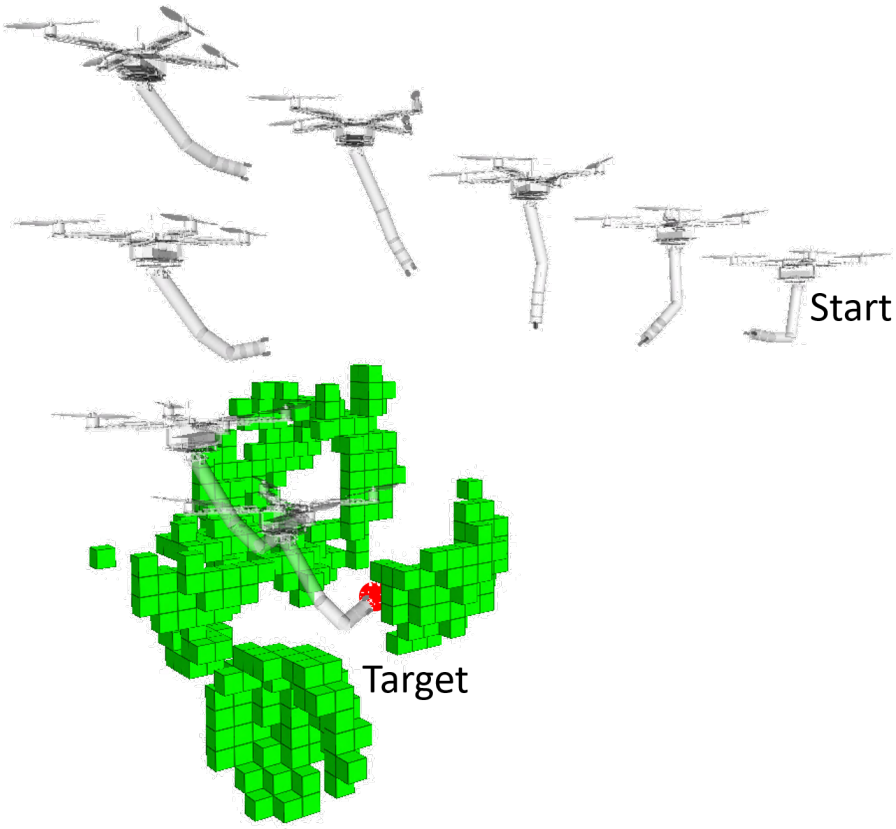

Figure 5.3 Solved path planning for a random scene as created by the dataset generation process

With this method, it is possible to select the possible optimal UAM within the fleet to accomplish a picking task in a specific scene and with a defined task, as per the scenario defined in Section 5.2. However a major drawback is its complexity of use and difficulty to be implemented on-board. The path planning framework, combined with the presence of all the different UAM models and the scene representation, would be difficult to transpose to an on-board system, where memory and storage are usually limited and dedicated to more essential tasks such as vision or control. To solve this problem and obtain a faster method, a deep learning classifier is developed to accomplish the same goal, by predicting the best suited UAM to perform a task.

5.4 Dataset for picking tasks in cluttered environments

The deep learning classifier must select the optimal UAM within the fleet, or decide if none is suitable. Therefore, a dataset with a high number of examples must be created in order

to train the classifier. As presented later in Section 5.5, the selected architectures have a large number of layers, and the more examples there are, the better the training. Therefore, a dataset containing 20,000 elements is constructed and used with the path planning based method in order to obtain the classes, i.e., the optimal UAM for a task, associated with each element. This high number of examples was retained based on similar dataset sizes. For instance, the ModelNet40 datasets is composed of 12,311 elements and used for classifications of objets in 40 categories [158], while ShapeNet contains 16,881 elements of 16 categories used for segmentation [167].

5.4.1 Dataset generation

For the purposes of this work, the scenes are generated by modifying other existing point cloud datasets, while tasks are randomly generated. In a limited amount of time, it would be impossible and impractical to create such a large number of physical scenes and create point clouds with a camera or laser sensor. in addition, existing datasets do not meet the requirements needed to represent cluttered environments, as they are composed of single objects, used in classification, or represent large tabletop scenes or large scale rooms with few objects, which are not appropriate for local object picking tasks in a three dimensional environment. The scene generation algorithm is similar to the one presented in SceneNet [168], where objects are sampled from existing datasets, then placed randomly in a 3D environment and iterated using an hierarchical annealing optimization process in order to obtain realistic scenes. A similar process is used in this work ; however obtaining realistic scenes were not a criteria in the generated scenes. The goal was primarily to obtain a variety of different geometries representing possible cluttered environments, realistic or not, so that the classifiers would be able to select UAMs for a wide range of scenes, and to include examples that are solvable by each UAM configuration. By combining uniform random distributions on the random variables used in the scene generation process and a large number of examples, we developed a dataset containing multiple examples of scene/task combinations for each configuration.

As presented in Section 5.3.2, the task targets are always positioned at the origin of the scene with a random orientation defined by a unit quaternion expressed in the inertial frame $\mathcal{F}_i$. The quaternions are randomly generated by picking three random numbers a, b, c from a

continuous uniform distribution in the range $[0, 1]$ and by using the formula :

$$\begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \sin(2\pi b)\sqrt{1-a} \\ \cos(2\pi b)\sqrt{1-a} \\ \sin(2\pi c)\sqrt{a} \\ \cos(2\pi c)\sqrt{a} \end{bmatrix} \tag{5.3}$$

From Eq. 5.3, 16,000 random orientations were generated [169]. As for the remaining 4,000 orientations, they were created as quaternions representing orientations that can be reached by the 2-DoF arm configuration ; they were obtained by multiplying two random rotations in the UAM workspace plane.

Multiple existing object point cloud datasets were modified and merged randomly together to create the scenes [170–175]. These datasets were originally used for either object recognition or for semantic labeling of cluttered scenes, and contained point clouds of common objects, with their associated labels ; only the point clouds were used in the dataset creation. To create the scenes, a random number of items (between 1 and 10) are selected from the available item categories and in random views. Each item in the datasets was represented in multiple views. Since the objective was simply to obtain random scenes with different shapes and obstacles, it was not necessary to have accurate representations of the existing objects, but only to obtain cluttered scenes.

For each object, a random starting position, a scale and an orientation are chosen. The position and the scale are constrained to ensure that the object is fully contained within a unit sphere when properly placed. The starting position $\mathbf{p}_c$ is chosen by selecting a random point inside the unit sphere and the objects are re-scaled using the following formula :

$$\mathbf{p}_{c,\text{norm}} = (\mathbf{p}_c - \bar{\mathbf{p}}_c)/(\max(\|\mathbf{p}_c\|) * s) \tag{5.4}$$

where the bar operator represents the average position of all points for each axis of the point cloud, and the norm is the Euler norm for a single point. The scale factor s is also constrained, so the object size does not become insignificant.

The orientation is obtained by applying three successive rotations with random angles α, β and γ to every point of the point cloud :

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}^r = \begin{bmatrix} c_\alpha & -s_\alpha & 0 \\ s_\alpha & c_\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\beta & 0 & s_\beta \\ 0 & 1 & 0 \\ -s_\beta & 0 & c_\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\gamma & -s_\gamma \\ 0 & s_\gamma & c_\gamma \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}^i \tag{5.5}$$

where the i superscript indicates the initial point cloud position and the r superscript indicates the rotated point cloud position. No constraints were applied to whether or not obstacles were crossing the origin of the sphere. A scene with obstacles near or crossing the origin would simply be categorized as unsolvable. The 20,000 elements in the dataset were generated by randomly associating a scene with an orientation.

Once generated, the dataset was used with the path planning based method described in Section 5.3 to obtain the label associated with each scene, i.e., the most suitable UAM able to accomplish the task, or no UAM, if not possible.

5.4.2 Dataset preparation for machine learning use

In this section, the created dataset is formally defined for use in machine learning training. As shown in Fig. 5.4, each example in the dataset has both features and a label. The features consist of a point cloud of dimension $M \times 3$, where M is the number of points in the point cloud, a quaternion representing the picking orientation in $\mathbb{R}^3$ and a label corresponding to each of the five available classes : the four UAM configuration classes and the unsolvable class. The unsolvable class is a special class in which no UAM from the fleet can reach the target, either because of the constraints in the geometry of the scene, or simply because the target is obscured by obstacles. Multiple elements of the dataset are combined in a batch of size B.

The created dataset does not possess balanced classes (20% of the dataset each), therefore class balancing is performed in preparation for training. A Synthetic Minority Oversampling Technique (SMOTE)-like method [176] is used, in which new elements of the minority classes are synthetically created while undersampling the majority class. This method has been shown to offer better performances than only oversampling members of the minority class [176]. Synthetic augmentation of dataset has also been used successfully in other related work [177]. New examples are created by rotating existing scenes by a random angle around the axis $\mathbf{z}^i$, rotating the quaternion by the same amount around the same axis and adding a Gaussian noise of mean $\mu = 0\,\text{m}$ and standard deviation $\sigma = 0.01\,\text{m}$ to each point of the point cloud. This noise does not affect the picking tasks results, as it only moves the points by a few millimeters in a random direction and therefore does not change the scene configuration and the label associated. These specific data augmentations were already used with success in a similar context in VoxelNet [178].

The dataset is randomly divided into a training set, a validation set and a test set, in proportions of 70%/15%/15%. The training set is used to train the convolutional neural network (CNN) classifiers, while the validation set is used during training to avoid overfitting. Finally,

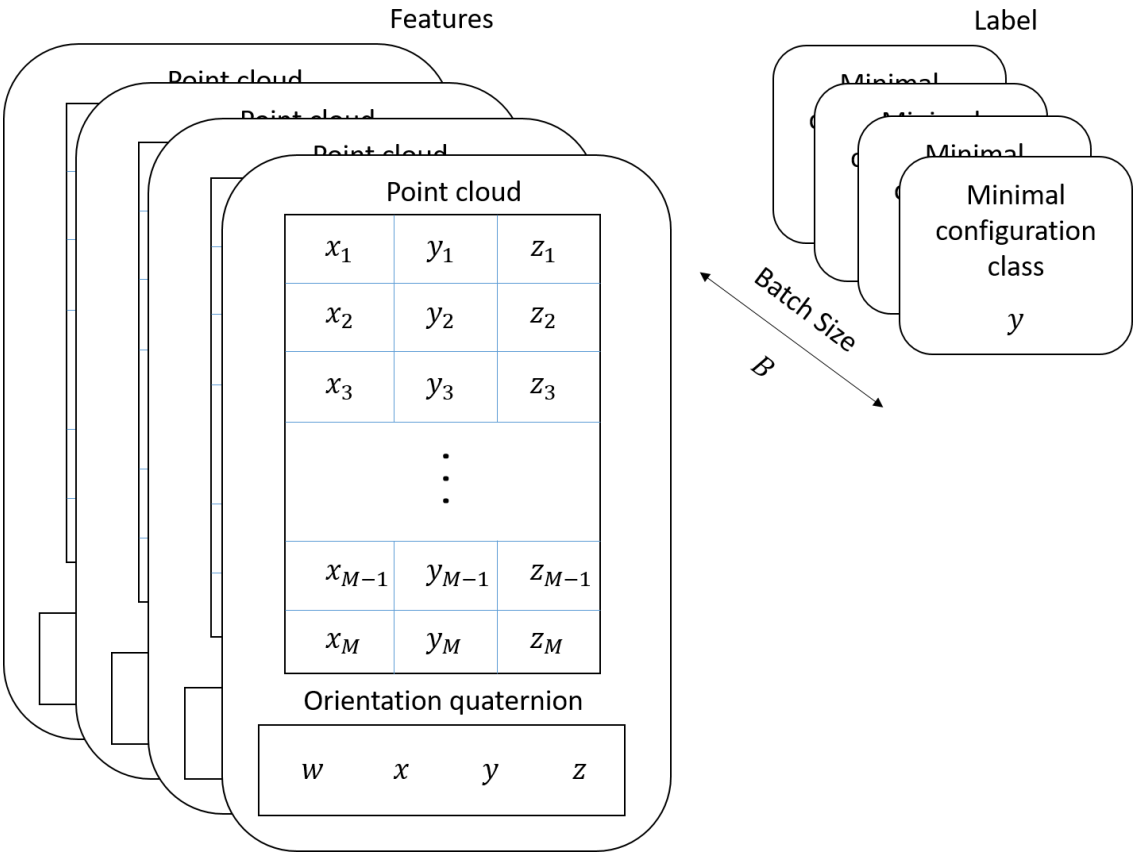

Figure 5.4 Structure of the dataset

the test set is not used in training so that the generalization capability of the classifiers can be evaluated. The complete process including the dataset generation, label calculations using the path planning based methodology and deep learning based methodology is illustrated in Fig. 5.5.

5.5 Deep learning based selection method

Deep classifiers are trained to obtain the desired UAM based from the scene and task description. Two architectures are used and compared : the first one is based on PointNet [90], and the second one is based on its successor, PointNet++ [91]. Both were trained on the created dataset to compare their results. While at first glance PointNet++ might be more suitable for the problem at hand, due to its ability to process local geometric information [91], it has a more complex architecture and can be more difficult to train. Notwithstanding the different architectures, the problem at hand is a classification problem where the classes are the different UAM configurations in the fleet and the unsolvable class. The modifications

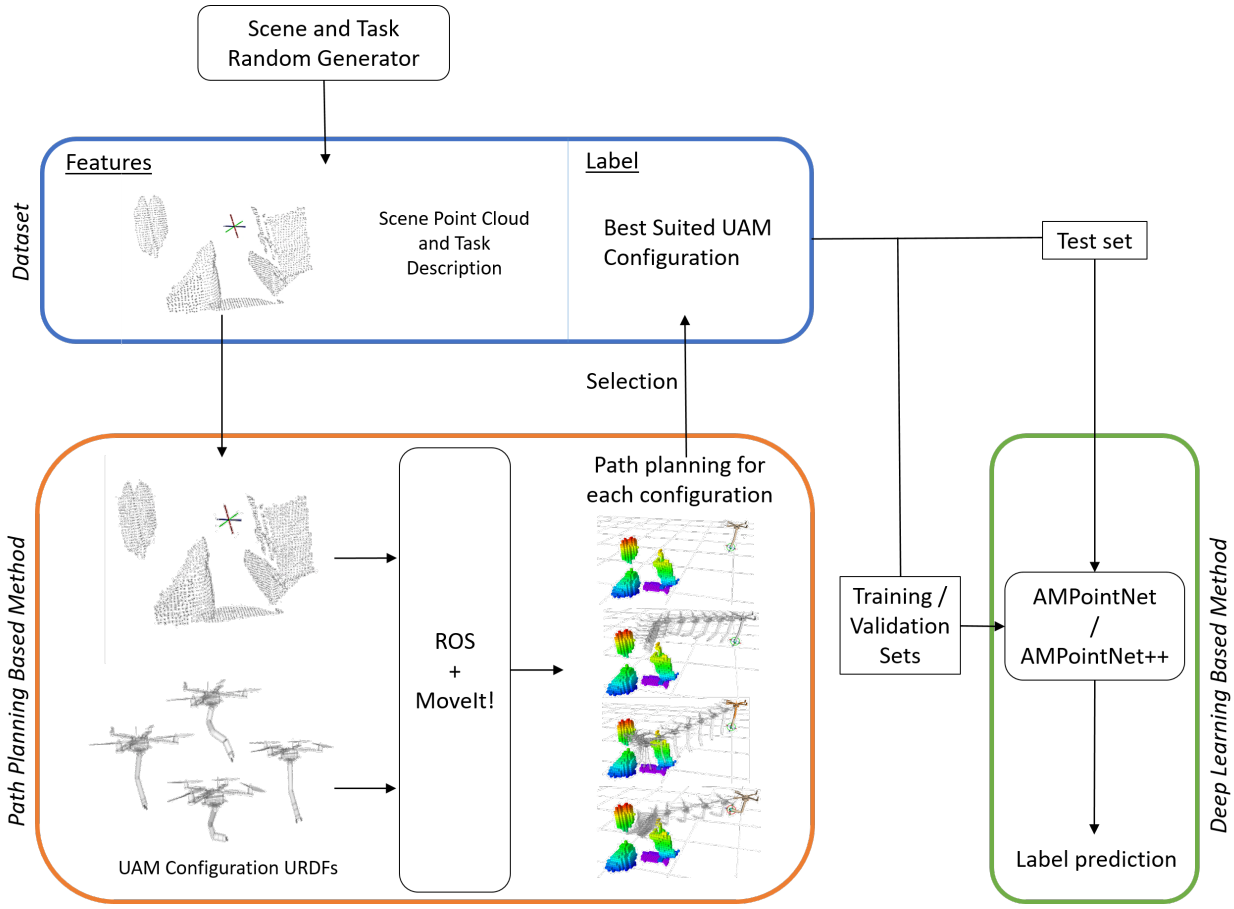

Figure 5.5 Graphic summary of the presented methodologies and their interactions, from the dataset generation to its use in the deep learning classifiers

made to the original architectures are presented here, as well as the choices made for the different hyperparameters.

