COMPTE RENDU PFE

Raffaello D'Andrea et ses jouets !

Commande d'un drone à la baguette !

Florian Dietrich

Damien Dubost

Résumé

Ce rapport présente et analyse le travail que nous avons réalisé pour contrôler et commander un drone Parrot repéré par un système de caméra infrarouge. L'objectif principal de ce travail a été de pouvoir piloter ce quadcopter grâce à une baguette repérée dans l'espace par la même méthode, c'est grâce à elle que les consignes de vol seront générées.

Après avoir étudié le protocole de communication et les différentes méthodes de commande du drone il était nécessaire de déterminer son état. La prise en main du logiciel Cortex avec l'aide de monsieur Galbrun a permis de reconstruire avec précision l'état de l'engin.

La procédure de travail a été linéaire, les algorithmes de commande étaient très simple dans un premier temps afin de vérifier que chaque étape du travail fonctionnait correctement. Dans un second temps ces algorithmes étaient complexifiés pour pouvoir piloter plusieurs variables de façon simultanées.

Finalement, le résultat n'est pas forcément à la hauteur des espérances que nous avions lorsque le projet a débuté. Mais un prototype fonctionnel de commande est disponible, il serait intéressant de proposer ce travail à un nouveau groupe d'étudiant l'année prochaine. D'ici là peut être que quelqu'un sera parvenu à modifier le firmware des drones Parrot et qu'il autorisera plus de liberté aux utilisateurs.

Abstract

This report presents and analyses the work made to control and command a Parrot UAV. Its position is determined with infrared cameras. The aim of this study was to drive this quadcopter with a wand and its position is measured by the same motion capture method. Flight commands are then generated according to these positions.

After studying the communication protocol and the various command method, it was necessary to build his state. A tutorial showed by Mr Galbrun allowed the group to get the state vector of the system with a lot of accuracy.

The work procedure was straight forward, control algorithms were very simple in the first place in order to check if they were working correctly. Then they were enhanced so that it was possible to drive several inputs simultaneously.

Finally, the result is not exactly what was expected when the project started. However a working prototype of control is available, it seems to be interesting to propose this subject to a new group of students next year. By then someone might have hacked the firmware allowing more freedom to the users.

Introduction

L'ENSEM dispose de matériel très performant mais sous-utilisé. Nous souhaitons pallier à ce problème en travaillant avec les *Quadcoptères* Parrot et le système de motion tracking par caméra infrarouge. Ce sujet de PFE a été motivé par une vidéo montrant les applications possibles de ce genre de systèmes. De plus nous souhaitons réaliser des stages liant l'aéronautique et l'automatique et ce sujet d'avoir un premier contact avec ce milieu.

Notre projet consiste à modéliser et stabiliser le drone, mesurer son état grâce au système de caméras. Une fois le modèle et la commande établis nous souhaitons diriger le drone à l'aide d'une baguette. Néanmoins, en pratique, nous n'utiliserons qu'un modèle cinématique simple et nous ne chercherons pas à être dans une situation de vol stationnaire mais plutôt en poursuite de trajectoire avec la baguette.

I - Présentation du problème

1 - Drone Parrot

a. Caractéristiques techniques

La société Parrot est une société française spécialisée en électronique et en traitement du signal et est depuis peu connue du grand public grâce à la commercialisation de robots connectés tels que le drone Parrot AR.Drone qui a eu un grand succès, drone que nous avons utilisé lors de notre projet de fin d'étude et qui sera présenté par la suite.

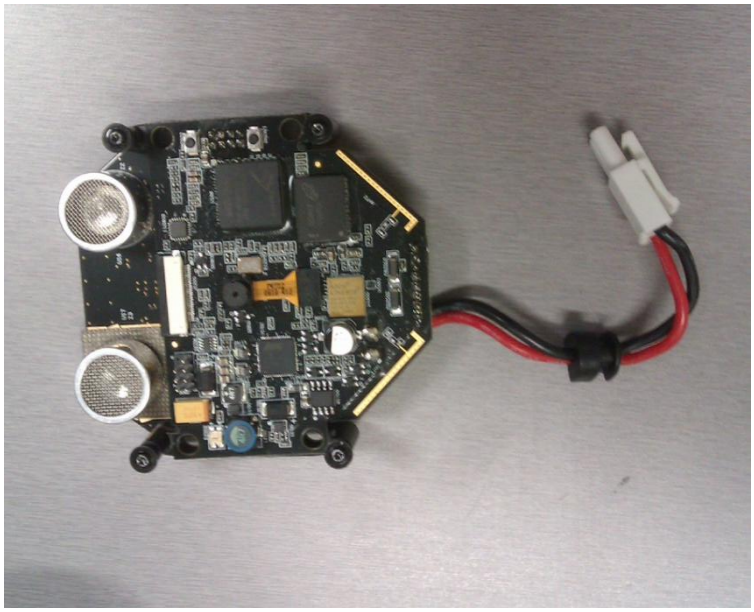
Logo de la société Parrot qui a commercialisé l'AR.Drone.

L'AR.Drone est un quadrocopter, c'est-à-dire que cet appareil dispose de quatre hélices servant à propulser l'engin et à le diriger. Il est fourni avec deux coques différentes : une adaptée pour le vol en plein air, et une pour le vol à l'intérieur. Cette dernière coque est en polystyrène et protège les quatre hélices de tous chocs. Ces coques sont fixés au drone lui-même via un aimant relativement puissant, ce qui permet à la coque de ce maintenir même en cas de chute du drone ou de choc violent.

AR Drone sans coque et sans batterie.

Cet appareil est équipé de deux caméras : une frontale au bout d'un petit mat, et une ventrale. Comme nous l'expliquerons plus tard, ces cameras peuvent intervenir dans la stabilisation du drone.

