OLIVIER THÉRIAULT

Intégration d'un système vidéo de poursuite de cible à un simulateur « hardware in the loop » d'avion sans pilote et évaluation d'algorithmes de surveillance

Mémoire présenté
à la Faculté des études supérieures de l'Université Laval
dans le cadre du programme de maîtrise en génie électrique
pour l'obtention du grade de Maître ès sciences (M.Sc.)

DÉPARTEMENT DE GÉNIE ÉLECTRIQUE ET DE GÉNIE INFORMATIQUE
FACULTÉ DES SCIENCES ET DE GÉNIE
UNIVERSITÉ LAVAL
QUÉBEC

2010

Résumé

L'emploi de véhicules aériens sans pilote pour surveiller l'environnement entourant les navires militaires est une avenue intéressante pour contrecarrer des menaces potentielles. Cet ouvrage présente le banc d'essais développé pour l'évaluation et l'analyse comparative d'algorithmes de surveillance de cible. Un système vidéo commercial de poursuite de cible a été intégré à un système « hardware in the loop » (HIL) d'avion sans pilote afin de retrouver la position d'une cible dans un environnement virtuel 3D. La démarche pour l'évaluation de la position de la cible est présentée. Le système HIL utilisé, les modifications matérielles et logicielles apportées ainsi que la performance du système sont décrits. Des algorithmes de surveillance allant des plus simples comme la navigation circulaire aux plus complexes basés sur la commande prédictive sont présentés, simulés sur le système HIL et comparés. Les résultats de cette analyse permettent d'établir les lignes directrices d'une stratégie de guidage efficace.

Avant-propos

En préambule à ce mémoire, je souhaite adresser mes remerciements les plus sincères aux personnes qui m'ont apporté leur aide et qui ont contribué à l'élaboration de ce mémoire ainsi qu'à la réussite de mes travaux de recherche. Je tiens à remercier monsieur Éric Poulin, qui, en tant que mon directeur de recherche, s'est toujours montré à l'écoute et disponible tout au long du projet et sans qui ce mémoire n'aurait jamais vu le jour. Mes remerciements s'adressent aussi aux scientifiques Éric Gagnon et Franklin Wong du RDDC Valcartier qui m'ont permis de me joindre à leur projet tout en offrant un soutien et un encadrement exceptionnels. Je tiens aussi à exprimer ma reconnaissance envers le professeur André Desbiens pour ses contributions toujours appréciées ainsi que pour la lecture et la correction de ce travail. Grâce à ces personnes, j'ai pu parfaire mes connaissances et développer une expertise de travail en recherche qui sera sans aucun doute très utile dans ma carrière. Enfin, j'adresse mes plus sincères remerciements à mes proches et amis, qui m'ont toujours soutenu et encouragé au cours de la réalisation de mon projet de maîtrise.

Table des matières

Table des figures

Liste des tableaux

Nomenclature

α	Angle d'Euler de lacet
β	Angle d'Euler de tangage
γ	Angle d'Euler de roulis
μ	Tolérance de guidage de l'algorithme de Quigley et al. [11]
Π	Plan en 3D
τ	Constante de temps d'un système de premier ordre
σ	Ratio de la vitesse du UAV sur celle de la cible
ξ	Bruit blanc
ψ	Consigne en orientation du UAV
ACR	Algorithme de cercle selon Rafi et al. [12]
AEKF	« Augmented Extended Kalman Filter »
APD	Algorithme de poursuite directe
BHSC	Algorithme de cercle selon Quigley et al. [11]
C	Matrice de rotation pour un changement de coordonnées
CBT	Comportement du bon timonier
CP	Centre de projection d'une caméra
CVL	Algorithme utilisant les champs de vecteur de Lyapunov
D	Vecteur partant du AUV vers la cible
D_s	Distance parallèle d'une période de sinus de l'algorithme à trajectoire sinusoïdale
d	Vecteur représentant l'orientation du UAV
d_p	Distance entre un point et une droite
d_s	Distance parallèle demandée pour l'algorithme à trajectoire sinusoïdale
F	Distance Focale d'une caméra
f	Champ de vue de la caméra
$\mathbf{f}(x_r, y_r)$	Champ de vecteur de guidage
GCS	« Ground Control Station »
GPS	« Global Positioning System »
G_c	Fonction de transfert d'un régulateur
G_o	Fonction de transfert du modèle d'orientation du UAV
G_p	Fonction de transfert du servo panoramique horizontal de la caméra
G_{pp}	Fonction de transfert du servo panoramique horizontal de la caméra avec le HIL
G_{pt}	Fonction de transfert du servo panoramique vertical de la caméra avec le HIL