5.5.1 AMPointNet and AMPointNet++ architectures

The first architecture is based on the PointNet architecture [90] with modifications, and named AMPointNet for this specific use (Fig. 5.6). The architecture consists of two parts. The first part extracts features directly from the point cloud, using multiple shared Multilayer Perceptrons (MLPs) and T-Nets, as shown in Fig. 5.6. The shared MLPs in the original architecture are implemented as multi-channel 1D convolution layers with a kernel size of 1. A batch normalization [179] and Rectified Linear Unit (ReLU) function are used after each layer of the shared MLPs. The second part is the actual classifier using the features from the first part and the desired orientation quaternion as input to the classifier. The classifier is a deep 3-layer MLP with batch normalization and ReLU activation functions after each layer. A dropout layer with a dropout rate of 50% is used after the penultimate classifier layer. The loss function for this architecture is a negative log likelihood (NLL) function applied on the output class with a regulator term to limit the deviations of both T-Nets from the identity matrix, as presented in [90].

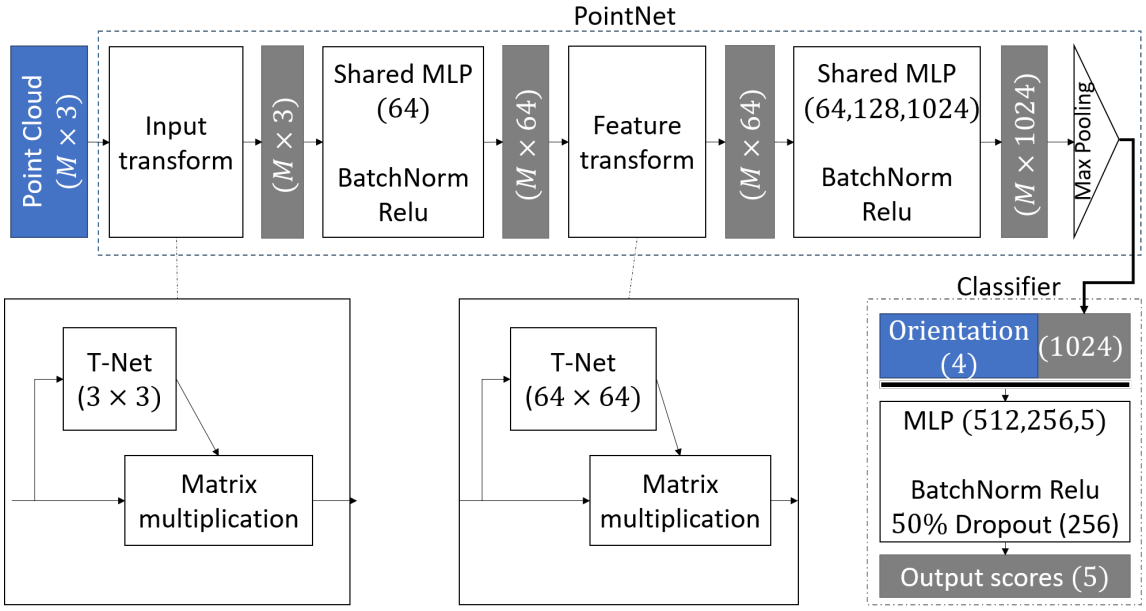

Figure 5.6 Architecture of the PointNet based AMPointNet CNN

The second architecture is based on PointNet++ [91] and is also modified to take into account the problem at hand ; this version is named AMPointNet++ (Fig. 5.7). The single scale

grouping (SSG) version of the original architecture is used, with the K-Nearest Neighboor (KNN) algorithm used for cluster generation. This architecture uses a sampling and grouping phase before applying mini-PointNets to nested partitions of the point cloud. This effectively allows AMPointNet++ to capture and characterize local structures, in a way that AMPointNet cannot. The first sampling and grouping group creates clusters of $K = 512$ points and samples the points in a radius of $r = 0.2\,\mathrm{m}$ around the center of the cluster. A Mini-PointNet is trained for use on the clusters, leveraging the AMPointNet features part of the architecture presented in this Section. The operation is repeated for a second sampling and grouping phase with $K = 128$ and $r = 0.4\,\mathrm{m}$ and finally applied to all points as shown in Fig. 5.7. The input of the classifier section also includes the orientation quaternion, as for AMPointNet. The loss function for this architecture is a simple NLL without regulations.

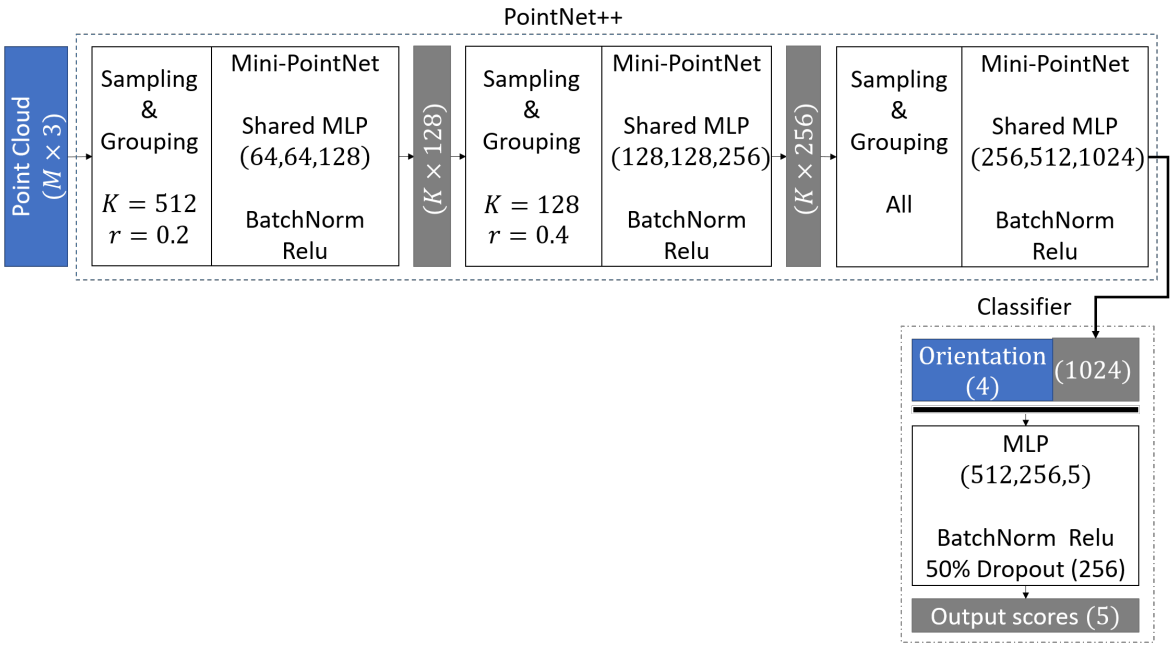

Figure 5.7 Architecture of the PointNet++ based AMPointNet++ CNN

The training parameters are summarized in Table 5.1. The LR scheduler used for AMPointNet was implemented following the training procedures for the original architectures [90, 91], and also confirmed by experiments on the modified architectures. After training, the validation set is used to select the optimal model obtained during training that achieves the most accurate results without overfitting, based on accuracy of the predicted classes in comparison to the actual classes. The hyperparameters used in this work were not optimized ; they were obtained based on commonly used values and an *ad hoc* manual search. Training was conducted using

PyTorch 1.7 and CUDA 10.2 for GPU calculations on a Windows 10 computer equipped with a NVIDIA Titan X GPU. Training of PointNet took between 6 and 12 hours, while PointNet++ took between 36 and 60 hours to complete.

5.5.2 Test Results, Accuracies and Confusion Matrices

An interesting aspect of the classifiers is that the same architecture can work with different point cloud sizes M as input. To choose the point cloud size, both architectures are trained with varying size. Figure 5.8 shows the classification accuracy of both architectures with different point cloud sizes. Using 2048 points does not seem to improve the accuracy of both architectures significantly, therefore, 1024 points is deemed to be an adequate point cloud size for further training.

With training conducted using the architectures and parameters defined above, results are obtained from the test set. This set was not used at any stage of the training in order to obtain an evaluation of the generalization of the classifiers. The CNNs are trained to predict the same exact class as the one identified with the path planning method, but it is not the only metric tracked. As mentioned in Section 5.3.1, UAMs can achieve the same exact trajectory as those with a lower number of DoFs. Therefore, if the predicted class is a UAM with a higher number of DoFs, it will still be able to accomplish the task, even if not optimally. This is defined as the task accuracy and obtained by averaging the accuracy per label, of the upper triangular matrix in the confusion matrix, except for the unsolvable class column, as shown in red in Figs 5.9 and 5.10. The other metric is the classification accuracy, i.e, the commonly used accuracy, and in this case represents the ability of the CNN to predict the same class as the one obtained by the path planning method. As shown in the confusion matrices in Figs 5.9 and 5.10, this is represented by the average of the diagonal of this matrix. It should be noted that this classification task is difficult to accomplish by humans alone. It is tedious for a person to try to select the best suited UAM configuration even if the scenes and tasks are represented visually. Therefore, comparison to human accuracy, as it is often carried out in

Parameter	Value
Optimizer	ADAM [180, 181]
Learning Rate (LR)	0.001
Batch Size (B)	16
LR scheduler	0.5 every 20 epoch (AMPointNet only)
Maximum epoch	200

Tableau 5.1 Training parameters

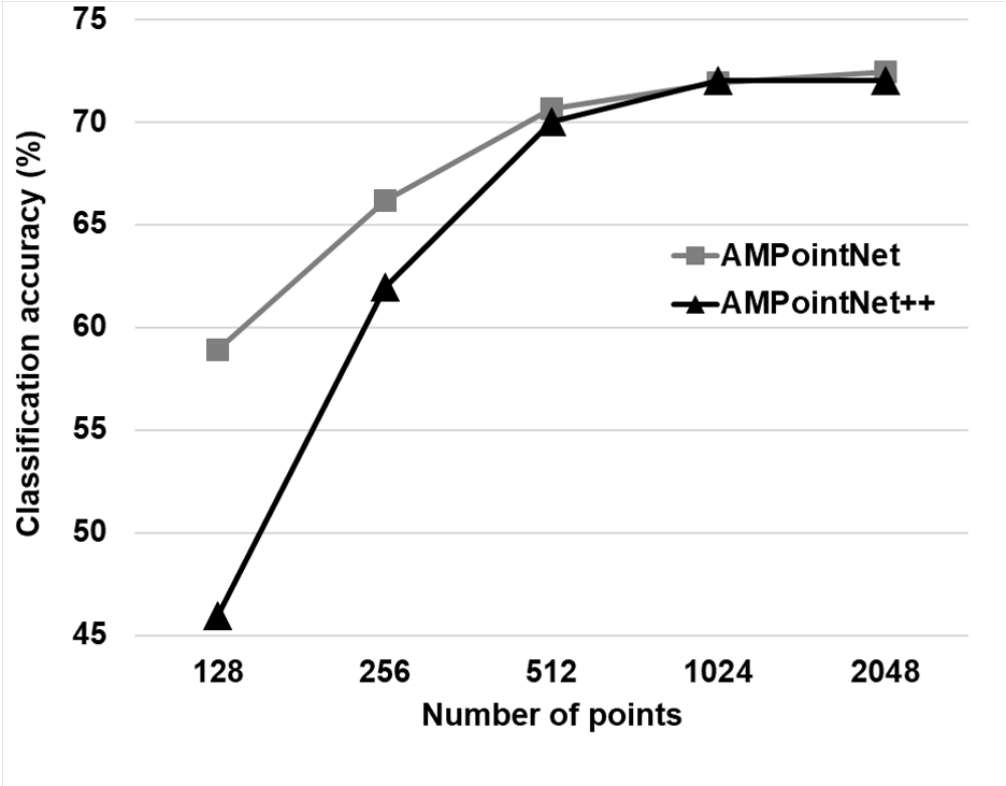

Figure 5.8 Effects of point cloud sizes on classification accuracy for AMPointNet and AMPointNet++ architectures

image classifications, is not possible in this case.

Figure 5.11 shows the average accuracies over five training runs, where the error bars show the minimum and maximum variations in accuracy induced by the training process for both AMPointNet and AMPointNet++.

The AMPointNet architecture obtains a 71.2% mean classification accuracy and a 80.4% mean task accuracy on the test dataset over five full training runs. A single prediction takes 3 ms without parameter loading. As shown in the confusion matrix in Fig. 5.9, the hardest classes to categorize are the 3-DoF and 4-DoF classes. This can be explained by the fact that those UAM configurations are similar to each other and therefore, small variations in the processed scene features, such as missing or misrepresenting some features, could easily change the classification. On the other hand, the 2-DoF configuration is restricted to planar motions, and the 6-DoF class has more unique capabilities, making them easier to classify.

The AMPointNet++ architecture is able to obtain a mean classification accuracy of 71.9% and mean task accuracy of 80.9%, also on the test dataset over five full training runs. Predictions take 391 ms without parameter loading. Fig. 5.10 shows the same magnitude of ca-

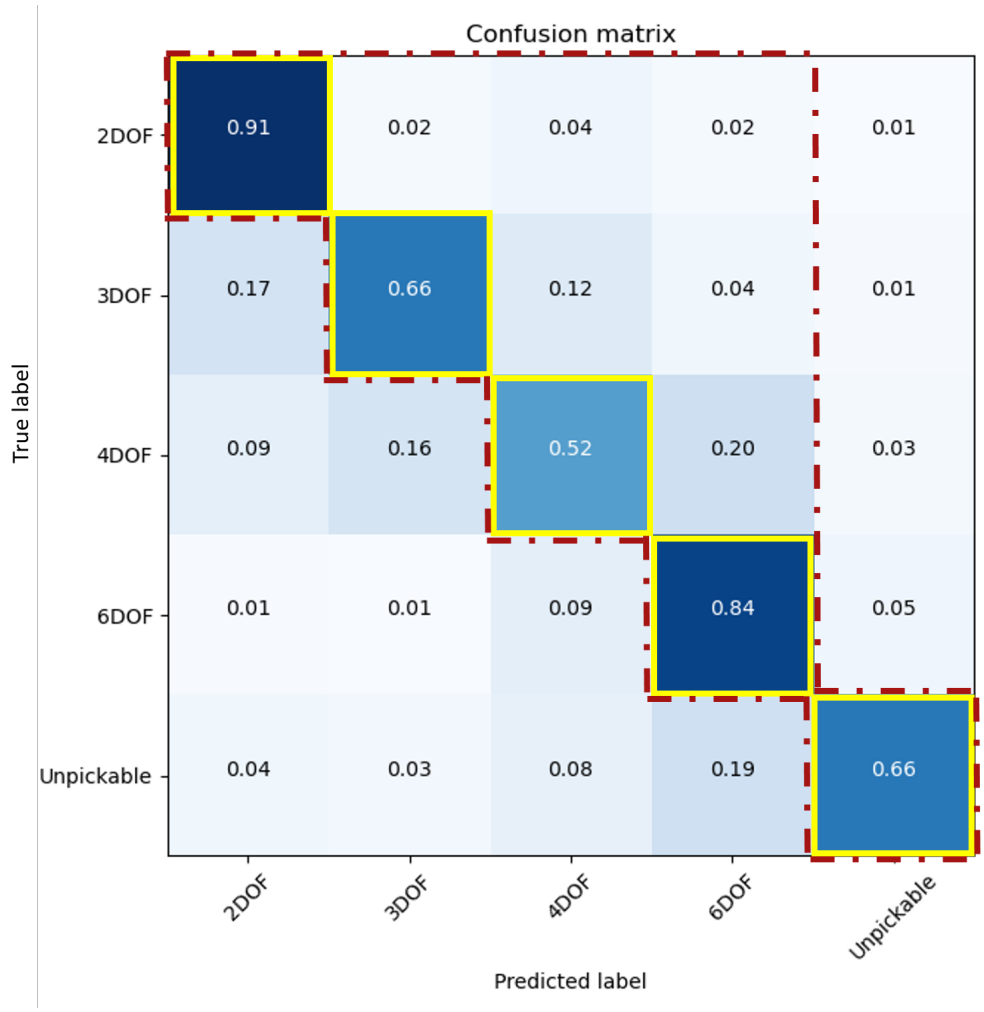

Figure 5.9 Confusion matrix of the best training obtained for AMPointNet

tegorization errors as for AMPointNet.