Le drone est également équipé de capteurs internes comme des accéléromètres et des gyroscopes pour mesurer les différentes accélérations linéaires et angulaires. Ces capteurs permettent au drone de faire du vol stationnaire et donnent des informations sur les déplacements et la rotation du drone. Il dispose également de capteurs ultrasons ventraux pour mesurer sa distance par rapport au sol. Le drone embarque également quatre LED disposées aux bouts des bras et il est possible de les faire clignoter et de les faire changer de couleurs.



Carte électronique d'un drone Parrot.

La communication avec le drone s'effectue en WiFi via le protocole UDP. Ce protocole internet est simple car il n'utilise pas d'acquittement, c'est-à-dire que tout paquet envoyé qui n'est pas correctement reçu n'est pas renvoyé. Ce protocole est couramment utilisé dans des applications temps réel, ou l'information doit être la plus fraîche possible. Le drone diffuse donc un réseau wifi sur lequel il est possible de se connecter avec n'importe quel type d'appareil.

Point d'accès wifi d'un drone Parrot.

Comme la communication avec le drone est assez simple d'un point de vue technique, il est possible de transmettre des données au drone avec un grand nombre de langages informatiques et donc des terminaux différents. Il est ainsi possible de réaliser aussi bien des applications pour smartphones Android que pour des ordinateurs. Tout ceci a été mis en place dans le but de permettre la réalisation d'applications à caractère ludique pour tablettes et smartphones, et ainsi toucher un public large.

A bord du drone est installé un système Linux minimaliste propre à Parrot. Il est possible d'accéder à ce système via une console en ligne de commandes et de visualiser certains fichiers comme le journal de bord, l'historique des commandes envoyées ou encore la configuration du

drone. On utilise pour cela le logiciel PuTTY après s'être connecté au drone en WiFi. La procédure détaillée dans le manuel d'utilisation en annexe.

```
BusyBox v1.14.0 () built-in shell (ash)
Enter 'help' for a list of built-in commands.

# ls
bin         etc         home        mnt         sbin        update
data        factory     lib         proc        sys         usr
dev         firmware    licenses    root        tmp         var
#
```

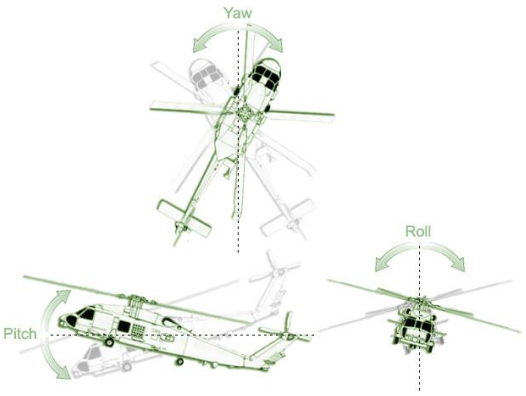
Racine du système de fichier du drone, vu avec PuTTY.

L'AR.Drone de Parrot est donc un système complet disposant de tout un arsenal de capteurs et de sous-système permettant d'en faire un objet d'étude très complet.

b. Dynamique de vol d'un quadcoptère

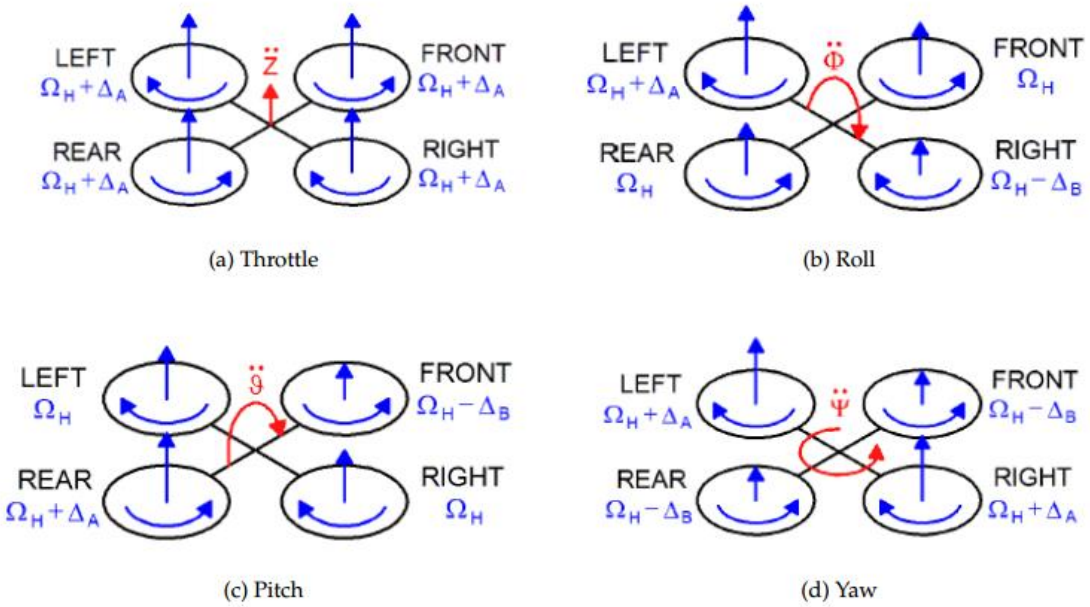
Un quadcoptère possède une dynamique de vol particulière que nous allons brièvement rappeler ici. Comme dit précédemment, le drone que nous utilisons possède quatre hélices donc quatre moteurs indépendants. Il est donc soumis au couple de chacun des rotors et à son propre poids et si l'on néglige les frottements dus à l'air, seules ces forces s'appliquent. Le problème des engins à voilure mobile, c'est-à-dire des hélicoptères et autres quadcoptères, c'est qu'il faut à tout moment compenser la rotation d'un rotor par celle d'un autre. Typiquement sur un hélicoptère, il faut compenser la rotation du rotor principal avec celle du rotor arrière pour faire du vol stationnaire. C'est le même principe avec un quadcoptère, les rotors se compensent deux à deux en tournant en sens inverse. Puis en jouant sur les différentes vitesses de rotations, donc les différents couples, on peut perturber l'équilibre des forces et ainsi induire un déplacement à l'appareil.