G_t	Fonction de transfert du servo panoramique vertical de la caméra
G_{cp}	Fonction de transfert du régulateur du servo panoramique horizontal de la caméra
G_{ct}	Fonction de transfert du régulateur du servo panoramique vertical de la caméra
$\boldsymbol{G^d_{uv}}(z)$	Fonction de transfert représentant la dynamique du UAV
$\boldsymbol{G^s_{uv}}(z)$	Fonction de transfert représentant les perturbations non mesurées du UAV
h_c	Horizon de contrôle pour la commande prédictive
h_p	Horizon de prédiction pour la commande prédictive
HIL	« Hardware in the loop »
IP	« Internet Protocol »
$j(k)$	Fonction objectif de la commande prédictive
K	Gain d'une fonction de transfert
N, E	Position Nord et Est du UAV
NED	Référentiel Nord-Est-Sol « North-East-Down »
NTSC	« National Television System Committee »
$\boldsymbol{n_{uv}}(k)$	Position du UAV pour la commande prédictive
$\boldsymbol{n_{tg}}(k)$	Position de la cible pour la commande prédictive
P_x, P_y	Position de la cible en pixel dans l'image
PWM	« Pulse Width Modulation »
R	Matrice de rotation
R_c	Rayon de courbure
R_{ij}	Élément de la rangée i et de la colonne j de la matrice R
Re	Résolultion de la caméra
RPV	« Remotly Piloted Vehicle »
r	Rayon de la trajectoire circulaire autour de la cible
s	Opérateur de Laplace
SDK	« Software Development Kit »
T	Période d'échantillonnage
T_s	Temps d'une période de sinus pour l'algorithme à trajectoire sinusoïdale
t	Temps
TCP	« Transmission Control Protocol »
UAV	« Unmanned Aerial Vehicle »
UDP	« User Datagram Protocol »
USB	« Universal Serial Bus »
$\boldsymbol{u_{uv}}(k)$	Consignes pour l'autopilote pour la commande prédictive
v_0	Vitesse initiale du UAV
v_p	Vitesse du UAV
v_t	Vitesse du UAV
X, Y, Z	Axes orthogonaux
x, y, z	Position selon les axes X, Y, Z
$\dot{x_d}, \dot{y_d}$	Vitesse désirée du UAV dans le plan horizontal

x_p, y_p	Position du UAV dans le plan horizontal
x_t, y_t	Position de la cible dans le plan horizontal
$\dot{x}_t, \dot{x}_t$	Vitesse de la cible dans le plan horizontal
x_r, y_r	Position du UAV par rapport à la cible dans le plan horizontal
x_s, y_s	Position sur la trajectoire sinusoïdale
$\boldsymbol{x_{uv}}(k)$	États du UAV pour la commande prédictive
z	Opérateur discret

Indice inférieur

x	Élément vectoriel selon l'axe X
y	Élément vectoriel selon l'axe Y
z	Élément vectoriel selon l'axe Z
I	Référentiel de l'image
C	Référentiel de la caméra
A	Référentiel de l'avion
M	Référentiel du monde
P	Panoramique horizontal
T	Panoramique vertical
d	Indice de dérivation
f	Indice de filtrage

Chapitre 1

Introduction

Les navires militaires canadiens sont de plus en plus confrontés à des menaces potentielles lorsqu'ils sont à quai, qu'ils sont encrés ou qu'ils sont près de petites embarcations civiles. L'une des méthodes envisagées pour contrecarrer ces menaces potentielles est l'emploi de véhicules aériens sans pilote (UAV) équipés de caméras et de systèmes vidéos de poursuite de cible dans le but de surveiller l'environnement entourant ces navires. Les petits UAVs sont maintenant abordables et pourraient être en mesure de détecter et de déterminer les intentions de petites embarcations à une distance suffisante pour que les navires militaires puissent se protéger. Dans ces scénarios, l'utilisation d'algorithmes de contrôle et de guidage efficaces est importante afin de surveiller les menaces potentielles. Il est possible d'opérer manuellement un UAV et sa caméra, mais les dynamiques incertaines des systèmes, les délais de communication et les perturbations peuvent rendre ces manœuvres difficiles. D'un autre côté, des systèmes autonomes bien conçus nécessiteraient peu d'intervention manuelle, ce qui devrait faciliter la surveillance.

Beaucoup de travaux ont été effectuées concernant les UAVs. De la recherche et sauvetage [1] à la patrouille frontalière [2], les UAVs peuvent être utilisés dans une multitude de scénarios. Pour le sujet qui nous intéresse, la surveillance et la poursuite de cible, il y a beaucoup d'algorithmes qui ont été étudiés. Certains d'entre eux s'intéressent aux équipes de UAVs en coopération [3, 4, 5]. Parmi ceux qui traitent d'un seul UAV, certains utilisent un patron de vagues [6, 7], ce qui semble être très efficace pour les menaces à grande vitesse. Pour les menaces plus lentes, les méthodes les plus simples font tourner le UAV autour de la cible. Quelques méthodes ont été proposées afin de calculer ces trajectoires circulaires. Parmi ces méthodes, on en trouve basées sur les vecteurs de champs de guidage de Lyapunov [8], le comportement du bon timonier [9, 10], les bifurcations de Hopf super-critiques [11] et la stratégie de navigation circulaire

[12]. D'autres algorithmes plus complexes ont aussi été étudiés utilisant les principes d'optimisation. Chen et al. [13] propose une méthode cherchant le chemin optimal à parcourir par le UAV pour surveiller une cible et Prévost et al. [14] utilisent la commande prédictive afin de minimiser la distance séparant le UAV de la cible.