Table 5.2 summarizes the average accuracies, predictions times and also includes parameter sizes for both architectures and the path planning based method (MoveIt). The parameter included for both CNN architectures is simply the size of the learned parameters in the networks. Although a bit counter-intuitive, the more complex architecture AMPointNet++ is lighter in the number of parameters. This is because the smaller networks are trained on a subset of the entire point cloud instead of larger layers that consider all points at once as with AMPointNet. The downside is that the architecture has a higher prediction time due to the number of operations that need to be performed, as shown in the prediction times. The path planning based method achieves a 100% accuracy due to its exact nature, but the cost is an extremely high prediction time, ranging from 5 to 20 seconds, depending on which UAM can carry out the task. While this method does not learn parameters, the parameter

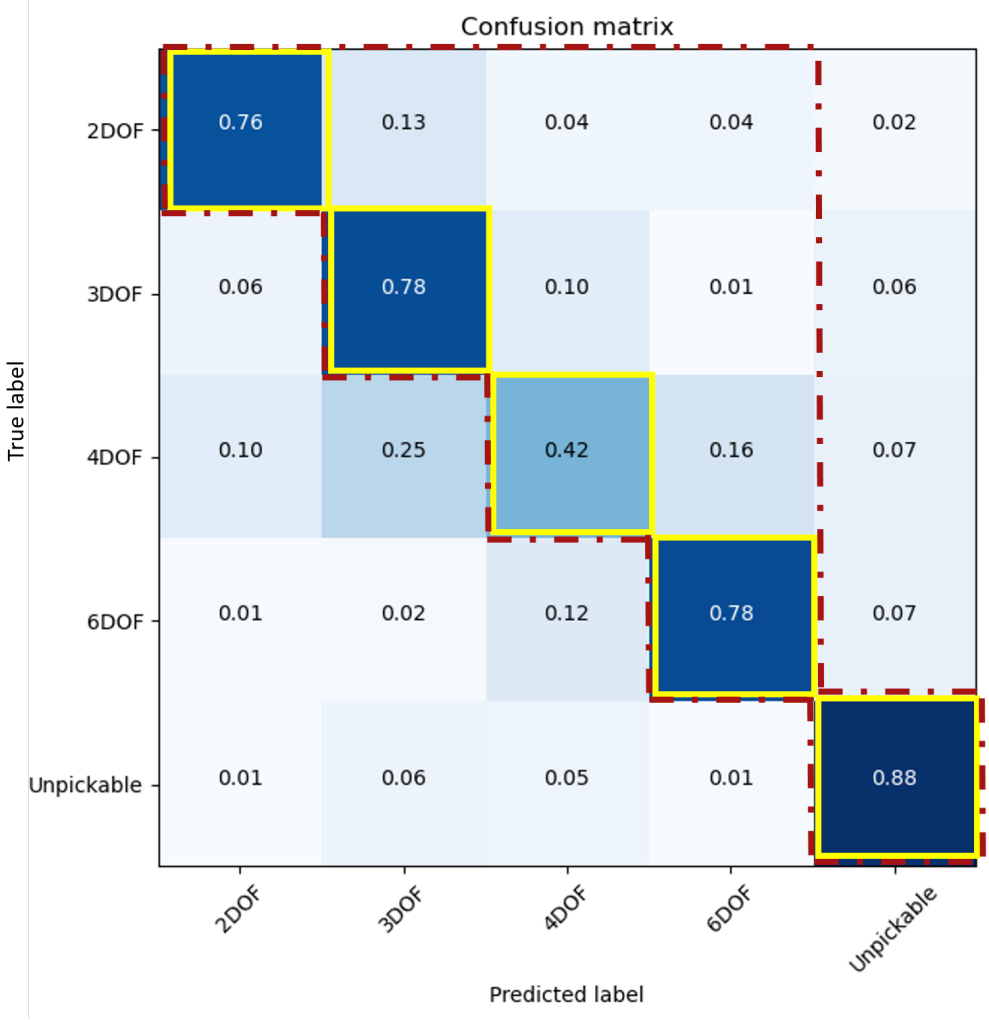

Figure 5.10 Confusion matrix of the best training obtained for AMPointNet++

size includes a special version of the point cloud needed to integrate MoveIt and the various UAM models needed to solve the path planning problem. The parameter size for the CNN and MoveIT based methods does not include the various libraries needed, such as PyTorch for CNN and ROS/MoveIt.

The differences in classification and task accuracies between AMPointNet and AMPointNet++ are less than one percent. The variations between different training runs remain below 3% as shown in Fig. 5.11. Therefore, it can be stated that both architectures have the same performance with AMPointNet++ having a slower prediction time. Since the predictions of AMPointNet++ are one hundred times slower than AMPointNet, using the latter is recommended for the UAM selection objective. Moreover, compared to the MoveIt-based method, the predictions are a thousand times faster, which confirms the choice of a deep

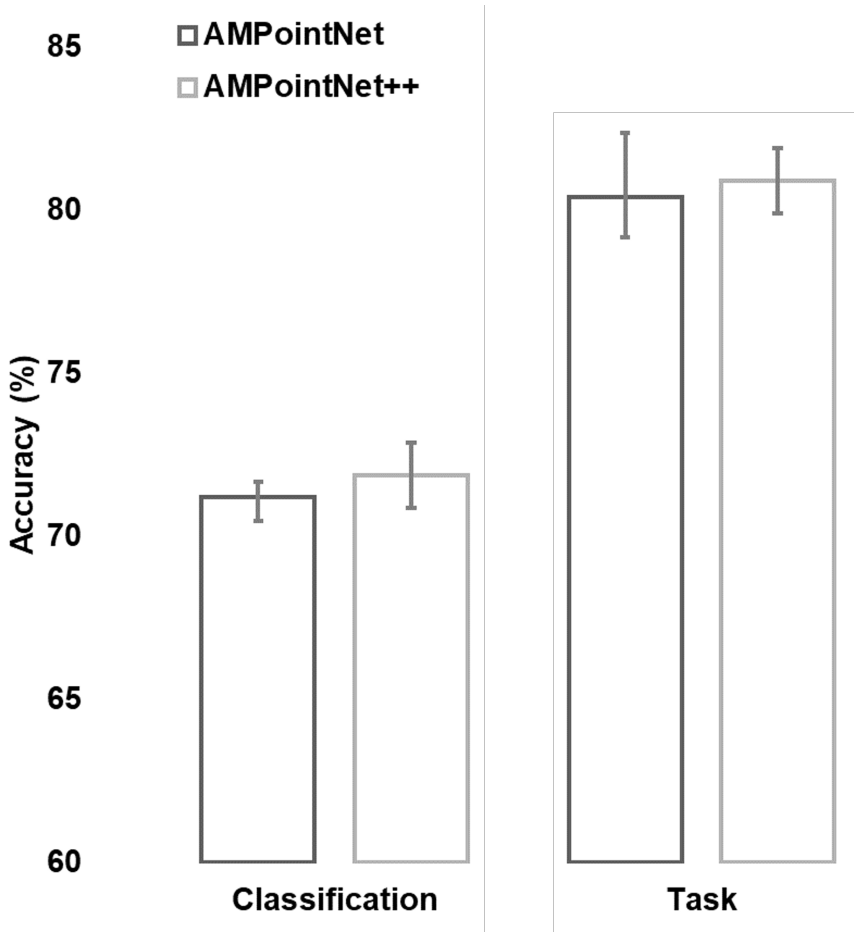

Figure 5.11 Classification and task accuracies of AMPointNet and AMPointNet++

Tableau 5.2 Accuracies, prediction times, and parameter sizes for AMPointNet and AMPointNet++ CNN and MoveIt based path planning method

	AMPointNet	AMPointNet++	MoveIt
Mean classification accuracy (%)	71.2	71.9	100
Mean task accuracy (%)	80.4	80.9	100
Prediction Time (s)	0.003	0.391	$5 - 20$
Parameter size (MB)	13.6	7.8	> 500

learning-based method. Four UAM configurations were used in this work, but if more were to be available in a fleet, the prediction times would become even more important with the MoveIt based method. The difference in parameter size for both architectures is negligible and does not impact the choice of architecture, but also confirms the choice of the classifiers over the path-planning based method.

5.5.3 Experiment using real-life scenes

An experiment was conducted in which tasks were defined in real-life scenes. The path planning method was used to obtain the optimal UAM configurations for each scene and then compared to the predictions using the classifiers. This experiment was conducted to whether the trained model could also be applied to real-life scenes. Five scenes were constructed in such a way as to obtain one scene solvable by each UAM configuration, including one unsolvable scene. They were then scanned using an Intel RealSense D415 camera to obtain RGB-D videos of the scenes. These videos were then used in a scene reconstruction pipeline, as described in [182,183], and implemented with the Open3D [184] library, in order to reconstruct point clouds of the different scenes. The reconstructed point clouds were sub-sampled to 1024 points using clustered vertex subsampling, i.e., a simple one-per-cell sampling. They were then translated to align the origin of the reference frame with the task target position, arbitrarily chosen with educated guess based on existing scenes in the dataset. Orientation quaternions were randomly generated for each scene to represent the tasks. The reference frames in the point clouds of Fig. 5.12 show the position and orientation of the associated tasks. Points within a unit sphere around the origin were retained, while all others were discarded.

After preprocessing the point clouds, each of them is used in the path planning method to obtain the UAM configuration with the smallest number of DoFs capable of solving the task, as identified in the path planning configuration column of Fig. 5.12. As mentioned in Section 5.3.1, each UAM in the fleet is able to accomplish the same tasks as those with fewer arm DoFs. This was confirmed in this experiment, since the 2-DoF scene can also be solved with all other UAM configurations and the same behaviour is seen on all scenes. After obtaining the class associated with each scene, the point clouds and quaternions are used as input in the trained AMPointNet and AMPointNet++ classifiers. The predicted class for each of these architectures are shown in the prediction column of Fig. 5.12. AMPointNet obtains a 80% classification accuracy, while AMPointNet++ do not classify the 2-DOF scene correctly, obtaining 60% classification accuracy. Both architectures seem to have difficulties with the 4-DOF scenes, as observed with the simulation test results in Figs 5.9 and 5.10. Although five scenes can be considered low, the results are similar to those obtained on the test dataset, and tend to confirm the choice of AMPointNet, due to the low increase in accuracy of using AMPointNet++ compared to its drawback. This experiment also assess the stability of the classifiers' results with respect to the point clouds. Even though the classifiers were trained using an accurate 3D representation of objects, reconstructed in the dataset generated with simulated scenes and tasks, they were able to achieve similar performance on 3D scans of

real-life scenes, with all their associated imperfections that were not included in the synthetic scenes.

5.6 Discussion

5.6.1 Potential use

The two methods presented in this paper are developed with the same goal of selecting the optimal UAM of a fleet to accomplish static picking tasks in cluttered environments. The path planning based method can be considered as an exact method, while the CNNs are estimators trained with a dataset constructed with the former. They could be used, for example, in a cluttered warehouse with multiple picking targets, where a quadcopter equipped with a point cloud generating sensor could travel the warehouse in order to identify the different targets, their grasping orientations and generate point clouds of the local scenes. Next, the introduced CNN based method could be used in conjunction with the point clouds and orientations to select the optimal UAM able to carry out this task. From there, the dispatcher quadcopter could notify the selected UAM to accomplish the task. This is where the usefulness of the UAM selection method is interesting. In our fleet example, it is true that sending the 6-DoF UAM would always result in a task accomplished (except for unsolvable tasks). However, using the best suited UAM configuration has multiple advantages. A lower number of arm DoFs is associated with a lower mass and therefore a longer flight time. UAMs with fewer DoFs can also lead to fewer errors in correct task completion due to their lower system complexity. In terms of fleet dispatch, if multiple UAMs are working simultaneously, the 6-DoF UAM can be occupied with another task or load. Optimal UAM selection could extend the total flight time of the fleet, while performing multiple tasks simultaneously. Failed predictions are also not as problematic as they seem, since if a task fails, another UAM with a superior DoF arm could be sent to perform the task. This information can also be stored and used for subsequent training of CNNs.

Another use of the path planning method could be made with minor modifications. Instead of selecting the most suitable UAM to perform a task, the method could be used in a system design context. If the tasks are already known and the problem is to design a UAM from existing parts, the problem could be reversed. In this case, the method still checks if the path planning problem is solvable, but iterates over the possible UAM configurations to find some configurations that can solve the tasks, based on the problem definition. With the new UAM configurations found, the CNN can be trained and used as presented.

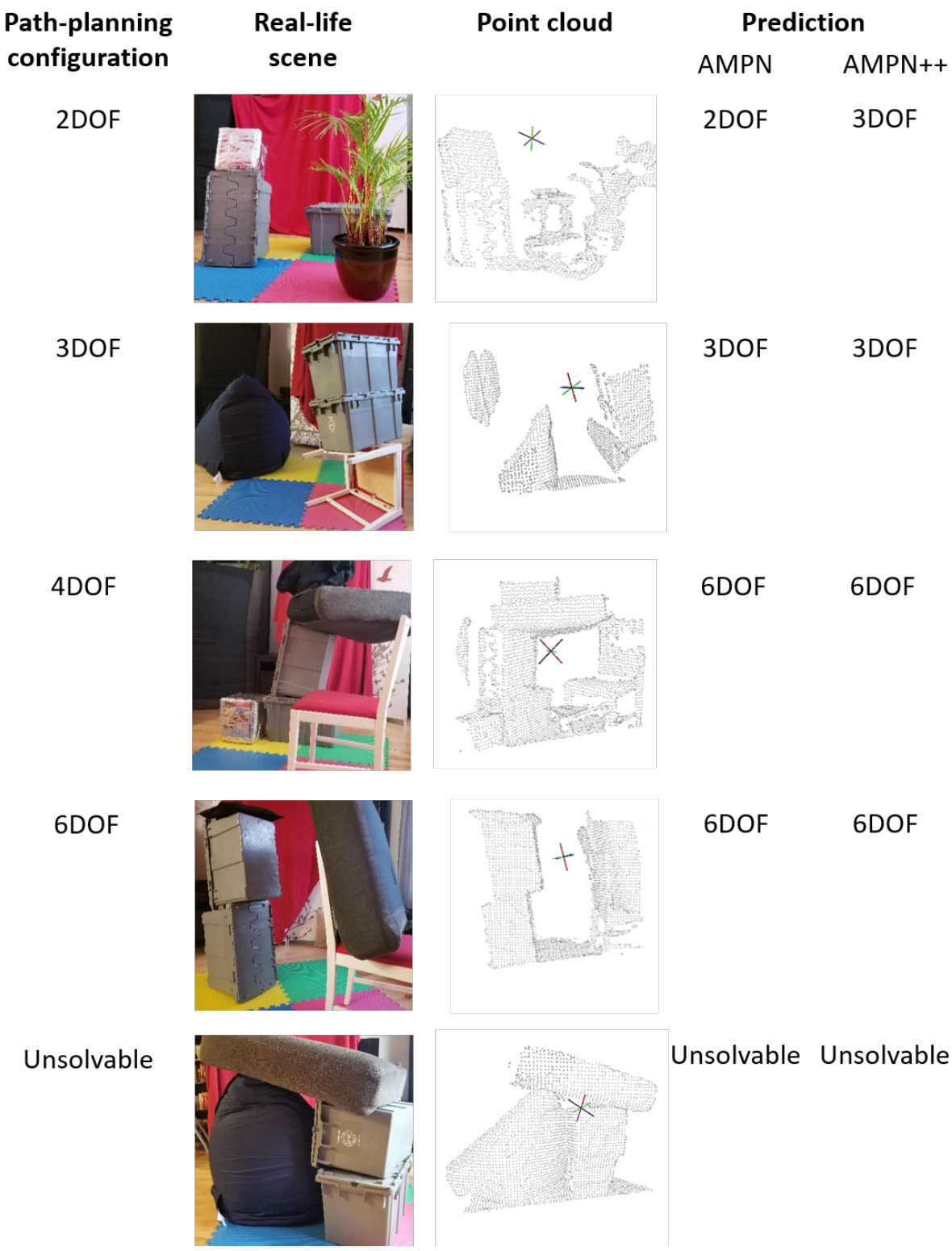

Figure 5.12 Point clouds extracted from real scenes, with the associated path planning based optimal configuration, and the predicted optimal configurations from AMPointNet (AMPN) and AMPointNet++ (AMPN++)

5.6.2 Assumption and limitations

Hyperparameters, either for the path planning based method or for the deep learning classifiers, were selected based on common knowledge and *ad hoc* manual search. Tuning and optimization could be conducted for each use case, dependently of fleets and conditions.