En aéronautique, on caractérise un appareil par ces six degrés de libertés que sont les 3 positions classiques (abscisse, ordonnée et altitude) et 3 angles : le tangage, le roulis et le lacet en langue de Molière et respectivement le pitch, roll et yaw en langue de Shakespeare. Ces angles sont définis comme la rotation autour de chacun des 3 axes de la manière suivante :



Visualisation des angles en aéronautique.

On peut alors, en jouant sur les couples des différents moteurs, induire différents mouvements. Dans le cas d'un quadcoptère, le schéma suivant résume les différents mouvements élémentaires possibles. Ces mouvements peuvent et seront bien entendu couplés en situation réelle.



Différents mouvements élémentaires possible pour un quadcoptère.

Par exemple en augmentant simultanément le couple des quatre moteurs on provoque une poussée strictement verticale qui a pour effet de contrecarrer la gravité, et donc de faire s'élever le drone. Ensuite en déséquilibrant les moteurs deux à deux, on induit du yaw, c'est-à-dire que le drone tourne sur lui-même à cause des écarts de couple. Les effets de pitch et de roll sont induits par le déséquilibre au sein d'un couple de moteur, en laissant l'autre couple équilibré.

c. Commande du drone

Comme expliqué précédemment, le drone communique à l'aide d'une connexion wifi. Il envoie des informations sur ce réseau, que cela soit de la vidéo ou des données de vol. Mais il est aussi et surtout capable de recevoir des commandes, celles-ci suivant une syntaxe particulière. Ces commandes, appelées *commandes AT* permettent aussi bien de configurer le drone que de le commander (décollage, déplacement, atterrissage …). On peut transmettre plusieurs commandes AT dans un seul paquet UDP tant que la taille limite autorisée n'est pas atteinte. Cependant, il n'est pas possible de séparer une commande en plusieurs paquets.

Chaque commande nécessite l'envoi d'un numéro de séquence. Ce numéro permet d'ordonner les commandes, et ainsi le drone peut traiter les commandes dans l'ordre. Ce paramètre est très important à respecter pour que le drone se comporte comme souhaité. Il doit donc être incrémenté après chaque commande envoyée au drone.

La configuration du drone se fait avec la commande *AT*CONFIG*. Cette commande permet d'effectuer les réglages de base concernant par exemple l'altitude maximale, les angles maximales, la

vitesse ascensionnelle maximale et d'autres qui sont détaillés dans le guide du développeur. Par exemple la commande : « AT*CONFIG=%d,"control:altitude_max,1000 » permet de fixer l'altitude maximale du drone à un mètre.

Pour contrôler le décollage et l'atterrissage, on utilise la commande *AT*REF* avec deux arguments différents. L'argument de la commande AT*REF est un entier codé sur 32 bit. On ne se sert que du bit numéro 9 pour commander le décollage ou l'atterrissage. En mettant le bit à 1 on lui ordonne de décoller, et en le mettant à 0 l'ordre lui est donner d'atterrir. On transforme ensuite le nombre sur 32 bits obtenue en un entier qui sera transmis au drone.

Ordre	Valeur du bit 9	Entier à transmettre
Décoller	1	290718208
Atterrir	0	290717696

*Construction de la commande AT*REF*

Pour contrôler le déplacement du drone dans l'espace il faut utiliser la commande *AT*PCMD*. Celle-ci permet de préciser les commandes suivantes, qui seront alors interprétées par le drone :

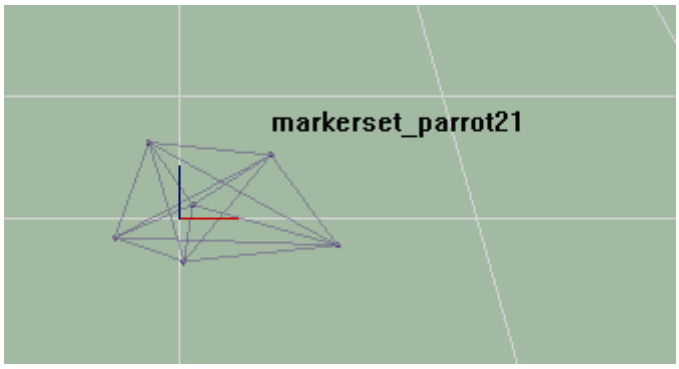
- angle de roll (roulis gauche/droite)
- angle de pitch (tangage avant/arrière)
- vitesse ascensionnelle
- vitesse angulaire de lacet ou vitesse yaw (rotation autour de l'axe verticale)

Ces quatre commande sont des entiers dans l'intervalle [-1 ;1], avec au bornes les vitesses et angles maximales définis dans la configuration. La description complète de cette fonction est disponible dans le guide du développeur et nous ne nous attarderons pas dessus. Il faut cependant garder à l'esprit le fait que les commandes portent sur deux angles et deux vitesses. Il n'est donc pas possible de commander le drone moteur par moteur, cela va influencer la manière dont la commande va être construite.

2 - Caméra IR

La salle SAMI est équipée d'un système complet de caméras infrarouges permettant de suivre des points dans l'espace entre les caméras. Ces points sont repérés par les caméras grâce à des capteurs, qui sont de petites boules argentées qui sont conçues pour réfléchir la lumière émises par les caméras.

Les données récupérées par les caméras sont ensuite traitées par un ordinateur via le logiciel Cortex. Ce logiciel permet entre autre de distinguer différents objets en se servant des capteurs qui y sont posés. En effet, suivant la disposition des points sur l'objet réel, le système est capable de le reconnaitre en le comparant à un modèle préalablement entré dans le logiciel. Pour entrer ce modèle dans le logiciel, on dispose l'objet à la vue des caméras et sur le logiciel, on relie les différents points qui sont relativement fixes pour former la structure élémentaire de l'objet en question.