Tandis que plusieurs s'intéressent au contrôle du UAV afin d'effectuer de la surveillance, certains s'intéressent plutôt au contrôle de la caméra. Au lieu de développer des modèles et des lois de contrôle séparément pour le UAV et la caméra, Li et Ding [15] les traitent comme étant un seul objet pour ainsi améliorer les performances de poursuite. D'autres encore s'intéressent plutôt à la localisation de la cible. Barber et al. [16] proposent différentes techniques permettant de diminuer l'erreur de localisation par des méthodes de filtrage, d'estimation de biais, de choix du chemin à parcourir et d'estimation de vent. Gibbins et al. [17] proposent quant à eux une technique augmentant la résolution vidéo afin de mieux détecter les cibles. Certains présentent même une méthode permettant de localiser une cible de façon précise ayant comme seul capteur, en plus de la caméra, un récepteur « Global Positioning System » (GPS) [18]. Whitacre et Wheeler [19] ont quant à eux fait une étude exhaustive des facteurs influençant la localisation de la cible à l'aide d'essais en vol.

Avant de tester les algorithmes de surveillance ou les techniques permettant d'en améliorer la performance lors de vols réels, il importe de s'assurer de leur bon fonctionnement en simulation. Les simulations permettent un développement rapide sans risque d'abimer le matériel. Toutefois, ces simulations s'exécutent habituellement dans des circonstances idéales sans tenir compte des différentes perturbations et latences parfois difficiles à simuler affectant le vol du UAV. L'ajout de matériel aux simulations permet de palier à ce problème en ajoutant de la fiabilité et de la crédibilité aux résultats obtenus toujours sans la complexité, les risques et les coûts associés aux vols réels. Une grande diversité est présente dans le choix du matériel utilisé. Certains utilisent un UAV complet ainsi qu'une soufflerie pour leurs simulations [20], tandis que d'autres utilisent les vrais servomoteurs jumelés à au capteur d'attitude monté sur une plateforme inertielle [21].

Le présent projet à pour objectifs la construction d'un banc d'essais pour la simulation d'algorithmes de surveillance de cible ainsi que l'évaluation et la comparaison de ceux-ci. Pour remplir ces fonctions, un système « hardware in the loop » (HIL) d'avion sans pilote [22] a aussi été utilisé. Ce système HIL est quant à lui monté autour d'un autopilote de la compagnie Micropilot. Ses différents capteurs ont été retirés et il a été relié à un modèle à six degrés de liberté par le biais de cartes d'acquisition. Ce système HIL a déjà montré son bon fonctionnement lors de tests d'algorithmes d'évitement de collisions [23] ainsi que lors de contrôle de vol respectant des spécifications d'arrivée [24].

Afin qu'il soit possible de tester les algorithmes de surveillance dans des circonstances s'approchant de la réalité, un système vidéo de poursuite de cible commercial a été intégré au montage HIL afin de permettre l'évaluation de la position de la cible. Ce mémoire présente les différentes étapes qui ont été nécessaires à cette intégration. Tout d'abord, l'environnement virtuel utilisé par le système HIL a dû être modifié afin de rendre possible le contrôle d'une caméra fournissant les images au système vidéo de poursuite de cible. Ce dernier est utilisé afin de retrouver la position de la cible dans l'image. En plus de servir à retrouver la position de la cible dans le monde 3D pour les algorithmes de surveillance, cette position dans l'image est aussi utilisée à l'intérieur d'une boucle de contrôle servant à garder à tout moment la cible au centre du champ de vue de la caméra.

Ce mémoire démontre aussi le bon fonctionnement du montage HIL par la mise à l'essai d'algorithmes de surveillance variés sur une base commune pour être capable de se prononcer de façon définitive sur leurs forces et faiblesses. Certains s'intègrent plus facilement compte tenu de leur simplicité d'utilisation et de calcul. Un de ceux-ci dirige le UAV directement vers la cible et quatre autres amènent le UAV à effectuer un cercle autour de la cible. Deux autres algorithmes de surveillance plus complexes sont aussi étudiés. Le premier amène le UAV à suivre une trajectoire sinusoïdale à une distance constante derrière la cible. Des tests préliminaires en simulation pure ont toutefois pu permettre d'exposer divers problèmes affectant cet algorithme. Des solutions sont proposées et rendent l'algorithme fonctionnel afin qu'il puisse être testé sur le système HIL. Le second algorithme est très prometteur. Toutefois, la complexité de son implantation dépassant le cadre du présent projet, celui-ci n'a pas été testé sur le montage HIL. L'algorithme profite des avantages de la commande prédictive en estimant la position future de la cible pour en assurer une surveillance efficace. Des simulations n'utilisant pas le système HIL sont tout de même présentées afin d'exposer la performance de l'algorithme.