Both methods were presented with a point cloud located entirely inside a unit sphere around the target. If necessary, this could be used with multiple scenes inside unit spheres where the objective of one scene is the beginning of the next one. Using the predicted labels for all scenes, the maximum DoF is taken as it is the most constraining. This would be akin to waypoint path planning, but in this case, with the configuration selection process.

In this method, only the arm complexity was used to obtain the optimal class during the path planning method, meaning that the algorithm only looked for the UAM configuration with the lowest number of DoFs that could solve the path planning problem for the task. It is possible to add other criteria to this optimization and train the CNNs with those results. For example, an energy minimization criteria could be used, where the total energy cost of accomplishing the task is also evaluated and used in the minimization process. This could be done by including an energy estimation module in the path planing method. This could be a potentially good criterion, since flight time is often one of the most constraining parameter when working with UAMs and quadcopters in general. Another criterion could also involve the fleet configuration, such as flight time from the current UAM position. For example, the lowest arm DoF configuration could be selected by the CNN but modulated by the flight time of the nearest UAM. Therefore, the nearest appropriate UAM could be used. These criteria are extremely dependent on the conditions under which this method is applied, and therefore in this paper, optimization for the simplest configuration was chosen.

5.7 Conclusion

A method to select the optimal UAM within a fleet in order to accomplish a specific picking task was presented in this paper. First, a path planning based method was introduced and used to generate a dataset of random orientations, scene point clouds and labels. Secondly, this dataset was utilized to train deep classifiers, named AMPointNet and AMPointNet++, to predict the optimal UAM for the task. AMPointNet was able to achieve a classification accuracy of 71.2%, while AMPointNet achieved an accuracy of 71.9% . These minor differences in results between the two architectures tend to suggest that using AMPointNet++ is not worth the slower computation. Both architectures were tested on real-life scenes and led to similar results, which supports our approach.

In terms of future research directions, the path planning based method could be used in a design science context to build UAMs specifically for the task(s) being performed. In addition, the methods are dependent on the fleet of available UAMs. This means that for a different fleet, a new dataset must be created and the models must be trained. Future works could include the UAM configurations directly into the network architectures, so that no new learning is required between fleets. Finally, although this work is limited to static picking tasks, it could be extended to mobile picking and other similar tasks.

5.8 Appendix

Tableau 5.3 Denavit-Hartenberg parameters of the UAM arms

θ	d (m)	a (m)	α (rad)
2-DoF arm			
θ_1	0	0.2	0
θ_2	0	0.2	0
3-DoF arm			
θ_1	0	0.2	0
θ_2	0	0	$\pi/2$
θ_3	0.2	0	0
4-DoF arm			
θ_1	0	0.1	$-\pi/2$
θ_2	0	0.1	$\pi/2$
θ_3	0	0.1	$-\pi/2$
θ_4	0	0.1	0
6-DoF arm			
θ_1	0	0	$-\pi/2$
θ_2	0	0.2	0
θ_3	0	0	$\pi/2$
θ_4	0.1	0	$\pi/2$
θ_5	0	0	$-\pi/2$
θ_6	0.1	0	0

CHAPITRE 6 DISCUSSION GÉNÉRALE

6.1 Utilisation des méthodes introduites dans un processus de conception

Les méthodes de support à la conception présentées dans les chapitres 4 et 5 s'inscrivent dans le cadre de l'accomplissement de tâches de saisie par un UAM tel que défini à l'aide de la taxonomie du chapitre 3. La taxonomie permet alors de réduire des missions effectués dans différents contextes en tâches simples. Durant la conception d'un système mécatronique complexe, comme un UAM, une attention particulière doit être accordée à la conception intégrée, soit la considération de tous les sous-systèmes et domaines de manière simultanée [10]. Cependant, les difficultés associées aux aspects techniques peuvent prendre une place trop importante dans le processus de conception [53]. Les méthodes introduites s'inscrivent donc en support au développement de la commande et du guidage, pouvant permettre de simplifier la tâche des concepteurs à ces étapes du design détaillé, et laisser plus de place aux tâches de conception intégrée. La taxonomie du chapitre 3 quant à elle permet de supporter la conception intégrée tout au long du processus, de manière plus qualitative, notamment en permettant de simplifier la description des tâches et des missions et en fournissant un langage commun pour décrire les actions possibles de l'UAM.

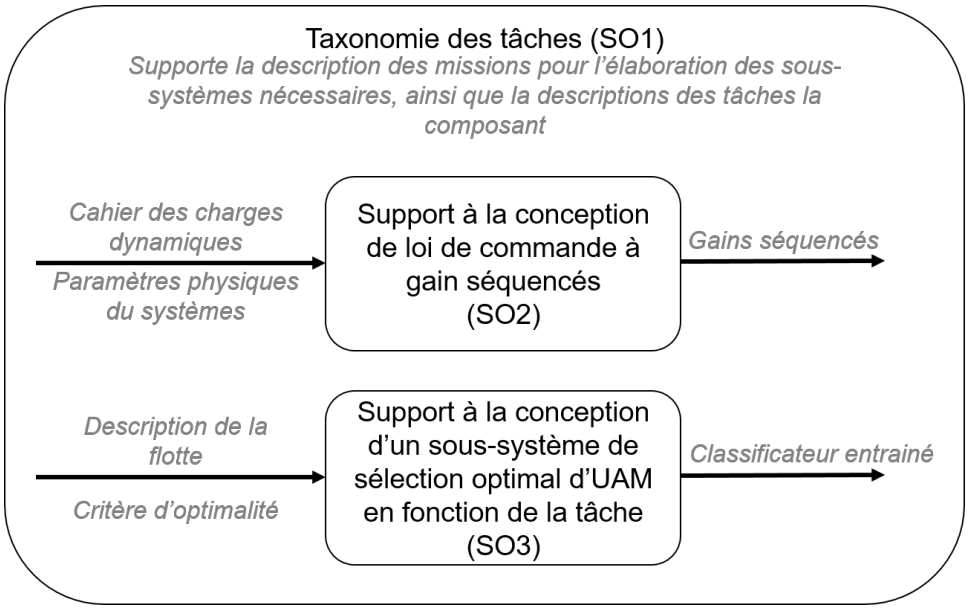

Figure 6.1 Schéma représentant les entrées et sorties des différentes méthodes de support à la conception développées dans cette thèse

Les méthodes développées répondent alors à l'objectif principal de cette thèse, soit de sup-

porter la conception d'UAM, au niveau des sous-systèmes choisis. La figure 6.1 présente l'utilisation des méthodes développées du point de vue de support à la conception. Elle montre les entrées nécessaires à choisir par le concepteur, ainsi que les résultats obtenus par ces méthodes. Ces deux outils sont supportés par la taxonomie de tâches, qui permet la classification des différents aspects de missions.

Bien évidemment, un UAM accomplissant des tâches de saisies ne sera pas seulement composé d'une loi de commande ainsi que d'un algorithme de sélection parmi une flotte. La figure 6.2 illustre un exemple d'architecture d'UAM pouvant servir à l'accomplissement d'une telle tâche de manière autonome, en considérant que la tâche doit être accomplie dans l'environnement local de l'UAM. Les sous-systèmes traités par cette thèse sont illustrés en couleur. Ceux servant à la navigation dans l'environnement large non cartographié ne sont cependant pas inclus ; cette capacité serait particulièrement utile pour un UAM effectuant, par exemple, des tâches de saisie suite à une catastrophe.

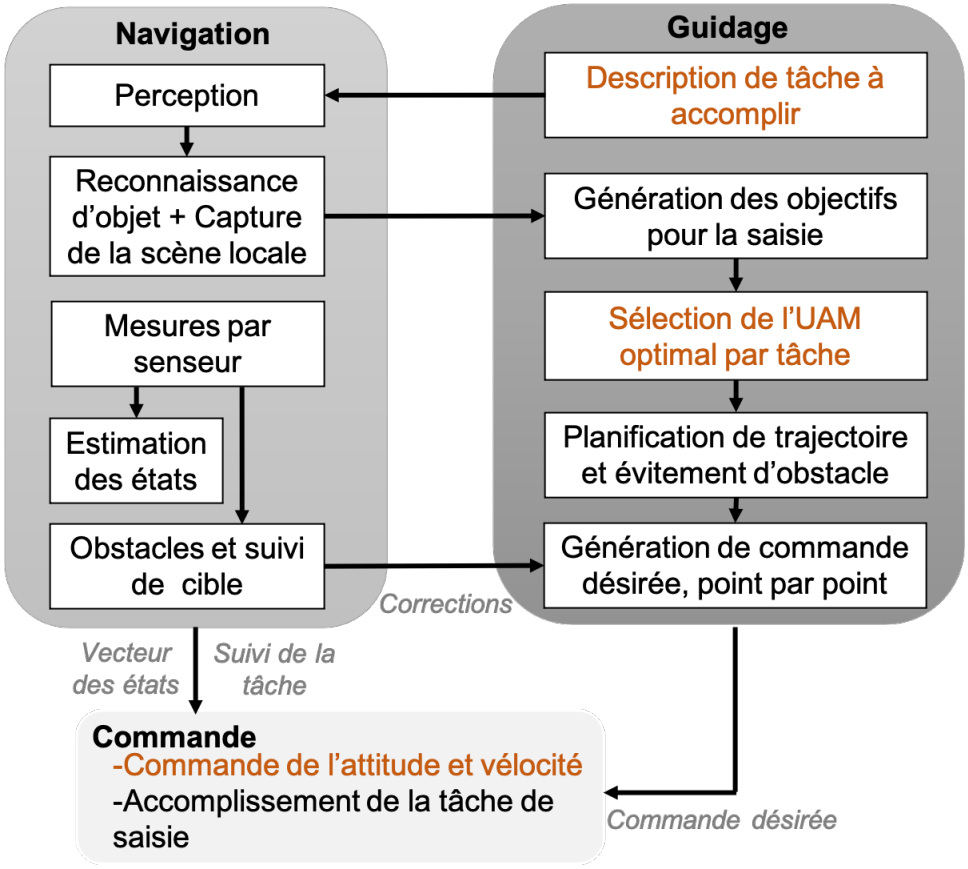

Figure 6.2 Schéma représentent les étapes nécessaires à l'accomplissement d'une tâche de saisie dans l'environnement local de l'UAM. Les sous-systèmes traités dans cette thèse sont en couleur.

Des capteurs et des caméras doivent obtenir une estimation des états dynamiques du système, mais aussi un visuel de la scène où se déroule la tâche, rajoutant des capacités de perception et de mesure des états. Ceci inclut donc les domaines de recherches se rapportant à la fusion de senseur, l'odométrie, l'estimation d'état, ainsi que la vision par ordinateur. De plus, il serait nécessaire de pouvoir générer des objectifs à partir du visuel de la cible. Par exemple, si la tâche choisie est de ramasser une tasse, non seulement l'UAM devrait être capable de reconnaître la tasse, mais aussi de générer de manière autonome les conditions de saisies (position et orientation) nécessaires. Le DL représente une piste particulièrement prometteuse pour tous ces sous-systèmes et de nombreux travaux s'y attellent actuellement [4]. Finalement, les aspects physiques de la saisie durant la tâche, incluant sa planification et sa validation, n'ont pas été abordés, mais demeurent un aspect essentiel. Le fait d'être capable de planifier la saisie et de s'assurer qu'elle a été accomplie de manière satisfaisante reste un défi considérable.

Les méthodes développées dans cette thèse sont conçues afin qu'elles soient facilement adaptables aux différents contextes d'utilisations. La taxonomie peut servir de base à un système de classification de tâche et donc de génération d'objectifs pour des missions comportant plusieurs tâches, en plus d'être utile comme outil de simplification et d'identification, lors de la conception initiale des systèmes. La méthode de support à la conception d'une loi de commande permet de supporter sa synthèse en obtenant les gains par optimisation et ce pour tout UAM possédant un bras robotique à 2DDL. Seuls les requis et les paramètres physiques du modèle dynamique sont nécessaires pour obtenir les gains séquencés correspondants à la structure proposée. La loi de commande dans la forme présentée au chapitre 4 ne permet pas de l'appliquer directement à une autre configuration qu'un bras planaire à 2DDL. Cependant, les modèles dynamiques sont développés indépendamment de la configuration du bras. Il ne resterait donc qu'à trouver une structure de séquence qui est fonction des angles du bras robotique et qui permet de stabiliser le système avec la configuration désirée.

Finalement, pour le support à la conception d'un algorithme de sélection d'UAM en fonction de la tâche, la flotte peut être définie comme désiré par le concepteur afin de correspondre à celle utilisée dans un cas réel. Il ne suffit que de calculer à nouveau les étiquettes associées aux scènes et aux tâches de la base de données à l'aide de la méthode par planification de trajectoire. Lors de cette étape, il est possible aussi d'adapter le critère de sélection si désiré. Par exemple, une combinaison de temps de vol ou de minimisation d'énergie pourrait être choisie au lieu de celle présentée, toujours en lien avec l'objectif défini par le concepteur. Une fois les étiquettes obtenues, AMPointNN peut être entraîné tel que présenté afin d'obtenir une méthode de sélection optimale d'UAM selon la tâche et pouvant accomplir cette sélection en une fraction de seconde. Ce travail présente une preuve de concept où les classificateurs n'ont pas été optimisés par rapport à leurs hyperparamètres. Il serait certainement possible

d'augmenter la précision des classificateurs avec une telle optimisation, et aussi en augmentant la taille de la base de données de scènes et tâches réelles provenant du contexte d'utilisation désiré.

L'utilisation de cet algorithme a été développée pour un contexte hypothétique où une flotte comporte à la fois des unités de type UAV qui sont dédiées à l'acquisition des scènes et la prise de décision, ainsi que des UAM effectuant uniquement les tâches de saisie. Le but était de restreindre l'utilisation des UAM, plus lourds et plus énergivores, aux tâches de saisies et de laisser de simples UAV s'occuper de la planification et de l'analyse. Bien que l'algorithme de sélection repose sur des systèmes de perception toujours en recherche active et non couvert par le travail de cette thèse, il est important de commencer à développer des sous-systèmes de guidage comme celui proposé afin de favoriser le développement d'UAM intelligents et autonomes.

Tableau 6.1 Liste des articles de journaux et de conférences.