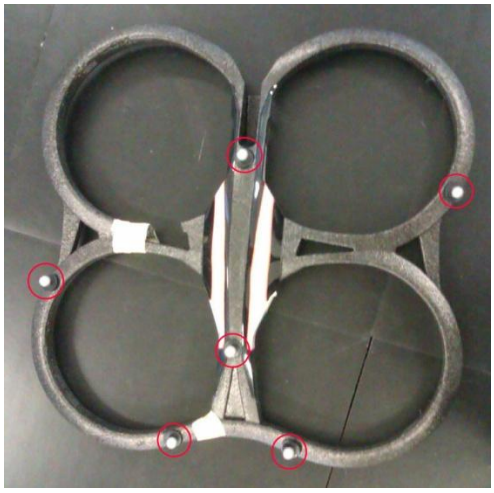


Un exemple d'objet modélisé sous Cortex : un drone Parrot

Une fois l'objet défini et dans le champ de vision des caméras, le logiciel traite les données envoyées par les caméras et les envois sur le réseau, ceci afin que tous les ordinateurs de la salle SAMI puissent y accéder. Ensuite en fonction de l'application, ces données peuvent être utilisées par des programmes écrits en C, en Java ou encore en Matlab.

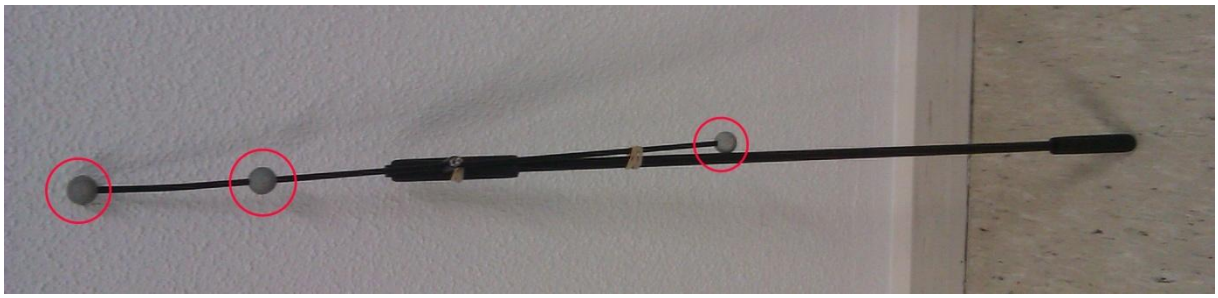
3 - Association des deux

Au cours de ce projet, nous avons disposés des capteurs infrarouges sur la coque du drone Parrot que nous allons utiliser. Ces capteurs ont été posés à des endroits nous permettant d'en déduire facilement les données que nous souhaitons récupérer via le système de capteurs, c'est dire les trois coordonnées du drone dans l'espace, ainsi que les trois angles qui caractérisent le drone. Il faut cependant prendre garde à ne pas disposer les capteurs de façon symétrique sur la coque, sans quoi la distinction des points et donc la connaissance de l'orientation du drone serait impossible.



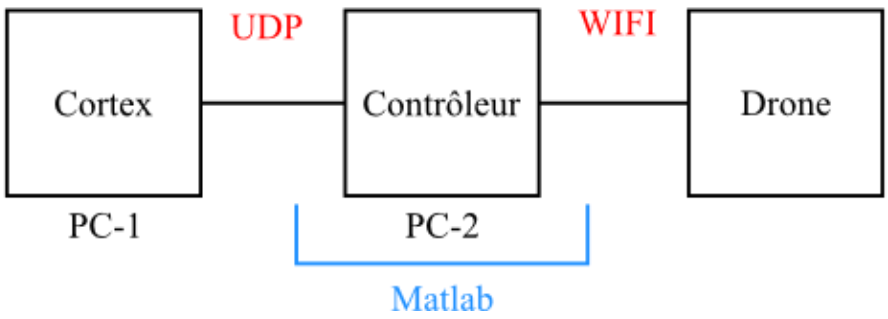
Coque du drone équipée des capteurs.

Nous nous dotons également d'une baguette équipée de capteurs infrarouges. Deux des points de cette baguette nous servirons à pointer un point dans l'espace. Ce point sera l'endroit où le drone sera censé se rendre.



Baguette munie de capteur utilisée pour pointer.

Les positions et orientations du drone et de la baguette sont donc récupérées par cortex via les caméras infrarouges et transmises à un ordinateur via le réseau. Nous avons choisis pour ce projet de développer un programme Matlab ayant pour but de faire aller le drone là où pointe la baguette. Différentes approches du problème seront détaillées par la suite. Le lien entre Matlab et le logiciel Cortex est établit à l'aide de bibliothèques qu'il faut utiliser pour récupérer les informations envoyées par Cortex sur le réseau. On aboutit donc au système global représenté de la manière suivante :



La boucle de retour est effectuée par Cortex qui récupère les informations liées au drone pour les transmettre au contrôleur. Ce contrôleur est donc un ordinateur exécutant un programme Matlab qui récupère les informations de Cortex pour élaborer une commande adaptée, cette commande étant ensuite transmise au drone via Wifi.

II – Approche automaticienne

1 - Plusieurs capteurs pour évaluer l'état

Durant la phase de développement de notre algorithme de contrôle nous avions énormément de difficulté à comprendre pourquoi le drone était parfois instable. Pour évaluer son état le drone utilise deux méthodes principales : des accéléromètres lorsque l'amplitude des mouvements est assez importante ainsi les mesures ne sont pas noyées dans le bruit et une méthode de corrélation d'image grâce à une caméra ventrale lorsque l'amplitude des mouvements est faible.