Le mémoire est organisé comme suit. Premièrement, la recherche de la position de la cible est expliquée ainsi que les mathématiques nécessaires aux calculs. Ensuite, le système HIL et l'intégration du système vidéo de poursuite de cible sont décrits. Cinq algorithmes de guidage de base sont par la suite présentés et simulés sur le système. Ces algorithmes sont analysés et comparés avant d'entreprendre la description et la simulation pure de deux algorithmes de surveillance avancés. Un qui amène le UAV à effectuer une trajectoire sinusoïdale derrière la cible et un autre basé sur la commande prédictive. Après avoir subit quelques correctifs, le premier algorithme est testé sur le système HIL et comparé à un algorithme de base pour en faire ressortir les forces et faiblesses. À partir de cette analyse, les principes d'utilisation stratégique des algorithmes sont exposés. Le mémoire termine en rappelant les éléments importants de son contenu

et en exposant des travaux futurs qui seraient intéressant d'effectuer.

Chapitre 2

Évaluation de la position d'une cible

L'objectif recherché par l'intégration du système vidéo de poursuite de cible est de retrouver la position d'une cible potentielle choisie par l'usager. Cette position peut ensuite être utilisée par des algorithmes de surveillance contrôlant le UAV afin de surveiller cette cible. La position pourra être retrouvée seulement si le système vidéo réussit à ne pas perdre de vue la cible. Dans ce cas, il est possible d'évaluer son emplacement dans le monde à partir de sa position dans l'image de la caméra donnée par le système vidéo. Cette transformation se fait à l'aide de mathématiques peu complexes utilisant la position et de l'orientation du UAV ainsi que de l'orientation de la caméra.

2.1 Les systèmes de coordonnées

La présence d'un système complexe comme un UAV utilisant une caméra implique que différentes pièces ou objets sont déplacés dans l'espace. Il est donc nécessaire de pouvoir représenter leur position et leur orientation. Ces quantités mathématiques sont toujours données par rapport à un système de coordonnées. Par ailleurs, puisqu'il est souvent plus facile de représenter les mouvements d'un objet dans un référentiel plutôt qu'un autre, il est utile de définir plus d'un système de coordonnées. Par exemple, en simulation, il est préférable de définir un système de coordonnées dont l'origine est située à la surface de la terre et de fonctionner en position relative à celle-ci plutôt que de toujours utiliser les coordonnées GPS « Global Positionning System ». Tant que les relations entre les différents systèmes sont connues, il sera toujours possible de transformer les positions et orientations d'un système à un autre. Les différents systèmes de coordonnées présents dans le projet sont décrits dans les sections suivantes.

2.1.1 Le monde

Pour ce projet, le monde a été approximé par un plan tangent en un point à la terre situé l'origine des simulations. Cette supposition est considérée valide puisque les simulations sont limitées dans un petit espace entourant l'origine. Le système de coordonnées NED « North-East-Down » est utilisé où l'axe X est dirigé vers le nord, l'axe Y vers l'est et l'axe Z vers le bas.

2.1.2 L'avion

Pour qu'il soit plus facile de décrire les mouvements de la caméra relativement à l'avion plutôt qu'au monde, il est nécessaire de définir un nouveau système de coordonnées situé sur l'avion. Le système orthogonal a son origine au centre de masse de l'avion et respecte la règle de la main droite ayant le devant de l'avion comme axe X, l'axe Y vers l'aile droite et l'axe Z vers le ventre de l'avion. Il est à noter ici qu'il y aurait pu avoir un autre système d'axe pour le support des servomoteurs panoramiques horizontal « Pan » et vertical « Tilt », mais comme il est considéré aligné avec celui de l'avion et qu'il est très près, il n'est pas répété inutilement.

2.1.3 La caméra

Puisque la caméra ne sera que très rarement alignée avec le UAV, elle doit avoir son propre système de coordonnées. Son origine est la position de la caméra et le système de coordonnées est encore une fois orthogonal. L'axe X est dirigé vers l'axe longitudinal de la caméra, l'axe Y pointe du coté droit et l'axe Z vers le bas.

2.1.4 Le plan image

Un autre système de coordonnées doit être défini pour représenter la position d'un point dans l'image 2D fourni par la caméra. L'origine est le centre de l'image, l'axe X se dirige vers la droite et l'axe Y vers le haut.

2.2 Les angles d'Euler autour d'axes fixes

Pour décrire les orientations angulaires d'un système de coordonnées par rapport à un autre, la méthode la plus utilisée est la définition des trois angles d'Euler : γ, β et α nommés respectivement roulis, tangage et lacet ou en anglais « roll », « pitch » et « yaw ». Le roulis représente la rotation autour de l'axe longitudinal, le tangage autour de l'axe latéral et la lacet autour de l'axe vertical. Par ailleurs, le sens de rotation doit respecter la règle de la main droite comme représenté sur la figure 2.1.