Article #	Référence
1	C. Coulombe, J.-F. Gamache, O. Barron, G. Descôteaux, D. Saussié et S. Achiche, "Task Taxonomy for Autonomous Unmanned Aerial Manipulator : A Review." Proceedings of the ASME 2020 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. Volume 9 : 40th Computers and Information in Engineering Conference (CIE). Virtual, Online. August 17–19, 2020. V009T09A049. ASME.
2	C. Coulombe, D. Saussié et S. Achiche, "Modeling and gain-scheduled control of an aerial manipulator". *International Journal of Dynamics and Control* (2021). https ://doi.org/10.1007/s40435-021-00807-2
3	C. Coulombe, D. Saussié et S. Achiche, "Selection of Unmanned Aerial Manipulator Configurations for Picking Tasks in Cluttered Environments Using Point Cloud Based Deep Learning", *Journal of Intelligent & Robotic Systems*. Soumis en mai 2021.

6.2 Livrables et contributions

Les travaux accomplis dans cette thèse ont été concentrés sur le développement de méthodes de support à la conception pour sous-systèmes d'UAM dans le contexte de tâches de saisie, identifié comme étant l'objectif de recherche principal. Afin d'accomplir cet objectif, trois sous-objectifs en découlent. Ces sous-objectifs ont été choisis pour leurs importances lors de tâches de saisies d'objets et leurs interactions sont illustrées en figure 6.2.

— **SO1** : Identifier et classifier les tâches pouvant être accomplies par un UAM.
— **SO2** : Développer une méthode de support à la conception d'une loi de commande interne pour un UAM effectuant une tâche de saisie d'objet statique.
— **SO3** : Développer un cadre supportant la conception d'un sous-système de prise de décision pour la planification de multiples tâches de saisies accomplies par une flotte d'UAM.

Les livrables de ce doctorat sont contenus dans des articles de journaux et de conférences. Les articles sont listés à la table 6.1.

Les contributions amenées par ces travaux pour chacun des sous-objectifs de la thèse sont résumées dans la table 6.2.

Tableau 6.2 Contributions de recherche des travaux associés aux sous-objectifs.

SO	Contributions	Articles #
1	-Catégorisation des types de tâches pouvant être accomplies par des UAM selon les capacités nécessaires à l'aide d'une revue de littérature et d'une méthodologie d'acquisition vidéo. -Développement d'une taxonomie permettant la catégorisation des tâches en fonctions de critères hauts-niveaux.	1
2	-Présentation du modèle dynamique d'un UAM, dans une forme permettant sa généralisation à différentes configurations et facilitant l'analyse dynamique. -Démonstration de la structure quadratique comme structure minimale pour la stabilisation d'un UAM avec manipulateur planaire à 2DDL commandé par retour d'état à gains séquencés. -Élaboration d'une structure de loi de commande en position à gains séquencés dont les gains peuvent être obtenus à partir de la synthèse $\mathcal{H}_\infty$ structurée.	2
3	-Développement d'une méthode pour la sélection d'UAM optimal associé à une tâche dans un environnement encombré, basé sur la génération de trajectoire. - Création d'un ensemble de données simulant des tâches de saisies dans des environnements encombrés et étiquetages à partir de la méthode à base de planification de trajectoire. - Utilisation de nuages de points dans un réseau de neuronnes DL pour la prédiction de l'UAM optimal, sans passer par les générations de trajectoires.	3

6.3 Disponibilité des outils

Les outils numériques développés durant ce doctorat sont disponibles en ligne au *https ://github.com/COSIM-Lab* avec leur documentation respective. Ces outils ont été développés afin d'obtenir une preuve de concept des méthodes présentées dans les chapitres 4 et 5 et ne sont pas prêt à une distribution générale. On y retrouve aussi la base de données créée lors du chapitre 5. La liste suivante résume les différentes boîtes de dépôts et les codes inclus.

- *aerial-manip* (*MATLAB/Simulink*) :
  - Outils de synthèse pour le correcteur à gains séquencés.
  - Permet la modification des paramètres du système ainsi que des requis dynamiques.
  - Modèles de simulations dynamiques dans différents contextes (comme défini au chapitre 4).
- *Aerialmanip_moveit* (*Python/ROS/MoveIT*) :
  - Modélisation des UAM sous forme de fichier URDF.
  - Simulations de planifications de trajectoires locales en fonction de la tâche défini et de l'environnement.
  - Génération de tâches et d'environnements pour la banque de données.
- *AMPointNN* (*Python/PyTorch*) :
  - Architectures des réseaux de neuronnes utilisés.
  - Préparation de la banque de données pour l'entrainement
  - Scripts d'entrainements et d'évaluations
- *dataset_AMpointNN Data* : Banque de données utilisées

CHAPITRE 7 CONCLUSION ET RECOMMANDATIONS

Cette thèse a introduit des méthodes de support à la conception appliquées aux UAM devant accomplir des tâches de saisies d'objets. Ces outils s'inscrivent dans le processus général de conception d'un UAM, qui est un système mécatronique. Ce processus de conception doit comporter des éléments de conception intégrée, qui consiste à considérer le système dans son ensemble lors de chaque étape de design, afin d'obtenir un système optimal. Cependant, une des difficultés identifiées lors du processus complet de design est la grande place prise par la conception des aspects techniques des sous-systèmes, si bien que la conception intégrée est laissée de côté. Afin de permettre de consacrer plus de temps à cette étape critique du processus de design, des méthodes de support à la conception sont introduites spécifiquement pour la commande et le guidage. De plus, la taxonomie de tâches permet de classifier les tâches possibles d'un UAM pour des missions complexes, simplifiant ainsi la conception générale et donnant un langage commun à tous. Alors que les méthodes pour la commande et le guidage s'utiliseront lors du design détaillé, la taxonomie peut supporter tout le processus, et en particulier le design conceptuel.

La taxonomie permet de classifier les tâches pouvant être accomplies par un UAM, en demeurant indépendantes de considérations comme la géométrie de la cible ou des capacités physiques du système. Elle ne considère qu'une séparation haut-niveau où l'UAM nécessite des capacités cognitives différentes, représentées par les algorithmes du système. Les différentes tâches sont regroupées tout d'abord en une catégorie de type d'interaction, soit une interaction ponctuelle avec un objet, une interaction continue avec un objet ou encore aucune interaction directe avec l'environnement. Puis de ces premières divisions, elles sont séparées selon la description de la tâche et des conditions environnementales. La plupart des tâches définies par chacune des branches de la taxonomie ont déjà été explorées dans la littérature et sont données en exemples. Une telle taxonomie peut être utilisée avec les autres classifications, comme celle de saisie, afin d'obtenir une classification plus précise. Elle offre un cadre permettant la simplification du problème de conception d'UAM, notamment au niveau de la définition des missions en tâches simples et permet ainsi de résoudre le **SO1**.

La méthode de support à la conception de loi de commande permet de simplifier sa conception pour un UAM possédant un bras robotique à 2DDL planaire. À l'aide d'une structure de correcteur soigneusement choisie et la synthèse $\mathcal{H}_\infty$ structurée, la tâche du concepteur devient simplement une tâche de sélection de cahier des charges dynamiques et d'évaluation des paramètres physique de l'UAM. Elle fournit donc un correcteur à gains séquencés par rapport

aux angles du bras robotique possédant une structure quadratique de manière automatique et répondant au cahier des charges. La loi de commande obtenue est aussi robuste aux variations entre les paramètres physiques réels du système et ceux utilisés pour l'obtention des gains. L'obtention d'une loi de commande est une des étapes critiques et complexes de la conception d'UAM compte tenu des difficultés amenées par l'ajout d'un bras robotique sous un multicoptère déjà instable. Cette méthode de support à la conception permet donc l'obtention d'un correcteur pour le positionnement lors des tâches de saisies, ce qui répond au **SO2**.

La méthode de support à la conception du système de sélection d'UAM en fonction des tâches utilise l'intelligence artificielle pour la gestion de mission complexe. Dans le cas où une flotte d'UAM doit accomplir plusieurs tâches de saisies, sélectionner l'unité la plus apte à accomplir la tâche permettrait d'optimiser le comportement de la flotte dans son ensemble. La méthode proposée emploie le DL afin de résoudre ce problème dans un algorithme rapide pouvant être déployé en temps réel. Elle est aussi formulée de manière à être facilement adaptable par les concepteurs afin de répondre aux besoins de la flotte et du contexte d'application. Par exemple, l'entraînement pourrait être accompli en modifiant les étiquettes provenant selon différents critères d'optimisation tels que le temps de vol ou la consommation d'énergie au lieu de la simplicité de la configuration. Cette méthode pourrait aussi être utilisée à rebours dans un contexte de design préliminaire. Supposant que l'on veut accomplir un ensemble de tâches avec un UAM, on pourrait estimer et optimiser la configuration de l'UAM à résoudre ces tâches avec la méthode se basant sur la planification de trajectoire. La méthode amène une contribution en introduisant un sous-système de guidage pour UAM, permettant d'obtenir des systèmes de plus en plus autonomes, en plus d'offrir un support à leur conception, répondant au **SO3**.

Bien que le fonctionnement des méthodes de support à la conception présentées ait été démontré par des simulations rigoureuses, l'implémentation dans un système réel est essentielle à une éventuelle validation. Leur utilisation par des concepteurs dans un contexte réel permettrait d'obtenir un retour sur leur capacité à supporter la conception d'UAM et pourrait être prise en compte lors de leurs améliorations.

Au niveau du correcteur, celui développé dans cette thèse s'applique aux configurations possédant un bras robotique à 2DDL planaire. Une structure se généralisant à toutes les configurations pourrait être obtenue de manière similaire, en trouvant une loi de commande à gains séquencés adéquate obtenue par la même méthode de synthèse. L'implémentation du contrôleur dans un système réel permettrait de valider son fonctionnement.

La méthode de sélection d'UAM pourrait être utilisée lors de tâches réelles en s'assurant de

conserver les données. Ces dernières pourraient par la suite être utilisées avec de l'apprentissage par transfert (*transfer learning*) afin d'améliorer les performances du classificateur et le spécialiser aux tâches rencontrées fréquemment par la flotte d'UAM. De cette manière, la méthode proposée avec la base de données devient une base servant à déployer la flotte qui par la suite peut se renforcer et devenir de plus en plus efficace. Il serait intéressant par la suite d'appliquer cette méthode à l'utilisation de multiples UAM accomplissant une même tâche en coopération, comme la manipulation d'objets trop lourds ou trop encombrants pour un simple UAM.

L'obtention de robots capables de fonctionner de manière autonome dans des environnements variés est un des objectifs en robotique pour les années futures. Les UAM sont des candidats intéressants dus à leur manœuvrabilité leur permettant d'œuvrer dans des conditions variées. Le développement de méthode de planification de missions devient alors essentiel et le ML pourrait facilement contribuer à leurs développements. L'idée de voir des UAM effectuant des missions complexes de réparations, de sauvetage, ou de récupération avec comme seule information une description générale de la mission semble à portée de main.

RÉFÉRENCES

[1] V. V. Klemas, "Coastal and Environmental Remote Sensing from Unmanned Aerial Vehicles : An Overview," *Journal of Coastal Research*, vol. 315, p. 1260–1267, sept. 2015. [En ligne]. Disponible : http://www.bioone.org/doi/10.2112/JCOASTRES-D-15-00005.1

[2] I. Sa et P. Corke, "Vertical Infrastructure Inspection Using a Quadcopter and Shared Autonomy Control," dans *Field and Service Robotics*, K. Yoshida et S. Tadokoro, édit. Berlin, Heidelberg : Springer Berlin Heidelberg, 2014, vol. 92, p. 219–232. [En ligne]. Disponible : http://link.springer.com/10.1007/978-3-642-40686-7_15

[3] C. Kanellakis et G. Nikolakopoulos, "Survey on Computer Vision for UAVs : Current Developments and Trends," *Journal of Intelligent & Robotic Systems*, vol. 87, n°. 1, p. 141–168, juill. 2017. [En ligne]. Disponible : http://link.springer.com/10.1007/s10846-017-0483-z

[4] H. A. Pierson et M. S. Gashler, "Deep learning in robotics : a review of recent research," *Advanced Robotics*, vol. 31, n°. 16, p. 821–835, août 2017. [En ligne]. Disponible : https://www.tandfonline.com/doi/full/10.1080/01691864.2017.1365009

[5] C. M. Korpela, T. W. Danko et P. Y. Oh, "MM-UAV : Mobile Manipulating Unmanned Aerial Vehicle," *Journal of Intelligent & Robotic Systems*, vol. 65, n°. 1-4, p. 93–101, janv. 2012. [En ligne]. Disponible : http://link.springer.com/10.1007/s10846-011-9591-3

[6] F. Ruggiero, V. Lippiello et A. Ollero, "Aerial Manipulation : A Literature Review," *IEEE Robotics and Automation Letters*, vol. 3, n°. 3, p. 1957–1964, juill. 2018. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/8299552/

[7] W. Bolton, *Mechatronics : a multidisciplinary approach*, sixth edition éd. Harlow, England ; New York : Pearson, 2015.

[8] B. C. Williams, "Interaction-based invention : Designing novel devices from first principles," dans *Expert Systems in Engineering Principles and Applications*, J. Siekmann, G. Goos, J. Hartmanis, G. Gottlob et W. Nejdl, édit. Berlin, Heidelberg : Springer Berlin Heidelberg, 1990, vol. 462, p. 119–134, series Title : Lecture Notes in Computer Science. [En ligne]. Disponible : http://link.springer.com/10.1007/3-540-53104-1_37

[9] T. Tomiyama, V. D'Amelio, J. Urbanic et W. ElMaraghy, "Complexity of Multi-Disciplinary Design," *CIRP Annals*, vol. 56, n°. 1, p. 185–188, 2007. [En ligne]. Disponible : https://linkinghub.elsevier.com/retrieve/pii/S0007850607000467

[10] G. Rzevski, "On conceptual design of intelligent mechatronic systems," *Mechatronics*, vol. 13, n°. 10, p. 1029–1044, déc. 2003. [En ligne]. Disponible : http://www.sciencedirect.com/science/article/pii/S0957415803000412

[11] D. Leidner, C. Borst, A. Dietrich, M. Beetz et A. Albu-Schaffer, "Classifying compliant manipulation tasks for automated planning in robotics," dans *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Hamburg, Germany : IEEE, sept. 2015, p. 1769–1776. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7353607/

[12] S. Hamaza, I. Georgilas, G. Heredia, A. Ollero et T. Richardson, "Design, modeling, and control of an aerial manipulator for placement and retrieval of sensors in the environment," *Journal of Field Robotics*, p. rob.21963, juin 2020. [En ligne]. Disponible : https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21963

[13] J.-J. E. Slotine et W. Li, *Applied nonlinear control*. Englewood Cliffs, N.J : Prentice Hall, 1991.