Dans la salle SAMI le sol est noir et est extrêmement réfléchissant par conséquent l'algorithme chargé de traiter le flux d'image provenant de la caméra ventrale ne peut pas fonctionner correctement. Nous avions des doutes quant à la stabilité du drone et ils se sont confirmés grâce à une expérience simple. Nous avons fait décoller l'appareil puis nous lui avons envoyé des commandes nulles pour qu'il réalise un vol stationnaire, en le soumettant à de petites perturbations il était incapable de corriger sa position. Lorsque le drone arrivait près du bord du terrain de test, le contraste entre le sol noir du plateau et le sol blanc de la salle était beaucoup plus important, dans ces conditions le drone était capable de corriger sa position même lorsque nous le soumettions à des perturbations.

Dans un second temps nous avons réalisé la même expérience en masquant la caméra ventrale à l'aide d'une matière opaque. Le flux vidéo était donc totalement inexploitable. Même lorsque le drone est arrivé près du bord du terrain il n'a pas réussi à se stabiliser quand nous le soumettions à de petites perturbations. Notre hypothèse était donc confirmée, il fallait aider le drone à se stabiliser lorsqu'il n'était soumis à aucune commande pour améliorer sa stabilité et ne pas forcément faire confiance à la boucle de régulation déjà en place.

Nous avons tenté d'aider l'algorithme d'évaluation de l'état obtenu grâce à la caméra ventrale en plaçant des images ayant un fort contraste sur le sol. Malheureusement même avec cette méthode, le drone n'arrive pas toujours à se stabiliser et nous ne pouvons rien y faire.

Nous verrons dans la suite de ce rapport que ce n'est pas forcément un problème car la commande que nous avons mise en place ne nous permet pas de réaliser un vol stationnaire propre. Il est capable de réaliser une poursuite de trajectoire mais il n'est pas capable de rester au-dessus d'un point.

2 - Le système est déjà asservi

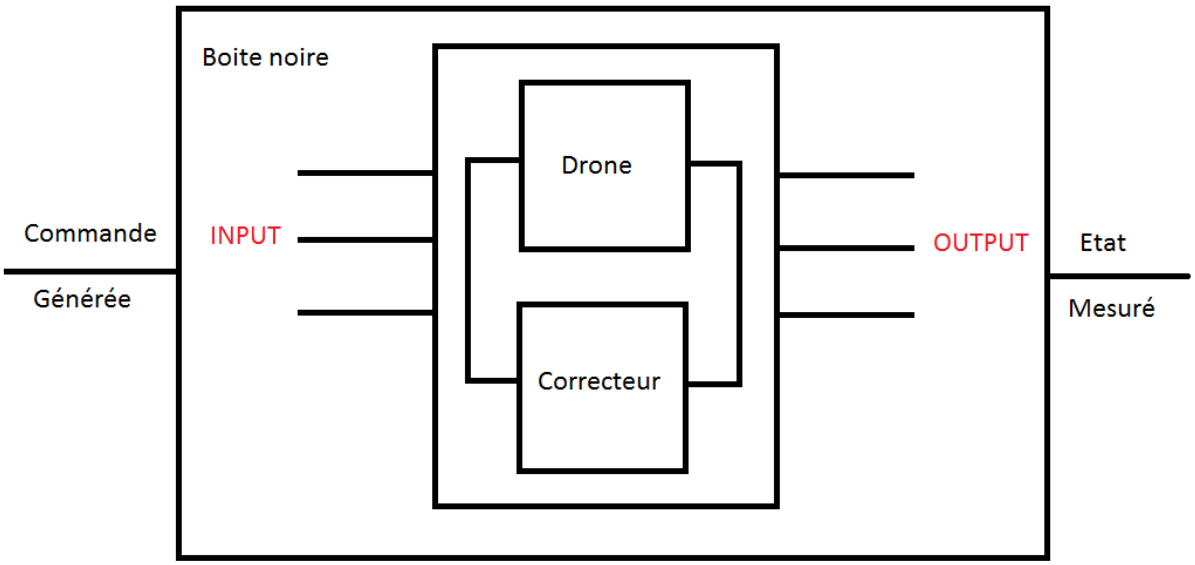
Lorsque nous avons eu l'idée de commander un drone à la baguette nous connaissions déjà le matériel que nous allions utiliser. Nous savions que nous allions utiliser les drones Parrot ainsi que les caméras infrarouges de la salle SAMI.

Cependant nous pensions que nous allions pouvoir utiliser un firmware plus libre du drone, malheureusement, la société française Parrot ne permet pas à ses utilisateurs de toucher directement à la partie matérielle du système. Leur surcouche logicielle n'est pas pratique pour réellement faire de l'automatique, nous ne pouvons pas piloter les variables que nous souhaitons.

Leur drone est surtout destiné aux utilisateurs souhaitant développer des applications à titre vidéo-ludique. Certes il existe des matrices permettant d'inverser les variables et de piloter celles que nous souhaitons. Cependant ces matrices dépendent des paramètres du drone, en particulier sa matrice d'inertie que nous n'avons pas et à laquelle nous ne pouvons pas accéder son identification nous parait également très compliquée.

Cependant si nous étions parvenus à trouver ces matrices d'inversion nous aurions pu piloter tous les degrés de liberté du drone. En effet même si il possède 6 degrés de liberté, nous n'avons besoin que de 4 commandes pour piloter l'intégralité de l'état. Il s'agit du minimum de variable à piloter pour pouvoir avoir une influence sur les 6 degrés de liberté. Bien qu'il soit possible de piloter ces 6 degrés de liberté, comme il y a moins d'entrée que de sorties, les sorties seront fortement corrélées et il ne sera pas possible d'avoir une influence sur une sortie sans avoir un effet souvent néfaste sur une autre.