Il est à noter que cette représentation peut amener certains problèmes dû à des singularités dans les matrices de rotation. En effet, il peut arriver qu'il y ait plus d'une orientation possible définie par une seule matrice. Plus de détails sont donnés dans la section 2.3. Toutefois, comme cette représentation est la plus simple et qu'elle n'amène pas de problème dans le cadre de cette étude, aucun travail n'a été effectué pour passer vers une autre représentation. L'utilisation des quaternions [25] aurait pu être envisagée si des problèmes avaient été observés.

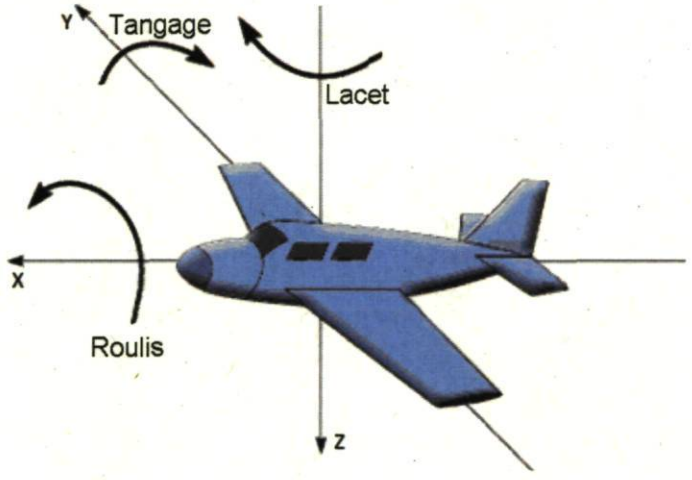


FIGURE 2.1 – Représentation des angles d'Euler

2.3 Les matrices de rotation

Le passage d'un système de coordonnées à un autre se fait par une suite de rotations. Il importe donc de les définir ainsi que le changement de système de coordonnées. Dans cette section et la suivante, la notation utilisée est celle définie par [26].

Tout d'abord, il est important de noter qu'une position est définie par un vecteur [3 x 1] et une matrice de rotation par une matrice [3 x 3].

Les rotations autour des axes individuels sont données par les matrices suivantes :

$$R_x(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix} \tag{2.1}$$

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \tag{2.2}$$

$$R_z(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

Une rotation en 3 dimensions est donc normalement composée d'une combinaison de ces 3 matrices comme suit :

$$R(\gamma,\beta,\alpha) = R_z(\alpha)R_y(\beta)R_x(\gamma) = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix}$$

$$= \begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\cos\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{bmatrix} \tag{2.4}$$

En étudiant cette dernière matrice, il est possible de retrouver les trois angles roulis, tangage et lacet à l'aide des équations suivantes si $\beta \neq \pm\pi/2$.

$$\gamma = \text{atan2}(R_{32}, R_{33}) = \text{atan2}(\sin\gamma, \cos\gamma) \tag{2.5}$$

$$\beta = \text{atan2}(-R_{31}, \sqrt{R_{11}^2 + R_{21}^2}) = \text{atan2}(\sin\beta, \cos\beta) \tag{2.6}$$

$$\alpha = \text{atan2}(R_{21}, R_{11}) = \text{atan2}(\sin\alpha, \cos\alpha) \tag{2.7}$$

Dans le cas où $\beta = \pm\pi/2$, $cos\beta = 0$. Il n'est alors plus possible d'utiliser ces dernières équations pour isoler γ ou α. Il est toutefois possible d'arriver à une soustraction de ces deux angles selon :

$$\begin{aligned} \alpha - \gamma &= \text{atan2}(R_{23}, R_{13}) = \text{atan2}(\sin\alpha\cos\gamma - \cos\alpha\sin\gamma, \cos\alpha\cos\gamma + \sin\alpha\sin\gamma) \\ &= \text{atan2}(sin(\alpha - \gamma), cos(\alpha - \gamma)) \end{aligned} \tag{2.8}$$

γ et α sont indéterminés, mais toujours en se fiant à la convention de Craig [26], il est possible de trouver γ en fixant $\alpha = 0$. Si $\beta = \pi/2$, les équations deviennent :

$$\gamma = \text{atan2}(R_{32}, R_{33}) = \text{atan2}(\sin\gamma, \cos\gamma) \tag{2.9}$$

$$\beta = \text{atan2}(-R_{31}, \sqrt{R_{11}^2 + R_{21}^2}) = \text{atan2}(\sin\beta, \cos\beta) \tag{2.10}$$

$$\alpha = \text{atan2}(R_{21}, R_{11}) = \text{atan2}(\sin\alpha, \cos\alpha) \tag{2.11}$$

Dans ces équations, atan2 est une fonction qui calcule l'arctangente de deux variables y et x de façon similaire au calcul de l'arctangente de y/x, mais le signe des 2 arguments est pris en compte pour déterminer le quadrant du résultat. Contrairement l'arctangente normalement utilisé, l'étendu du résultat est de $]-\pi, \pi]$ au lieu de $]-\pi/2, \pi/2]$.