[14] M. Orsag, C. Korpela, P. Oh et S. Bogdan, "Mission Planning and Control," dans *Aerial Manipulation*. Cham : Springer International Publishing, 2018, p. 209–231, series Title : Advances in Industrial Control. [En ligne]. Disponible : http://link.springer.com/10.1007/978-3-319-61022-1_7

[15] X. Ding, P. Guo, K. Xu et Y. Yu, "A review of aerial manipulation of small-scale rotorcraft unmanned robotic systems," *Chinese Journal of Aeronautics*, vol. 32, n°. 1, p. 200–214, janv. 2019. [En ligne]. Disponible : https://linkinghub.elsevier.com/retrieve/pii/S1000936118301894

[16] J. M. Gomez-de Gabriel, J. M. Gandarias, F. J. Perez-Maldonado, F. J. Garcia-Nuncz, E. J. Fernandez-Garcia et A. J. Garcia-Cerezo, "Methods for Autonomous Wristband Placement with a Search-and-Rescue Aerial Manipulator," dans *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid : IEEE, oct. 2018, p. 7838–7844. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/8594202/

[17] M. Orsag, C. Korpela et P. Oh, "Modeling and Control of MM-UAV : Mobile Manipulating Unmanned Aerial Vehicle," *Journal of Intelligent & Robotic Systems*, vol. 69, n°. 1-4, p. 227–240, janv. 2013. [En ligne]. Disponible : http://link.springer.com/10.1007/s10846-012-9723-4

[18] 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 (IROS)*. Chicago, IL, USA : IEEE, sept. 2014, p. 3417–3422. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/6943038/

[19] P. E. Pounds et A. M. Dollar, "Aerial Grasping from a Helicopter UAV Platform," dans *Experimental Robotics*, O. Khatib, V. Kumar et G. Sukhatme, édit. Berlin, Heidelberg : Springer Berlin Heidelberg, 2014, vol. 79, p. 269–283, series Title : Springer Tracts in Advanced Robotics. [En ligne]. Disponible : http://link.springer.com/10.1007/978-3-642-28572-1_19

[20] V. Ghadiok, J. Goldin et W. Ren, "On the design and development of attitude stabilization, vision-based navigation, and aerial gripping for a low-cost quadrotor," *Autonomous Robots*, vol. 33, nº. 1-2, p. 41–68, août 2012. [En ligne]. Disponible : http://link.springer.com/10.1007/s10514-012-9286-z

[21] C. E. Doyle, J. J. Bird, T. A. Isom, J. C. Kallman, D. F. Bareiss, D. J. Dunlop, R. J. King, J. J. Abbott et M. A. Minor, "An Avian-Inspired Passive Mechanism for Quadrotor Perching," *IEEE/ASME Transactions on Mechatronics*, vol. 18, nº. 2, p. 506–517, avr. 2013. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/6291791/

[22] I. Palunko, R. Fierro et P. Cruz, "Trajectory generation for swing-free maneuvers of a quadrotor with suspended payload : A dynamic programming approach," dans *2012 IEEE International Conference on Robotics and Automation*. St Paul, MN, USA : IEEE, mai 2012, p. 2691–2697. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/6225213/

[23] K. Sreenath, Taeyoung Lee et V. Kumar, "Geometric control and differential flatness of a quadrotor UAV with a cable-suspended load," dans *52nd IEEE Conference on Decision and Control*. Firenze : IEEE, déc. 2013, p. 2269–2274. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/6760219/

[24] S. J. Lee et H. J. Kim, "Autonomous swing-angle estimation for stable slung-load flight of multi-rotor UAVs," dans *2017 IEEE International Conference on Robotics and Automation (ICRA)*. Singapore, Singapore : IEEE, mai 2017, p. 4576–4581. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7989532/

[25] H. Lee et H. J. Kim, "Estimation, Control, and Planning for Autonomous Aerial Transportation," *IEEE Transactions on Industrial Electronics*, vol. 64, nº. 4, p. 3369–3379, avr. 2017. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7534766/

[26] Suseong Kim, Seungwon 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 (IROS)*. Tokyo, Japan : IEEE, nov. 2013, p. 4990–4995. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/6697077/

[27] S. Kim, Hoseong Seo et H. J. Kim, “Operating an unknown drawer using an aerial manipulator,” dans *2015 IEEE International Conference on Robotics and Automation (ICRA)*. Seattle, WA, USA : IEEE, mai 2015, p. 5503–5508. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7139968/

[28] A. Jimenez-Cano, J. Braga, G. Heredia et A. Ollero, “Aerial manipulator for structure inspection by contact from the underside,” dans *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Hamburg, Germany : IEEE, sept. 2015, p. 1879–1884. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7353623/

[29] T. W. Danko et P. Y. Oh, “A hyper-redundant manipulator for Mobile Manipulating Unmanned Aerial Vehicles,” dans *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*. Atlanta, GA, USA : IEEE, mai 2013, p. 974–981. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/6564784/

[30] 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)*. Daejeon, South Korea : IEEE, oct. 2016, p. 3848–3853. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7759566/

[31] G. Garimella et M. Kobilarov, “Towards model-predictive control for aerial pick-and-place,” dans *2015 IEEE International Conference on Robotics and Automation (ICRA)*. Seattle, WA, USA : IEEE, mai 2015, p. 4692–4697. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7139850/

[32] C. Korpela, M. Orsag et P. Oh, “Towards valve turning using a dual-arm aerial manipulator,” dans *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Chicago, IL, USA : IEEE, sept. 2014, p. 3411–3416. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/6943037/

[33] C. Korpela, M. Orsag, M. Pekala et P. Oh, “Dynamic stability of a mobile manipulating unmanned aerial vehicle,” dans *2013 IEEE International Conference on Robotics and Automation (ICRA)*. Karlsruhe, Germany : IEEE, mai 2013, p. 4922–4927. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/6631280/

[34] M. Orsag, C. Korpela, S. Bogdan et P. Oh, “Valve turning using a dual-arm aerial manipulator.” IEEE, mai 2014, p. 836–841. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/6842330/

[35] M. Orsag, C. M. Korpela, S. Bogdan et P. Y. Oh, “Hybrid Adaptive Control for Aerial Manipulation,” *Journal of Intelligent & Robotic Systems*, vol. 73, n°. 1-4, p. 693–707, janv. 2014. [En ligne]. Disponible : http://link.springer.com/10.1007/s10846-013-9936-1

[36] M. Orsag, C. Korpela, S. Bogdan et P. Oh, “Dexterous Aerial Robots—Mobile Manipulation Using Unmanned Aerial Systems,” *IEEE Transactions on Robotics*, vol. 33, n°. 6, p. 1453–1466, déc. 2017. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/8059875/

[37] 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)*. Seattle, WA, USA : IEEE, mai 2015, p. 4014–4020. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7139760/

[38] V. Lippiello et F. Ruggiero, “Exploiting redundancy in Cartesian impedance control of UAVs equipped with a robotic arm,” dans *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vilamoura-Algarve, Portugal : IEEE, oct. 2012, p. 3768–3773. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/6386021/

[39] V. Lippiello et F. Ruggiero, “Cartesian Impedance Control of a UAV with a Robotic Arm,” *IFAC Proceedings Volumes*, vol. 45, n°. 22, p. 704–709, 2012. [En ligne]. Disponible : https://linkinghub.elsevier.com/retrieve/pii/S1474667016336928

[40] R. Mebarki, V. Lippiello et B. Siciliano, “Image-based control for dynamically cross-coupled aerial manipulation,” dans *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Chicago, IL, USA : IEEE, sept. 2014, p. 4827–4833. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/6943248/

[41] R. Mebarki, V. Lippiello et B. Siciliano, “Toward image-based visual servoing for cooperative aerial manipulation,” dans *2015 IEEE International Conference on Robotics and Automation (ICRA)*. Seattle, WA, USA : IEEE, mai 2015, p. 6074–6080. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7140051/

[42] L. R. Buonocore, J. Cacace et V. Lippiello, “Hybrid visual servoing for aerial grasping with hierarchical task-priority control,” dans *2015 23rd Mediterranean Conference on Control and Automation (MED)*. Torremolinos, Malaga, Spain : IEEE, juin 2015, p. 617–623. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7158815/

[43] V. Lippiello, J. Cacace, A. Santamaria-Navarro, J. Andrade-Cetto, M. A. Trujillo, Y. R. Esteves et A. Viguria, "Hybrid Visual Servoing With Hierarchical Task Composition for Aerial Manipulation," *IEEE Robotics and Automation Letters*, vol. 1, n°. 1, p. 259–266, janv. 2016. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7361979/

[44] A. Santamaria-Navarro, V. Lippiello et J. Andrade-Cetto, "Task priority control for aerial manipulation," dans *2014 IEEE International Symposium on Safety, Security, and Rescue Robotics (2014)*. Hokkaido, Japan : IEEE, oct. 2014, p. 1–6. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7017672/

[45] C. D. Bellicoso, L. R. Buonocore, V. Lippiello et B. Siciliano, "Design, modeling and control of a 5-DoF light-weight robot arm for aerial manipulation," dans *2015 23rd Mediterranean Conference on Control and Automation (MED)*. Torremolinos, Spain : IEEE, juin 2015, p. 853–858. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7158852/

[46] A. Suarez, A. E. Jimenez-Cano, V. M. Vega, G. Heredia, A. Rodriguez-Castano et A. Ollero, "Lightweight and human-size dual arm aerial manipulator." IEEE, juin 2017, p. 1778–1784. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7991357/

[47] 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 (ICRA)*. Karlsruhe, Germany : IEEE, mai 2013, p. 4916–4921. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/6631279/

[48] K. Baizid, G. Giglio, F. Pierri, M. A. Trujillo, G. Antonelli, F. Caccavale, A. Viguria, S. Chiaverini et A. Ollero, "Experiments on behavioral coordinated control of an Unmanned Aerial Vehicle manipulator system," dans *2015 IEEE International Conference on Robotics and Automation (ICRA)*. Seattle, WA, USA : IEEE, mai 2015, p. 4680–4685. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7139848/

[49] Hyeonbeom Lee, Hyoin Kim et H. J. Kim, "Path planning and control of multiple aerial manipulators for a cooperative transportation," dans *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Hamburg, Germany : IEEE, sept. 2015, p. 2386–2391. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7353700/

[50] S. Kim, H. Seo, S. Choi et H. J. Kim, "Vision-Guided Aerial Manipulation Using a Multirotor With a Robotic Arm," *IEEE/ASME Transactions on Mechatronics*, vol. 21, n°. 4, p. 1912–1923, août 2016. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7395364/

[51] H. Kim, H. Lee, S. Choi, Y.-k. Noh et H. J. Kim, “Motion planning with movement primitives for cooperative aerial transportation in obstacle environment,” dans *2017 IEEE International Conference on Robotics and Automation (ICRA)*. Singapore, Singapore : IEEE, mai 2017, p. 2328–2334. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7989269/

[52] H. Lee, S. Kim et H. J. Kim, “Control of an aerial manipulator using on-line parameter estimator for an unknown payload,” dans *2015 IEEE International Conference on Automation Science and Engineering (CASE)*. Gothenburg, Sweden : IEEE, août 2015, p. 316–321. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7294098/

[53] J. Mørkeberg Torry-Smith, A. Qamar, S. Achiche, J. Wikander, N. Henrik Mortensen et C. During, “Challenges in Designing Mechatronic Systems,” *Journal of Mechanical Design*, vol. 135, n°. 1, p. 011005, janv. 2013. [En ligne]. Disponible : https://asmedigitalcollection.asme.org/mechanicaldesign/article/doi/10.1115/1.4007929/375680/Challenges-in-Designing-Mechatronic-Systems

[54] L. Wang, W. Shen, H. Xie, J. Neelamkavil et A. Pardasani, “Collaborative conceptual design—state of the art and future trends,” *Computer-Aided Design*, vol. 34, n°. 13, p. 981–996, nov. 2002. [En ligne]. Disponible : http://www.sciencedirect.com/science/article/pii/S0010448501001579

[55] H. Komoto et T. Tomiyama, “Multi-disciplinary system decomposition of complex mechatronics systems,” *CIRP Annals*, vol. 60, n°. 1, p. 191–194, 2011. [En ligne]. Disponible : https://linkinghub.elsevier.com/retrieve/pii/S000785061100103X

[56] J. Gausemeier et S. Moehringer, “VDI 2206- A New Guideline for the Design of Mechatronic Systems,” *IFAC Proceedings Volumes*, vol. 35, n°. 2, p. 785–790, déc. 2002. [En ligne]. Disponible : https://linkinghub.elsevier.com/retrieve/pii/S1474667017340351

[57] M. Mamrot, S. Marchlewitz, J.-P. Nicklas et P. Winzer, “Using systems engineering for a requirement-based design support for autonomous robots,” dans *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. San Diego, CA, USA : IEEE, oct. 2014, p. 3115–3120. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/6974406

[58] J. van Amerongen, “Mechatronic design,” *Mechatronics*, vol. 13, n°. 10, p. 1045–1066, déc. 2003. [En ligne]. Disponible : https://linkinghub.elsevier.com/retrieve/pii/S0957415803000424

[59] Q. Lindsey, D. Mellinger et V. Kumar, “Construction with quadrotor teams,” *Autonomous Robots*, vol. 33, n°. 3, p. 323–336, oct. 2012. [En ligne]. Disponible : http://link.springer.com/10.1007/s10514-012-9305-0

[60] K. Alexis, C. Papachristos, R. Siegwart et A. Tzes, “Robust Model Predictive Flight Control of Unmanned Rotorcrafts,” *Journal of Intelligent & Robotic Systems*, vol. 81, n°. 3-4, p. 443–469, mars 2016. [En ligne]. Disponible : http://link.springer.com/10.1007/s10846-015-0238-7

[61] X. Meng, Y. He, Q. Li, F. Gu, L. Yang, T. Yan et J. Han, “Contact Force Control of an Aerial Manipulator in Pressing an Emergency Switch Process,” dans *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid : IEEE, oct. 2018, p. 2107–2113. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/8593535/

[62] F. Kendoul, “Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems,” *Journal of Field Robotics*, vol. 29, n°. 2, p. 315–378, mars 2012. [En ligne]. Disponible : http://doi.wiley.com/10.1002/rob.20414

[63] H.-M. Huang, “Autonomy Levels For Unmanned Systems (ALFUS) framework, volume I : : terminology version 2.1,” National Institute of Standards and Technology, Gaithersburg, MD, Rapport technique NIST SP 1011-I-2.0, 2008, edition : 0. [En ligne]. Disponible : https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication1011-I-2.0.pdf

[64] M. Cutkosky, “On grasp choice, grasp models, and the design of hands for manufacturing tasks,” *IEEE Transactions on Robotics and Automation*, vol. 5, n°. 3, p. 269–279, juin 1989. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/34763/

[65] T. Feix, J. Romero, H.-B. Schmiedmayer, A. M. Dollar et D. Kragic, “The GRASP Taxonomy of Human Grasp Types,” *IEEE Transactions on Human-Machine Systems*, vol. 46, n°. 1, p. 66–77, févr. 2016. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7243327/

[66] F. Worgotter, E. E. Aksoy, N. Kruger, J. Piater, A. Ude et M. Tamosiunaite, “A Simple Ontology of Manipulation Actions Based on Hand-Object Relations,” *IEEE Transactions on Autonomous Mental Development*, vol. 5, n°. 2, p. 117–134, juin 2013. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/6493410/

[67] J. Liu, F. Feng, Y. C. Nakamura et N. S. Pollard, “A taxonomy of everyday grasps in action,” dans *2014 IEEE-RAS International Conference on Humanoid Robots*. Madrid, Spain : IEEE, nov. 2014, p. 573–580. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7041420/

[68] M. Orsag, C. Korpela, P. Oh et S. Bogdan, “Aerial Manipulator Dynamics,” dans *Aerial Manipulation*. Cham : Springer International Publishing, 2018, p. 123–163. [En ligne]. Disponible : http://link.springer.com/10.1007/978-3-319-61022-1_5

[69] A. Suarez, V. M. Vega, M. Fernandez, G. Heredia et A. Ollero, “Benchmarks for Aerial Manipulation,” *IEEE Robotics and Automation Letters*, vol. 5, n°. 2, p. 2650–2657, avr. 2020. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/8990026/

[70] C. T. Recchiuto et A. Sgorbissa, “Post-disaster assessment with unmanned aerial vehicles : A survey on practical implementations and research approaches,” *Journal of Field Robotics*, vol. 35, n°. 4, p. 459–490, juin 2018. [En ligne]. Disponible : http://doi.wiley.com/10.1002/rob.21756

[71] P. Fabiani, V. Fuertes, A. Piquereau, R. Mampey et F. Teichteil-Königsbuch, “Autonomous flight and navigation of VTOL UAVs : from autonomy demonstrations to out-of-sight flights,” *Aerospace Science and Technology*, vol. 11, n°. 2-3, p. 183–193, mars 2007. [En ligne]. Disponible : https://linkinghub.elsevier.com/retrieve/pii/S1270963806001040

[72] J. Kvarnstrom et P. Doherty, “Automated planning for collaborative UAV systems,” dans *2010 11th International Conference on Control Automation Robotics & Vision*. Singapore, Singapore : IEEE, déc. 2010, p. 1078–1085. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/5707969/