Au final ce PFE revient à piloter une boite noire ressemblant à :

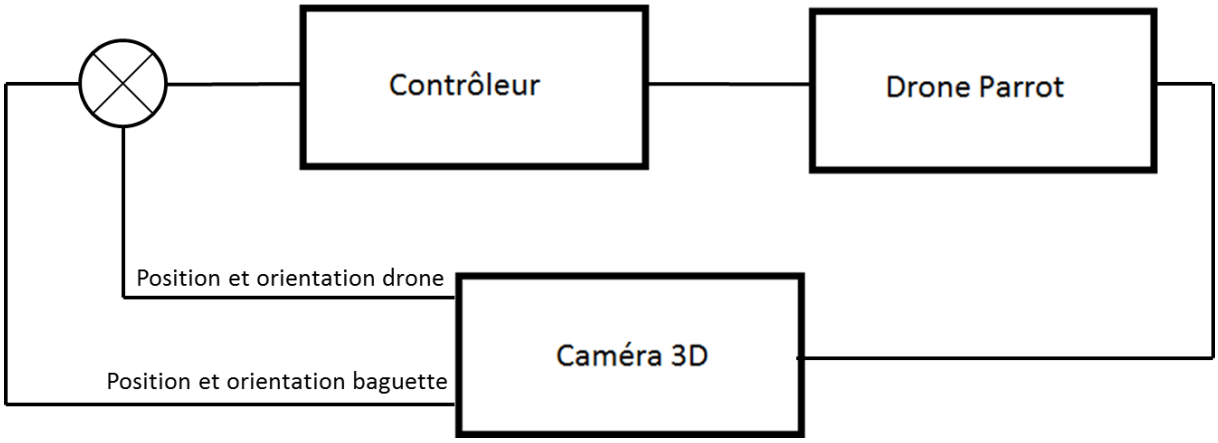


Nous ne connaissons pas du toutes les équations du drone ou bien le type de correcteur qui est utilisé. Une approche simplifiée du modèle peut nous donner un ordre de grandeur sur les commandes à appliquer mais l'estimation de l'état ne sera pas assez précise. Nous allons devoir réaliser une nouvelle boucle pour pouvoir piloter correctement l'appareil mais cette nouvelle boucle peut entrainer des instabilités. Nous ne sommes d'ailleurs pas parvenus à réaliser l'asservissement que nous souhaitions, principalement à cause de l'absence de modèle mathématique et la méconnaissance des méthodes de stabilisation déjà utilisées.

III - Méthode de contrôle

1 - Commande linéaire par retour d'état empirique

Lorsque nous avons commencé la partie asservissement par un retour d'état empirique nous avions le schéma de régulation suivant en tête :



Il est très difficile d'évaluer correctement la matrice des gains que nous devons appliquer aux différents paramètres de l'état pour obtenir un contrôle correct. De plus nous n'avons pas de modèle mathématique nous ne pouvons donc pas étudier le signe de la partie réelle des valeurs propres de la matrice liant les paramètres d'état avec leur dérivée une fois que le retour d'état est établi. Nous avons fait l'hypothèse que le système était assez robuste pour pouvoir accepter n'importe quel type de commande et tout de même rester stable.

Nous sommes parvenus à trouver un quadruplet de gain nous permettant d'avoir une dynamique de vol assez correcte, cependant le drone n'arrivait pas à se stabiliser correctement autour d'un point fixe. Il cherchait en permanence à corriger en recevant des commandes de très petite amplitude ce qui entrainait un phénomène de pompage très gênant. Nous imposions un yaw constant afin de n'avoir que 3 paramètres de vol à piloter, à savoir l'altitude qui doit rester constante, le pitch et le roll.

Nous avons donc décidé d'autoriser le système à avoir une erreur statique non nulle, si le drone se trouvait dans un disque assez proche de la position que nous lui demandions d'atteindre alors il ne recevait plus les commandes ou du moins il n'en tenait plus rigueur. Malgré nos efforts pour obtenir une erreur statique nulle, l'inertie de l'engin l'entrainait toujours hors du disque de stabilité et le phénomène de pompage avait de nouveau lieu car la caméra ventrale n'était pas capable de stabiliser le système.

Nous pensions que l'instabilité était principalement due au fait que nous tentions de piloter trop de variables en même temps. Nous avons donc donné la priorité aux variables qui semblaient les plus importantes. L'altitude est toujours corrigée pour que le drone se trouve à l'altitude de référence, puis le yaw de l'appareil doit toujours rester nul quel que soit la valeur des autres commandes. Pour ce qui est du pitch et du roll, les paramètres qui vont nous permettre de positionner le drone, nous donnons la priorité à la commande dont la norme est la plus grande.

Malheureusement nous ne sommes pas parvenus à changer radicalement le comportement du quadcopter. Il était donc nécessaire de trouver une autre méthode de contrôle.

2 - Commande non linéaire par pondération

Suite aux problèmes que nous avons rencontrés lors de l'implémentation de la commande précédente nous avons décidé de changer radicalement de méthode de contrôle. Désormais la stratégie de commande est la suivante : le drone cherche à s'orienter dans la direction pointée par la baguette tout en avançant vers cette direction en modulant sa vitesse d'avance. Il cherche toujours à corriger son altitude, le problème se ramène donc en théorie à de la poursuite de trajectoire dans le plan.