2.4 Changement de système de coordonnées

Le changement de système de coordonnées du plan image au monde se fait en multipliant tous les changements de coordonnées individuels suivant l'équation 2.12.

$$^{M}_{I}C = {}^{M}_{A}C\,{}^{A}_{C}C\,{}^{C}_{I}C \tag{2.12}$$

où $^{M}_{I}C$ signifie le référentiel I relativement au référentiel M, I est le plan image, C est le référentiel de la caméra, A est le référentiel de l'avion et M est le référentiel du monde.

2.4.1 Passage du référentiel de l'image à celui de la caméra

Afin de mieux comprendre la transformation ${}^{C}_{I}C$, il est important de définir la notion de plan image d'une caméra. La modélisation d'une caméra sera tout d'abord présentée, pour ensuite décrire la transformation.

Modélisation d'une caméra

Tout d'abord, une caméra peut être modélisée simplement à l'aide d'un très petit trou, appelé sténopé, dans la paroi d'une chambre noire qui permet à tous les rayons de lumière qui le traverse de ne frapper le plan image qu'en seulement un point [25]. Son principe est représenté à la figure 2.2 où CP représente le centre de projection et F la distance focale de la caméra. En pratique, pour faire passer plus de lumière dans les caméras réelles, le trou est agrandi et une lentille convergeant les rayons sur le capteur est ajoutée.

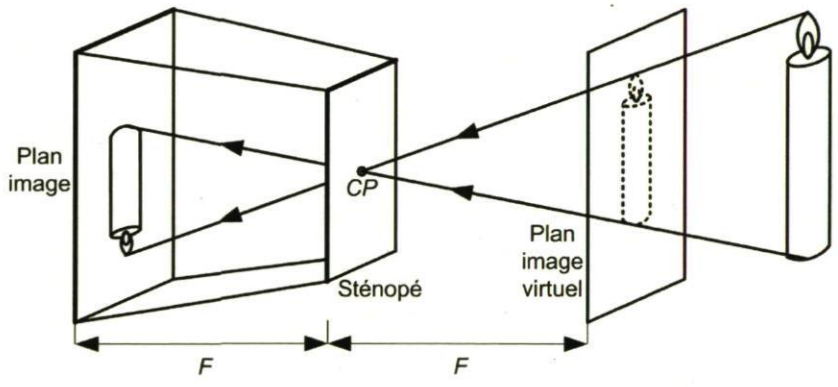


FIGURE 2.2 – Principe du sténopé

Pour simplifier le modèle, le plan image est souvent représenté devant le centre de projection pour éviter les changements de signe dans la modélisation. Ce modèle appelé sténopé non-inverseur est utilisé dans la présente section.

Description de la transformation

Maintenant qu'il est possible de représenter le plan image dans un modèle non-inverseur, il est possible de s'intéresser au passage du référentiel de l'image à celui de la caméra. Il se fait à partir de la position de la cible en pixel (P_x, P_y) par rapport au centre de l'image illustrée à la figure 2.3. Cette figure présente la position de la cible

sur le plan image virtuel dans le référentiel 3D de la caméra. À partir de celle-ci, il est possible de retrouver quels sont les angles d'azimuth et d'élévation α_P et β_P que fait cette position avec l'axe longitudinal de la caméra. L'application de la rotation suivante contenant ces deux angles fait passer du référentiel de l'image à celui de la caméra.