[73] B. Yun, B. M. Chen, K. Y. Lum et T. H. Lee, “Design and implementation of a leader-follower cooperative control system for unmanned helicopters,” *Journal of Control Theory and Applications*, vol. 8, n°. 1, p. 61–68, févr. 2010. [En ligne]. Disponible : http://link.springer.com/10.1007/s11768-010-9188-6

[74] M. Shalaby, C. C. Cossette, J. R. Forbes et J. Le Ny, “Relative Position Estimation in Multi-Agent Systems Using Attitude-Coupled Range Measurements,” *IEEE Robotics and Automation Letters*, vol. 6, n°. 3, p. 4955–4961, juill. 2021. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/9381606/

[75] C. C. Cossette, M. Shalaby, D. Saussie, J. R. Forbes et J. Le Ny, “Relative Position Estimation Between Two UWB Devices With IMUs,” *IEEE Robotics and Automation Letters*, vol. 6, n°. 3, p. 4313–4320, juill. 2021. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/9382085/

[76] D. H. Shim et S. Sastry, “An Evasive Maneuvering Algorithm for UAVs in See-and-Avoid Situations,” dans *2007 American Control Conference*. New York, NY, USA : IEEE, juill. 2007, p. 3886–3891, iSSN : 0743-1619. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/4283147/

[77] S. Lacroix, R. Alami, T. Lemaire, G. Hattenberger et J. Gancet, “Decision Making in Multi-UAVs Systems : Architecture and Algorithms,” dans *Multiple Heterogeneous Unmanned Aerial Vehicles*, A. Ollero et I. Maza, édit. Berlin, Heidelberg : Springer Berlin Heidelberg, 2007, vol. 37, p. 15–48, series Title : Springer Tracts in Advanced Robotics. [En ligne]. Disponible : http://link.springer.com/10.1007/978-3-540-73958-6_2

[78] Y. Tansel İç, M. Yurdakul et B. Dengiz, “Development of a decision support system for robot selection,” *Robotics and Computer-Integrated Manufacturing*, vol. 29, n°. 4, p. 142–157, août 2013. [En ligne]. Disponible : https://linkinghub.elsevier.com/retrieve/pii/S073658451200141X

[79] R. Parameshwaran, S. Praveen Kumar et K. Saravanakumar, “An integrated fuzzy MCDM based approach for robot selection considering objective and subjective criteria,” *Applied Soft Computing*, vol. 26, p. 31–41, janv. 2015. [En ligne]. Disponible : https://linkinghub.elsevier.com/retrieve/pii/S1568494614004785

[80] D. K. Sen, S. Datta, S. K. Patel et S. S. Mahapatra, “Multi-criteria decision making towards selection of industrial robot : Exploration of PROMETHEE II method,” *Benchmarking : An International Journal*, vol. 22, n°. 3, p. 465–487, avr. 2015. [En ligne]. Disponible : https://www.emerald.com/insight/content/doi/10.1108/BIJ-05-2014-0046/full/html

[81] I. Goodfellow, Y. Bengio et A. Courville, *Deep learning*, ser. Adaptive computation and machine learning. Cambridge, Massachusetts : The MIT Press, 2016.

[82] Z. Ghahramani, “Unsupervised Learning,” dans *Advanced Lectures on Machine Learning*, O. Bousquet, U. von Luxburg et G. Rätsch, édit. Berlin, Heidelberg : Springer Berlin Heidelberg, 2004, vol. 3176, p. 72–112, series Title : Lecture Notes in Computer Science. [En ligne]. Disponible : http://link.springer.com/10.1007/978-3-540-28650-9_5

[83] X. Zhu et A. B. Goldberg, “Introduction to Semi-Supervised Learning,” *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 3, n°. 1, p. 1–130, janv. 2009. [En ligne]. Disponible : http://www.morganclaypool.com/doi/abs/10.2200/S00196ED1V01Y200906AIM006

[84] A. Krizhevsky, I. Sutskever et G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” dans *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou et K. Q. Weinberger, édit. Curran Associates, Inc., 2012, p. 1097–1105. [En ligne]. Disponible : http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[85] I. Lenz, H. Lee et A. Saxena, “Deep learning for detecting robotic grasps,” *The International Journal of Robotics Research*, vol. 34, n°. 4-5, p. 705–724, avr. 2015. [En ligne]. Disponible : https://doi.org/10.1177/0278364914549607

[86] J. Yu, K. Weng, G. Liang et G. Xie, “A vision-based robotic grasping system using deep learning for 3D object recognition and pose estimation,” dans *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, déc. 2013, p. 1175–1180.

[87] I. Mariolis, G. Peleka, A. Kargakos et S. Malassiotis, “Pose and category recognition of highly deformable objects using deep learning,” dans *2015 International Conference on Advanced Robotics (ICAR)*, juill. 2015, p. 655–662.

[88] D. Maturana et S. Scherer, “3D Convolutional Neural Networks for landing zone detection from LiDAR,” dans *2015 IEEE International Conference on Robotics and Automation (ICRA)*. Seattle, WA, USA : IEEE, mai 2015, p. 3471–3478. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7139679/

[89] Hanzhang Hu, D. Munoz, J. A. Bagnell et M. Hebert, “Efficient 3-D scene analysis from streaming data,” dans *2013 IEEE International Conference on Robotics and Automation*. Karlsruhe, Germany : IEEE, mai 2013, p. 2297–2304. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/6630888/

[90] R. Q. Charles, H. Su, M. Kaichun et L. J. Guibas, “PointNet : Deep Learning on Point Sets for 3D Classification and Segmentation,” dans *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI : IEEE, juill. 2017, p. 77–85. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/8099499/

[91] C. R. Qi, L. Yi, H. Su et L. J. Guibas, “PointNet++ : Deep Hierarchical Feature Learning on Point Sets in a Metric Space,” *arXiv :1706.02413 [cs]*, juin 2017, arXiv : 1706.02413. [En ligne]. Disponible : http://arxiv.org/abs/1706.02413

[92] A. Mohebbi, S. Achiche et L. Baron, “Design of a Vision Guided Mechatronic Quadrotor System Using Design for Control Methodology,” *Transactions of the Canadian Society for Mechanical Engineering*, vol. 40, n°. 2, p. 201–219, juin 2016. [En ligne]. Disponible : http://www.nrcresearchpress.com/doi/10.1139/tcsme-2016-0016

[93] Q. Li, W. Zhang et L. Chen, “Design for control-a concurrent engineering approach for mechatronic systems design,” *IEEE/ASME Transactions on Mechatronics*, vol. 6, n°. 2, p. 161–169, juin 2001. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/928731/

[94] A. Mohebbi, S. Achiche et L. Baron, “Integrated and concurrent detailed design of a mechatronic quadrotor system using a fuzzy-based particle swarm optimization,”

Engineering Applications of Artificial Intelligence, vol. 82, p. 192–206, juin 2019. [En ligne]. Disponible : https://linkinghub.elsevier.com/retrieve/pii/S0952197619300752

[95] A. Mohebbi, L. Baron, S. Achiche et L. Birglen, "Trends in concurrent, multi-criteria and optimal design of mechatronic systems : A review," dans *Proceedings of the 2014 International Conference on Innovative Design and Manufacturing (ICIDM)*. Montreal, QC, Canada : IEEE, août 2014, p. 88–93. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/6912676/

[96] A. Y. Mersha, S. Stramigioli et R. Carloni, "Variable impedance control for aerial interaction," dans *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Chicago, IL, USA : IEEE, sept. 2014, p. 3435–3440. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/6943041/

[97] S. Bouabdallah et R. Siegwart, "Backstepping and Sliding-mode Techniques Applied to an Indoor Micro Quadrotor," dans *2005 IEEE International Conference on Robotics and Automation*. Barcelona, Spain : IEEE, 2005, p. 2247–2252. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/1570447/

[98] H. Yang et D. Lee, "Dynamics and control of quadrotor with robotic manipulator," dans *2014 IEEE International Conference on Robotics and Automation (ICRA)*. Hong Kong, China : IEEE, mai 2014, p. 5544–5549. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/6907674/

[99] T. Mathworks, "Matlab and Simulink," Natick, Massachusetts, 2020.

[100] P. Apkarian, "Tuning controllers against multiple design requirements," dans *2013 American Control Conference (ACC)*. Washington, DC, USA : IEEE, juin 2013, p. 3888–3893. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/6580433/

[101] P. Gahinet et P. Apkarian, "Decentralized and fixed-structure H_∞ control in MATLAB," dans *2011 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*. Orlando, FL, USA : IEEE, déc. 2011, p. 8205–8210. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/6160298/

[102] H. Lhachemi, D. Saussié et G. Zhu, "A structured H_∞-based optimization approach for integrated plant and self-scheduled flight control system design," *Aerospace Science and Technology*, vol. 45, p. 30–38, sept. 2015. [En ligne]. Disponible : https://linkinghub.elsevier.com/retrieve/pii/S1270963815001182

[103] S. Skogestad et I. Postlethwaite, *Multivariable feedback control : analysis and design*, 2^{e} éd. Hoboken, NJ : John Wiley, 2005.

[104] P. Apkarian et D. Noll, "Nonsmooth H_∞ Synthesis," *IEEE Transactions on Automatic Control*, vol. 51, nº. 1, p. 71–86, janv. 2006. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/1576856/

[105] F. Augugliaro, S. Lupashin, M. Hamer, C. Male, M. Hehn, M. W. Mueller, J. S. Willmann, F. Gramazio, M. Kohler et R. D'Andrea, "The Flight Assembled Architecture installation : Cooperative construction with flying machines," *IEEE Control Systems*, vol. 34, nº. 4, p. 46–64, août 2014. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/6853477/

[106] J. R. Kutia, K. A. Stol et W. Xu, "Canopy sampling using an aerial manipulator : A preliminary study," dans *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, juin 2015, p. 477–484.

[107] A. Carrio, C. Sampedro, A. Rodriguez-Ramos et P. Campoy, "A Review of Deep Learning Methods and Applications for Unmanned Aerial Vehicles," *Journal of Sensors*, vol. 2017, p. 1–13, 2017. [En ligne]. Disponible : https://www.hindawi.com/journals/js/2017/3296874/

[108] "X - The Everyday Robot Project." [En ligne]. Disponible : https://x.company/projects/everyday-robots

[109] R. Rashad, J. B. C. Engelen et S. Stramigioli, "Energy Tank-Based Wrench/Impedance Control of a Fully-Actuated Hexarotor : A Geometric Port-Hamiltonian Approach," dans *2019 International Conference on Robotics and Automation (ICRA)*. Montreal, QC, Canada : IEEE, mai 2019, p. 6418–6424. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/8793939/

[110] G. Zhang, Y. He, B. Dai, F. Gu, L. Yang, J. Han, G. Liu et J. Qi, "Grasp a Moving Target from the Air : System & Control of an Aerial Manipulator," dans *2018 IEEE International Conference on Robotics and Automation (ICRA)*. Brisbane, QLD : IEEE, mai 2018, p. 1681–1687. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/8461103/

[111] F. Suarez-Ruiz et Q.-C. Pham, "A framework for fine robotic assembly," dans *2016 IEEE International Conference on Robotics and Automation (ICRA)*. Stockholm, Sweden : IEEE, mai 2016, p. 421–426. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7487162/

[112] G. A. Korsah, A. Stentz et M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *The International Journal of Robotics Research*, vol. 32, nº. 12, p. 1495–1512, oct. 2013. [En ligne]. Disponible : http://journals.sagepub.com/doi/10.1177/0278364913496484

[113] H. Peng, C. Zhou, H. Hu, F. Chao et J. Li, “Robotic Dance in Social Robotics—A Taxonomy,” *IEEE Transactions on Human-Machine Systems*, vol. 45, n°. 3, p. 281–293, juin 2015. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7042758/

[114] J. W. Firestone, R. Quinones et B. A. Duncan, “Learning from Users : an Elicitation Study and Taxonomy for Communicating Small Unmanned Aerial System States Through Gestures,” dans *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. Daegu, Korea (South) : IEEE, mars 2019, p. 163–171. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/8673010/

[115] A. Bloomfield, Yu Deng, J. Wampler, P. Rondot, D. Harth, M. McManus et N. Badler, “A taxonomy and comparison of haptic actions for disassembly tasks,” dans *IEEE Virtual Reality, 2003. Proceedings.* Los Angeles, CA, USA : IEEE Comput. Soc, 2003, p. 225–231. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/1191143/

[116] C. Kanellakis, M. Terreran, D. Kominiak et G. Nikolakopoulos, “On vision enabled aerial manipulation for multirotors,” dans *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, sept. 2017, p. 1–7.

[117] R. Ritz, M. W. Müller, M. Hehn et R. D’Andrea, “Cooperative quadrocopter ball throwing and catching,” dans *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vilamoura-Algarve, Portugal : IEEE, oct. 2012, p. 4972–4978. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/6385963/

[118] S. Park, J. Lee, J. Ahn, M. Kim, J. Her, G.-H. Yang et D. Lee, “ODAR : Aerial Manipulation Platform Enabling Omnidirectional Wrench Generation,” *IEEE/ASME Transactions on Mechatronics*, vol. 23, n°. 4, p. 1907–1918, août 2018. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/8401328/

[119] A. Zeng, S. Song, J. Lee, A. Rodriguez et T. Funkhouser, “TossingBot : Learning to Throw Arbitrary Objects with Residual Physics,” *arXiv :1903.11239 [cs, stat]*, juill. 2019, arXiv : 1903.11239. [En ligne]. Disponible : http://arxiv.org/abs/1903.11239

[120] S. Tang, V. Wuest et V. Kumar, “Aggressive Flight With Suspended Payloads Using Vision-Based Control,” *IEEE Robotics and Automation Letters*, vol. 3, n°. 2, p. 1152–1159, avr. 2018. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/8258883/

[121] K. Bodie, M. Brunner, M. Pantic, S. Walser, P. Pfändler, U. Angst, R. Siegwart et J. Nieto, “An Omnidirectional Aerial Manipulation Platform for Contact-Based Inspection,” *arXiv :1905.03502 [cs]*, juill. 2019, arXiv : 1905.03502. [En ligne]. Disponible : http://arxiv.org/abs/1905.03502

[122] H. Jiang, M. T. Pope, E. W. Hawkes, D. L. Christensen, M. A. Estrada, A. Parlier, R. Tran et M. R. Cutkosky, “Modeling the dynamics of perching with opposed-grip

mechanisms," dans *2014 IEEE International Conference on Robotics and Automation (ICRA)*. Hong Kong, China : IEEE, mai 2014, p. 3102–3108. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/6907305/

[123] 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)*. Valletta, Malta : IEEE, juill. 2017, p. 1385–1391. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7984312/

[124] DJI, "DJI - Introducing the Spreading Wings S1000." [En ligne]. Disponible : https://www.youtube.com/watch?v=XU4rEbCxSW0

[125] I. Sa, M. Kamel, M. Burri, M. Bloesch, R. Khanna, M. Popovic, J. Nieto et R. Siegwart, "Build Your Own Visual-Inertial Drone : A Cost-Effective and Open-Source Autonomous Drone," *IEEE Robotics & Automation Magazine*, vol. 25, n°. 1, p. 89–103, mars 2018. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/8233182/

[126] S. Jung, S. Hwang, H. Shin et D. H. Shim, "Perception, Guidance, and Navigation for Indoor Autonomous Drone Racing Using Deep Learning," *IEEE Robotics and Automation Letters*, vol. 3, n°. 3, p. 2539–2544, juill. 2018. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/8299437/

[127] T. Baltovski, S. Nokleby et R. Pop-Iliev, "Towards Performing Remote Manipulation Using an Autonomous Aerial Vehicle," dans *CCToMM Mechanisms, Machines, and Mechatronics (M3) Symposium, 2015*, Ottawa, Ontario, Canada, mai 2015.