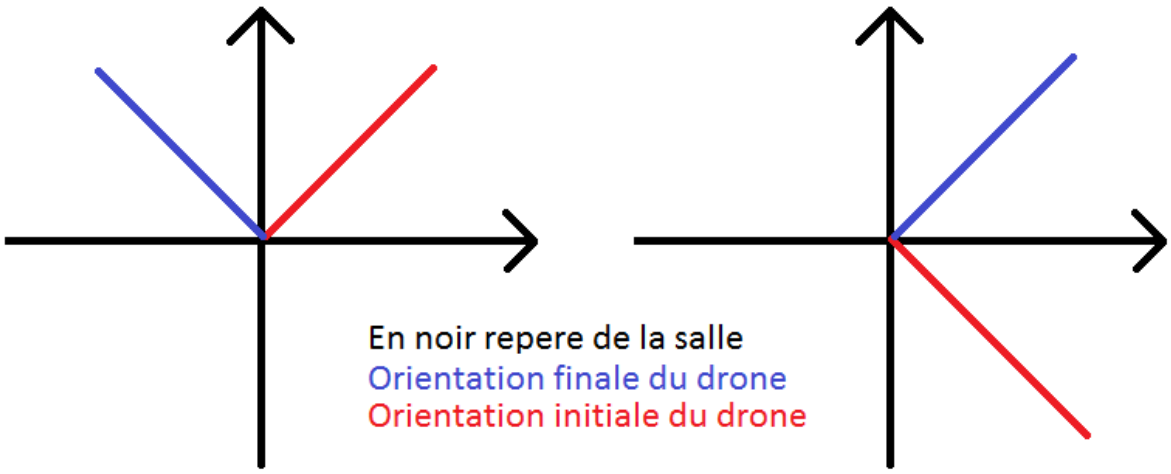
Pour simplifier la synthèse de la commande nous ne pilotons que deux variables : le yaw (rotation autour de *z*) et le pitch (rotation autour de *y*). Cette méthode de contrôle n'est certes pas optimale car nous n'exploitons pas l'intégralité des degrés de liberté que le système nous offre mais elle simplifie grandement le problème tout en donnant des résultats corrects en poursuite.

Etant donné que nous n'avons pas le modèle mathématique nous avons utilisé notre bon sens pour mettre en place cette commande et nous l'avons réglé de façon empirique ce qui a parfois conduit le drone à réaliser quelques cascades inattendues !

a - Commande du yaw

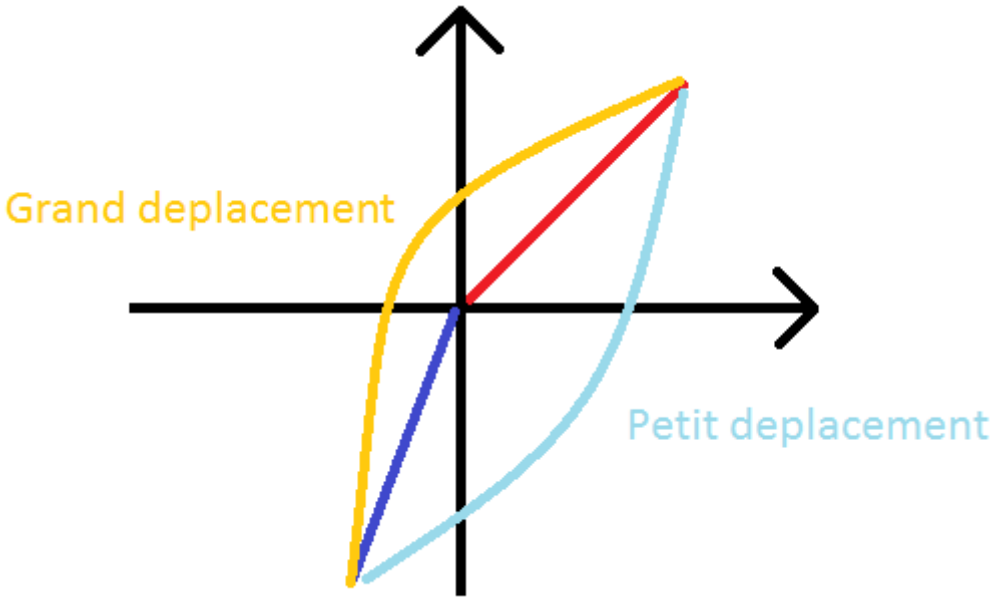
Pour commander le yaw nous avons dû faire un peu de trigonométrie et comme à chaque fois que nous en faisons nous avons des problèmes avec l'orientation des angles et les modulos ! Mais à force de patience et de persévérance nous avons réussi à gérer ces problèmes.

Pour piloter le yaw nous devons faire attention à une chose très importante, nous envoyons des commandes non pas par rapport à une position de référence mais par rapport à la position dans laquelle se trouve le drone. Un schéma sera bien plus clair pour expliquer le problème :



Sur la figure sur dessus nous envoyons la même commande, à savoir nous voulons réaliser une rotation de $\frac{\pi}{2}$ dans le sens trigonométrique. Mais l'état final n'est pas le même, le système est dépendant des conditions initiales, la commande est relative et non pas absolue.

Nous avons donc dû mettre en place un algorithme nous permettant de gérer les différentes commandes de yaw, afin de toujours appliquer la commande qui consommera le moins d'énergie, c'est-à-dire celle o la distance à parcourir est la plus courte. Comme nous pouvons le voir sur la figure suivante pour aller sur une position le système peut emprunter deux chemins qui le conduiront dans le même état final :



Nous choisissons donc logiquement, de toujours réaliser le déplacement le plus court afin d'améliorer la réactivité du système.

b - Commande du pitch

La commande du pitch a elle été beaucoup moins naturelle à mettre en place mais lorsque nous l'étudions de plus prêt elle est tout à fait logique. Nous allons vous la présenter d'une façon assez brutale dans un premier temps et expliquer les termes un par un dans un second temps. Sur le drone commander le pitch maintenant une altitude constante revient à commander sa vitesse de pénétration dans l'air.

$$c_{pitch} = K_{pitch} * e * \cos(c_{yaw})$$

Le terme K_{pitch} nous permet de régler la puissance de la commande, c'est un terme que nous avons réglé par dichotomie afin d'éviter que la commande en pitch soit saturée en permanence ce qui nous permet de construire une commande plus douce.

Le terme $\cos(c_{yaw})$ est très important pour notre application, c'est lui qui va être chargé de gérer une partie de la douceur de la commande. Lorsque le yaw a besoin d'être fortement corrigé, en sachant qu'il prend forcément ses valeurs entre $[-\pi; \pi]$ alors l'importance donnée à la commande

en pitch sera très faible grâce au terme en cosinus. Le système va donc dans un premier temps chercher à s'orienter avant d'avancer vers la position désirée.