$$ {}^{C}_{I}C = R(0, \beta_P, \alpha_P) \tag{2.13} $$

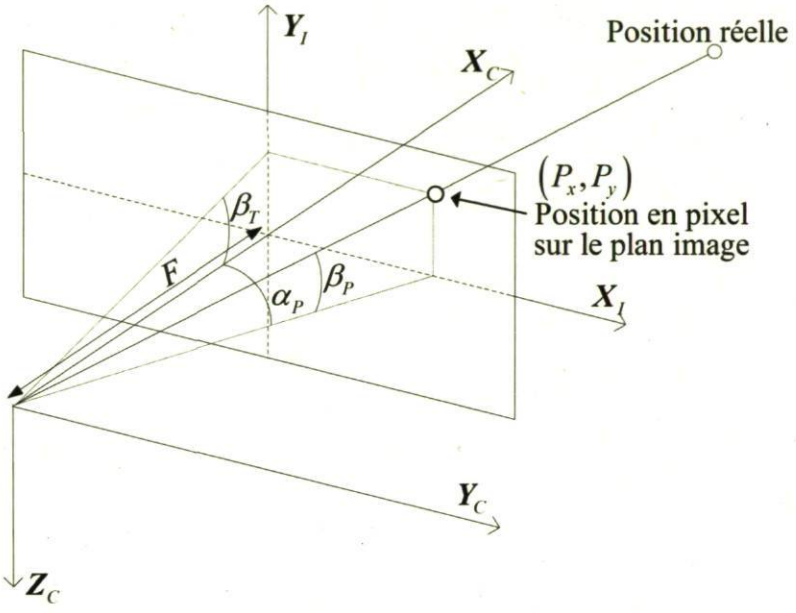


FIGURE 2.3 – Plan image de la caméra

Les angles α_P et β_T se calculent de façon similaire à l'aide de relations trigonométriques montrées à la figure 2.4 où (f_x, f_y) est le champ de vue de la caméra horizontal et vertical et (Re_x, Re_y) est la résolution en pixel de la caméra. L'angle β_T est un angle d'élévation temporaire défini afin de simplifier le calcul de l'angle β_P. Les deux illustrations présentent chacune deux triangles ayant un côté commun. Il est possible de trouver, pour chaque illustration, deux équations permettant de d'obtenir les angles recherchés α_P et β_T en fonction de la résolution, du champ de vue et de la position de la cible en pixel sans tenir compte de la distance focale F. Les deux équations servant à retrouver l'azimut sont donnée par :

$$ \alpha_P = \tan^{-1}\left(\frac{P_x}{F}\right) \tag{2.14} $$

$$ f_x/2 = \tan^{-1}\left(\frac{Re_x/2}{F}\right) \tag{2.15} $$

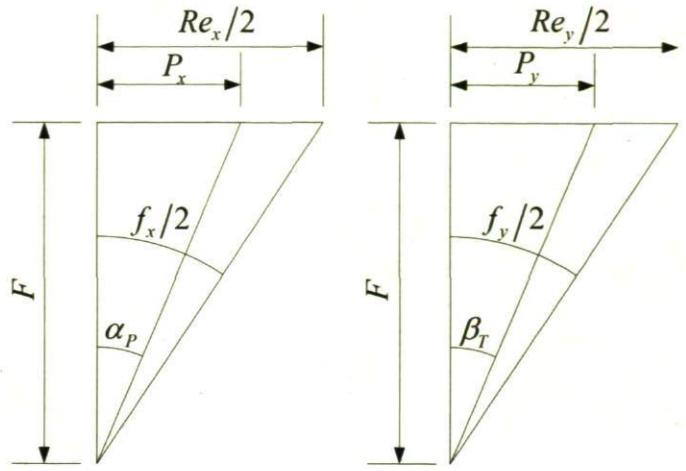


FIGURE 2.4 – Angles dans le repère de la caméra à partir de la position en pixel

En isolant F dans l'équation 2.15 et en le remplaçant dans l'équation 2.14, on retrouve :

$$\alpha_P = \tan^{-1}\left(\frac{2P_x \tan(f_x/2)}{Re_x}\right) \quad . \tag{2.16}$$

Suivant la même démarche, l'angle d'élévation temporaire est donné par :

$$\beta_T = \tan^{-1}\left(\frac{2P_y \tan(f_y/2)}{Re_y}\right) \tag{2.17}$$

La figure 2.5 présente de façon similaire l'angle d'élévation β_P et permet de trouver la relation :

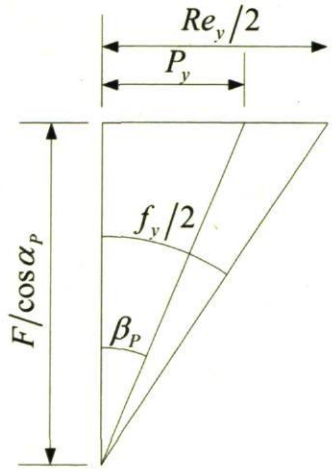


FIGURE 2.5 – Angle d'élévation de la position de la cible en pixel dans le repère de la caméra

$$\beta_P = \tan^{-1}\left(\frac{P_y \cos(\alpha_P)}{F}\right) \tag{2.18}$$

À partir de l'équation 2.18 et sachant que :

$$\tan(\beta_T) = \frac{P_y}{F} \tag{2.19}$$

il est possible d'exprimer la relation entre les angles β_P et β_T par :

$$\beta_P = \tan^{-1}\left(\cos(\alpha_P)\tan(\beta_T)\right) \tag{2.20}$$

Finalement, à l'aide de l'équation 2.17, l'équation 2.20 peut être manipulée pour devenir :

$$\beta_P = \tan^{-1}\left(\frac{2P_y \tan(f_y/2)\cos(\alpha)}{Re_y}\right) \tag{2.21}$$

2.4.2 Passage du référentiel de la caméra à celui de l'avion

Pour passer du référentiel de la caméra à celui de l'avion, il suffit d'appliquer la rotation :

$$^{A}_{C}C = R(0, \beta_C, \alpha_C) \tag{2.22}$$

où les angles β_C et α_C, sont respectivement les angles panoramiques vertical et horizontal de la caméra. Puisque celle-ci est montée sur un système à 2 axes de rotation, il n'y a pas de roulis appliqué à la caméra par son support.