[128] S. Bouabdallah, P. Murrieri et R. Siegwart, "Design and control of an indoor micro quadrotor," dans *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04.* New Orleans, LA, USA : IEEE, 2004, p. 4393–4398 Vol.5. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/1302409/

[129] A. Tayebi et S. McGilvray, "Attitude stabilization of a VTOL quadrotor aircraft," *IEEE Transactions on Control Systems Technology*, vol. 14, n°. 3, p. 562–571, mai 2006. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/1624481/

[130] P. Pounds, R. Mahony et P. Corke, "Modelling and control of a large quadrotor robot," *Control Engineering Practice*, vol. 18, n°. 7, p. 691–699, juill. 2010. [En ligne]. Disponible : http://linkinghub.elsevier.com/retrieve/pii/S0967066110000456

[131] T. Madani et A. Benallegue, "Backstepping Control for a Quadrotor Helicopter," dans *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Beijing, China : IEEE, oct. 2006, p. 3255–3260. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/4058900/

[132] I. Palunko et R. Fierro, “Adaptive Control of a Quadrotor with Dynamic Changes in the Center of Gravity,” *IFAC Proceedings Volumes*, vol. 44, n°. 1, p. 2626–2631, janv. 2011. [En ligne]. Disponible : http://linkinghub.elsevier.com/retrieve/pii/S1474667016440097

[133] B. L. Stevens, F. L. Lewis et E. N. Johnson, *Aircraft control and simulation : dynamics, controls design, and autonomous systems*, third edition éd. Hoboken, N.J : John Wiley & Sons, 2016.

[134] M. Fanni et A. Khalifa, “A New 6-DOF Quadrotor Manipulation System : Design, Kinematics, Dynamics, and Control,” *IEEE/ASME Transactions on Mechatronics*, vol. 22, n°. 3, p. 1315–1326, juin 2017. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7875412/

[135] D. Bazylev, K. Zimenko, A. Margun, A. Bobtsov et A. Kremlev, “Adaptive control system for quadrotor equiped with robotic arm,” dans *2014 19th International Conference on Methods and Models in Automation and Robotics (MMAR)*. Miedzyzdroje, Poland : IEEE, sept. 2014, p. 705–710. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/6957440/

[136] S. Bouabdallah, A. Noth et R. Siegwart, “PID vs LQ control techniques applied to an indoor micro quadrotor,” dans *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3. Sendai, Japan : IEEE, 2004, p. 2451–2456. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/1389776/

[137] H. Bouadi, S. Simoes Cunha, A. Drouin et F. Mora-Camino, “Adaptive sliding mode control for quadrotor attitude stabilization and altitude tracking,” dans *2011 IEEE 12th International Symposium on Computational Intelligence and Informatics (CINTI)*. Budapest, Hungary : IEEE, nov. 2011, p. 449–455. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/6108547/

[138] F. Caccavale, G. Giglio, G. Muscio et F. Pierri, “Adaptive control for UAVs equipped with a robotic arm,” *IFAC Proceedings Volumes*, vol. 47, n°. 3, p. 11 049–11 054, 2014. [En ligne]. Disponible : http://linkinghub.elsevier.com/retrieve/pii/S1474667016433719

[139] S. Kannan, M. Alma, M. A. Olivares-Mendez et H. Voos, “Adaptive control of Aerial Manipulation Vehicle,” dans *2014 IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*. Batu Ferringhi, Malaysia : IEEE, nov. 2014, p. 273–278. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7072729/

[140] G. Antonelli et E. Cataldi, “Adaptive control of arm-equipped quadrotors. Theory and simulations,” dans *2014 22nd Mediterranean Conference of Control and Automation*

(MED). Palermo, Italy : IEEE, juin 2014, p. 1446–1451. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/6961579/

[141] S. Rajappa, C. Masone, H. H. Bulthoff et P. Stegagno, "Adaptive Super Twisting Controller for a quadrotor UAV," dans *2016 IEEE International Conference on Robotics and Automation (ICRA)*. Stockholm, Sweden : IEEE, mai 2016, p. 2971–2977. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7487462/

[142] M. Sharifi et H. Sayyaadi, "Nonlinear robust adaptive Cartesian impedance control of UAVs equipped with a robot manipulator," *Advanced Robotics*, vol. 29, n°. 3, p. 171–186, févr. 2015. [En ligne]. Disponible : http://www.tandfonline.com/doi/abs/10.1080/01691864.2014.1002529

[143] P. Gahinet et P. Apkarian, "Structured H_∞ Synthesis in MATLAB," *IFAC Proceedings Volumes*, vol. 44, n°. 1, p. 1435–1440, janv. 2011. [En ligne]. Disponible : http://linkinghub.elsevier.com/retrieve/pii/S1474667016438115

[144] W. J. Rugh et J. S. Shamma, "Research on gain scheduling," *Automatica*, vol. 36, n°. 10, p. 1401–1425, oct. 2000. [En ligne]. Disponible : http://linkinghub.elsevier.com/retrieve/pii/S0005109800000583

[145] H. Lhachemi, D. Saussié et G. Zhu, "Hidden Coupling Terms Inclusion in Gain-Scheduling Control Design : Extension of an Eigenstructure Assignment-Based Technique," *IFAC-PapersOnLine*, vol. 49, n°. 17, p. 403–408, 2016. [En ligne]. Disponible : http://linkinghub.elsevier.com/retrieve/pii/S2405896316315427

[146] H. Lhachemi, D. Saussié et G. Zhu, "Explicit hidden coupling terms handling in gain-scheduling control design via eigenstructure assignment," *Control Engineering Practice*, vol. 58, p. 1–11, janv. 2017. [En ligne]. Disponible : http://linkinghub.elsevier.com/retrieve/pii/S0967066116301964

[147] D.-T. Nguyen, D. Saussie et L. Saydy, "Fault-Tolerant Control of a Hexacopter UAV based on Self-Scheduled Control Allocation," dans *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*. Dallas, TX : IEEE, juin 2018, p. 385–393. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/8453440/

[148] T. Nguyen, D. Saussie et L. Saydy, "Design and Experimental Validation of Robust Self-Scheduled Fault-Tolerant Control Laws for a Multicopter UAV," *IEEE/ASME Transactions on Mechatronics*, p. 1–1, 2020. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/9279322/

[149] J. J. Craig, *Introduction to robotics : mechanics and control*, 3e éd., ser. Pearson education international. Upper Saddle River, NJ : Pearson, Prentice Hall, 2005, oCLC : 249488662.

[150] M. W. Spong, S. Hutchinson et M. Vidyasagar, *Robot modeling and control*. Hoboken, NJ : John Wiley & Sons, 2006, oCLC : ocm61458391.

[151] A. Andry, E. Shapiro et J. Chung, "Eigenstructure Assignment for Linear Systems," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-19, nº. 5, p. 711–729, sept. 1983. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/4102853/

[152] B. Yang, Y. He, J. Han et G. Liu, "Rotor-Flying Manipulator : Modeling, Analysis, and Control," *Mathematical Problems in Engineering*, vol. 2014, p. 1–13, 2014. [En ligne]. Disponible : http://www.hindawi.com/journals/mpe/2014/492965/

[153] C. Coulombe, J.-F. Gamache, O. Barron, G. Descôteaux, D. Saussié et S. Achiche, "Task Taxonomy for Autonomous Unmanned Aerial Manipulator : A Review," dans *Volume 9 : 40th Computers and Information in Engineering Conference (CIE)*. Virtual, Online : American Society of Mechanical Engineers, août 2020, p. V009T09A049. [En ligne]. Disponible : https://asmedigitalcollection.asme.org/IDETC-CIE/proceedings/IDETC-CIE2020/83983/Virtual,%20Online/1090096

[154] S. Kim, H. Seo, J. Shin et H. J. Kim, "Cooperative Aerial Manipulation Using Multirotors With Multi-DOF Robotic Arms," *IEEE/ASME Transactions on Mechatronics*, vol. 23, nº. 2, p. 702–713, avr. 2018. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/8253830/

[155] Y.-S. L.-K. Cio, M. Raison, C. Leblond Menard et S. Achiche, "Proof of Concept of an Assistive Robotic Arm Control Using Artificial Stereovision and Eye-Tracking," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 27, nº. 12, p. 2344–2352, déc. 2019. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/8887462/

[156] S. Y. Choi et D. Cha, "Unmanned aerial vehicles using machine learning for autonomous flight ; state-of-the-art," *Advanced Robotics*, vol. 33, nº. 6, p. 265–277, mars 2019. [En ligne]. Disponible : https://www.tandfonline.com/doi/full/10.1080/01691864.2019.1586760

[157] K. Xu, H. Huang, Y. Shi, H. Li, P. Long, J. Caichen, W. Sun et B. Chen, "Autoscanning for coupled scene reconstruction and proactive object analysis," *ACM Transactions on Graphics*, vol. 34, nº. 6, p. 1–14, nov. 2015. [En ligne]. Disponible : https://dl.acm.org/doi/10.1145/2816795.2818075

[158] Zhirong Wu, S. Song, A. Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang et J. Xiao, "3D ShapeNets : A deep representation for volumetric shapes," dans *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Boston, MA, USA : IEEE, juin 2015, p. 1912–1920. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7298801/

[159] R. Klokov et V. Lempitsky, "Escape from Cells : Deep Kd-Networks for the Recognition of 3D Point Cloud Models," *arXiv :1704.01222 [cs]*, oct. 2017, arXiv : 1704.01222. [En ligne]. Disponible : http://arxiv.org/abs/1704.01222

[160] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein et J. M. Solomon, "Dynamic Graph CNN for Learning on Point Clouds," *arXiv :1801.07829 [cs]*, juin 2019, arXiv : 1801.07829. [En ligne]. Disponible : http://arxiv.org/abs/1801.07829

[161] Z. Zhang, B.-S. Hua, D. W. Rosen et S.-K. Yeung, "Rotation Invariant Convolutions for 3D Point Clouds Deep Learning," *arXiv :1908.06297 [cs]*, août 2019, arXiv : 1908.06297. [En ligne]. Disponible : http://arxiv.org/abs/1908.06297

[162] Y. Shen, C. Feng, Y. Yang et D. Tian, "Mining Point Cloud Local Structures by Kernel Correlation and Graph Pooling," *arXiv :1712.06760 [cs]*, avr. 2018, arXiv : 1712.06760. [En ligne]. Disponible : http://arxiv.org/abs/1712.06760

[163] M. Bellone, G. Reina, L. Caltagirone et M. Wahde, "Learning Traversability From Point Clouds in Challenging Scenarios," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, n°. 1, p. 296–305, janv. 2018. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/8168386/

[164] D. Coleman, I. A. Sucan, S. Chitta et N. Correll, "Reducing the Barrier to Entry of Complex Robotic Software : a MoveIt ! Case Study," *Journal of Software Engineering for Robotics*, vol. 5, n°. 1, p. 3–16, mai 2014, publisher : Università degli studi di Bergamo. [En ligne]. Disponible : https://aisberg.unibg.it//handle/10446/87657

[165] L. Blanchet, S. Achiche, Q. Docquier, P. Fisette et M. Raison, "A procedure to optimize the geometric and dynamic designs of assistive upper limb exoskeletons," *Multibody System Dynamics*, vol. 51, n°. 2, p. 221–245, févr. 2021. [En ligne]. Disponible : http://link.springer.com/10.1007/s11044-020-09766-6

[166] J. Kuffner et S. LaValle, "RRT-connect : An efficient approach to single-query path planning," dans *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 2. San Francisco, CA, USA : IEEE, 2000, p. 995–1001. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/844730/

[167] L. Yi, V. G. Kim, D. Ceylan, I.-C. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer et L. Guibas, "A scalable active framework for region annotation in 3D shape collections," *ACM Transactions on Graphics*, vol. 35, n°. 6, p. 1–12, nov. 2016. [En ligne]. Disponible : https://dl.acm.org/doi/10.1145/2980179.2980238

[168] A. Handa, V. Patraucean, S. Stent et R. Cipolla, “SceneNet : An annotated model generator for indoor scene understanding,” dans *2016 IEEE International Conference on Robotics and Automation (ICRA)*. Stockholm, Sweden : IEEE, mai 2016, p. 5737–5743. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7487797/

[169] K. Shoemake, “Uniform Random Rotations,” dans *Graphics Gems III (IBM Version)*. Elsevier, 1992, p. 124–132. [En ligne]. Disponible : https://linkinghub.elsevier.com/retrieve/pii/B9780080507552500361

[170] A. Anand, H. S. Koppula, T. Joachims et A. Saxena, “Contextually guided semantic labeling and search for three-dimensional point clouds,” *The International Journal of Robotics Research*, vol. 32, n°. 1, p. 19–34, janv. 2013. [En ligne]. Disponible : http://journals.sagepub.com/doi/10.1177/0278364912461538

[171] H. Koppula, A. Anand, T. Joachims et A. Saxena, “Semantic labeling of 3d point clouds for indoor scenes,” dans *Advances in neural information processing systems*, 2011, p. 244–252.

[172] K. Lai, L. Bo, X. Ren et D. Fox, “A large-scale hierarchical multi-view RGB-D object dataset,” dans *2011 IEEE International Conference on Robotics and Automation*. Shanghai, China : IEEE, mai 2011, p. 1817–1824. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/5980382/

[173] K. Lai, L. Bo et D. Fox, “Unsupervised feature learning for 3D scene labeling,” dans *2014 IEEE International Conference on Robotics and Automation (ICRA)*. Hong Kong, China : IEEE, mai 2014, p. 3050–3057. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/6907298/

[174] A. Ecins, C. Fermuller et Y. Aloimonos, “Seeing Behind the Scene : Using Symmetry to Reason About Objects in Cluttered Environments,” dans *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid : IEEE, oct. 2018, p. 7193–7200. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/8593822/

[175] T. Solund, A. G. Buch, N. Kruger et H. Aanas, “A Large-Scale 3D Object Recognition Dataset,” dans *2016 Fourth International Conference on 3D Vision (3DV)*. Stanford, CA, USA : IEEE, oct. 2016, p. 73–82. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7785079/

[176] N. V. Chawla, K. W. Bowyer, L. O. Hall et W. P. Kegelmeyer, “SMOTE : Synthetic Minority Over-sampling Technique,” *Journal of Artificial Intelligence Research*, vol. 16, p. 321–357, juin 2002. [En ligne]. Disponible : https://www.jair.org/index.php/jair/article/view/10302

[177] X. Yue, B. Wu, S. A. Seshia, K. Keutzer et A. L. Sangiovanni-Vincentelli, “A LiDAR Point Cloud Generator : from a Virtual World to Autonomous Driving,” dans *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*. Yokohama Japan : ACM, juin 2018, p. 458–464. [En ligne]. Disponible : https://dl.acm.org/doi/10.1145/3206025.3206080

[178] Y. Zhou et O. Tuzel, “VoxelNet : End-to-End Learning for Point Cloud Based 3D Object Detection,” dans *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT, USA : IEEE, juin 2018, p. 4490–4499. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/8578570/

[179] S. Ioffe et C. Szegedy, “Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariate Shift,” *arXiv :1502.03167 [cs]*, mars 2015, arXiv : 1502.03167. [En ligne]. Disponible : http://arxiv.org/abs/1502.03167

[180] D. P. Kingma et J. Ba, “Adam : A Method for Stochastic Optimization,” *arXiv :1412.6980 [cs]*, janv. 2017, arXiv : 1412.6980. [En ligne]. Disponible : http://arxiv.org/abs/1412.6980

[181] I. Loshchilov et F. Hutter, “Decoupled Weight Decay Regularization,” *arXiv :1711.05101 [cs, math]*, janv. 2019, arXiv : 1711.05101. [En ligne]. Disponible : http://arxiv.org/abs/1711.05101

[182] Sungjoon Choi, Q.-Y. Zhou et V. Koltun, “Robust reconstruction of indoor scenes,” dans *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Boston, MA, USA : IEEE, juin 2015, p. 5556–5565. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7299195/

[183] J. Park, Q.-Y. Zhou et V. Koltun, “Colored Point Cloud Registration Revisited,” dans *2017 IEEE International Conference on Computer Vision (ICCV)*. Venice : IEEE, oct. 2017, p. 143–152. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/8237287/

[184] Q.-Y. Zhou, J. Park et V. Koltun, “Open3D : A Modern Library for 3D Data Processing,” *arXiv :1801.09847 [cs]*, janv. 2018, arXiv : 1801.09847. [En ligne]. Disponible : http://arxiv.org/abs/1801.09847