Le terme *e* quant à lui représente la norme de l'erreur par rapport à la position de consigne, si le drone est éloigné de la position visée par la baguette alors la commande en pitch sera très forte et le drone répondra vite. Si il est proche de la position voulue alors ce terme sera faible ce qui en théorie devrait éviter au drone de trop dépasser sa position de consigne.

3 - Améliorations possibles

La commande que nous avons développée ne nous permet pas de positionner le drone au-dessus d'un point fixe en vol stationnaire. Il y aura toujours une erreur statique non nulle que le drone cherchera à corriger par le biais de notre commande et cela entrainera forcément des oscillations. Nous avons tenté de pallier à ce problème en demandant au drone de rester en mode stationnaire lorsqu'il était dans un disque autour de la position demandée. Mais dans ce cas-là nous laissons la boucle interne générée l'ensemble des commandes et comme nous l'avons vu à cause de la nature du sol le drone est incapable de se stabiliser correctement lorsqu'il est dans le champ de vision des caméras. Il serait donc intéressant de développer une nouvelle commande qui utiliserait de façon optimale toutes les entrées que nous avons à notre disposition mais nous ne sommes pas parvenus à développer un tel algorithme de contrôle.

Le drone est très sensible à son inertie ce qui le rend très compliqué à stabiliser autour d'un point. En effet il est assez simple de lui demander d'aller sur une position mais à cause de son inertie, même si nous avons du mal pour le faire passer en mode de vol stationnaire. En règle général le drone a le comportement suivant : il passe au-dessus de la position demandée, se rend compte qu'il doit corriger sa position, inverse sa commande et redépasse à nouveau. Ce pompage entraine des oscillations de plus en plus grandes qui peuvent conduire le drone hors du champ des caméras.

Conclusion

En définitive nous avons, malgré certaines difficultés, réussi à contrôler le drone à l'aide d'une baguette même si cela a été fait d'une manière différente de celle escompté au départ. Le drone n'est donc pas stabilisé autour d'une position fixe mais on peut le guider le long de trajectoires à l'aide de la baguette, ce qui correspond bien au but initial. Des améliorations sont donc possibles tant au niveau de la complexité de la commande qu'au niveau de l'approche adoptée. Notre projet pourrait donc servir de bonne base pour un nouveau sujet de projet de fin d'étude à l'ENSEM. Il pourrait être également intéressant et profitable de lancer un projet de construction d'un drone propre à l'école, par l'intermédiaire du club aéronautique. L'ENSEM disposerait ainsi d'un drone dont elle pourrait exploiter toutes les capacités et montrerait ainsi son intérêt pour ce domaine pluri-scientifique qu'est l'aéronautique !

Ce projet que nous pensions au départ à forte dominante automatique, s'est en réalité révélé moins théorique et plutôt orienté informatique et réseau au départ. Il nous a néanmoins permis de mieux appréhender la commande d'un drone, et ce même sans modèle précis. Nous avons également pu utiliser le système de caméra infrarouge, en constatant ses avantages et ses inconvénients. Même si nos différentes approches du problème qui nous a été posé n'ont pas toujours été fructueuses, ce projet nous a poussés à être inventifs, à tester nos idées et à en analyser les résultats. Il nous été donné un aperçu du métier d'ingénieur, qui sera surement encore mis à l'épreuve lors de notre stage de fin d'année.

Comment pouvez-vous utiliser nos codes ?

Pour ne pas voir besoin de gérer plusieurs cartes réseaux et pour simplifier la communication entre les différentes entités que nous utilisons (les caméras infrarouges et le drone) nous devons modifier l'adresse IP du drone et le placer sur le même réseau que le serveur gérant les données issues des caméras.

Lorsque le drone démarre il se met à émettre son propre réseau WiFi. Le système d'exploitation du drone est basé sur un noyau linux, il suffit donc de reconfigurer les paramètres de la carte réseau afin de lui donner une nouvelle adresse IP et de changer le nom du réseau auquel il appartient.

En se connectant via le protocole TELNET sur l'adresse 192.168.1.1 sur le réseau créé par le drone, il faut alors taper la commande suivante dans la console :

```
iwconfig ath0 mode managed essid ENSEM-SAMI; ifconfig ath0 192.168.1.10 up
```

Le nom du réseau créé par le drone deviendra alors le même que celui présent dans la salle SAMI et son adresse IP sera modifiée. Il faut faire attention à ne pas avoir un conflit d'adresse IP sans quoi la communication avec le drone serait impossible.

Nous nous sommes occupés du drone, passons maintenant à la configuration des caméras. Il faut lancer le logiciel Cortex, sélectionner les modèles correspondant au drone et à la caméra puis activer la capture.

Enfin le dossier contenant l'intégralité de nos fichiers Matlab et très simple à utiliser. Les premières lignes de code du fichier *main.m* permettent de configurer les paramètres du drone, nous vous conseillons de ne pas toucher aux paramètres autres que temps et l'altitude de vol sans quoi nous ne pourrons pas garantir le fonctionnement de nos programmes.

Avant de lancer ce programme nous recommandons également aux utilisateurs de lancer une requête de ping sur l'adresse IP du drone, sans cette manipulation le drone ne décollera pas le temps que sa carte WiFi réagisse et sorte du mode veille.

Par défaut, lorsque la baguette n'est pas dans la scène le drone cherchera à aller vers le centre du plateau à l'altitude constante que vous lui avez demandée. Puis dès que la baguette rentrera dans la scène c'est elle qui deviendra prioritaire. Si jamais le drone pour une raison quelconque sort du champ de vision des caméras alors il cherchera à corriger son état sur des données erronées et il deviendra totalement incontrôlable, arrangez-vous pour ne pas que cela arrive.