2.4.3 Passage du référentiel de l'avion à celui du monde

Le passage du référentiel de l'avion à celui du monde se fait aussi simplement en appliquant la matrice de rotation suivante :

$$^{M}_{A}C = R(\gamma_A, \beta_A, \alpha_A) \tag{2.23}$$

où γ_A représente avec le roulis, β_A le tangage et α_A le lacet de l'avion.

2.5 Calcul de la position de la cible

Les sections précédentes ont définies les relations servant à faire passer la position en pixel de la cible dans l'image de la caméra vers une orientation définie par trois angles dans le monde. La figure 2.6 présente la droite créée par cette orientation sous forme de vecteur (a, b, c) et la position (x_1, y_1, z_1) du UAV. Pour trouver la position de la cible (x, y, z), il suffit ensuite de trouver en quel point cette droite traverse le plan $\Pi : z = 0$. De plus, puisque les simulations se déroulent dans un environnement maritime, l'altitude connue de 0 mètre ne permet qu'une solution valide. Il n'est donc pas nécessaire de faire de la triangulation comme Dobrokhodov et al. [27].

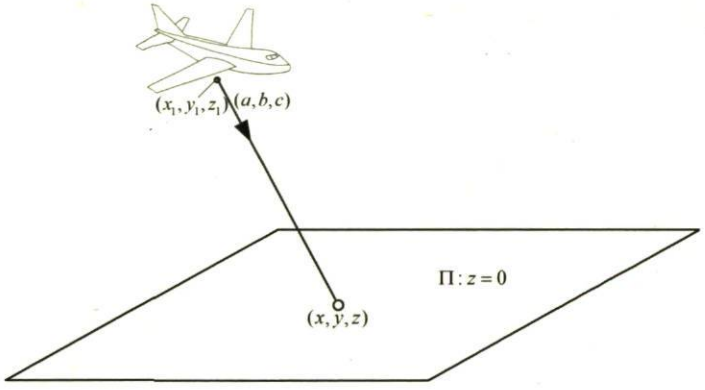


FIGURE 2.6 – Recherche de la position de la cible

L'équation sous forme vectorielle d'une droite dans l'espace est donnée par :

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} + \begin{bmatrix} a \\ b \\ c \end{bmatrix} r \tag{2.24}$$

où le vecteur parallèle à la droite (a, b, c) se retrouve à partir de la position en pixel dans l'image, de l'orientation du modèle de la caméra ainsi que de l'orientation du UAV. À partir de ces informations, il est possible de passer du référentiel de l'image à celui du monde par l'équation 2.12 décrite à la section 2.4. Il faut par la suite retrouver les angles de lacet et de tangage α et β de la matrice de rotation résultante à l'aide des relations 2.6 et 2.7. Il est à noter que le roulis n'est pas nécessaire puisqu'un vecteur en 3 dimensions peut être représenté seulement par 2 angles. Une dernière étape est ensuite nécessaire pour retrouver quelles sont les coordonnées cartésiennes de ce vecteur décrit à l'aide d'angles. Les trois équations suivantes donnent finalement les coordonnées du vecteur (a, b, c).

$$a = \cos(\alpha)cos(\beta); \tag{2.25}$$

$$b = \sin(\alpha)cos(\beta); \tag{2.26}$$

$$c = -\sin(\beta) \tag{2.27}$$

Toutes les informations sont maintenant disponibles afin de solutionner le problème. Comme l'équation du plan Π nous donne $z = 0$, on peut isoler r et le remplacer pour retrouver la position x et y de la cible au sol :

$$z = 0 = z_1 + cr \tag{2.28}$$

$$r = \frac{-z_1}{c} \tag{2.29}$$

$$x = x_1 - \frac{az_1}{c} \tag{2.30}$$

$$y = y_1 - \frac{bz_1}{c} \tag{2.31}$$

2.6 Conclusion

Ce chapitre a présenté les différentes notions mathématiques nécessaires à l'évaluation de la position d'une cible. Dans un premier temps, les différents systèmes de coordonnées présents ont été exposés. Ensuite, les angles d'Euler et les matrices de rotation ont été définis afin d'introduire les changements de systèmes de coordonnées. Ceux-ci, qui sont normalement composés d'une suite de rotations, sont au cœur du calcul de la position de la cible. Ils permettent de trouver un vecteur parallèle à la droite reliant le UAV à la cible. La résolution de l'équation de cette droite pour une altitude de 0 mètre permet enfin de trouver les deux coordonnées x et y inconnues de la cible.

Chapitre 3

Système HIL

Ce chapitre présente le système HIL utilisé ainsi que les modifications apportées afin qu'il soit possible d'évaluer et comparer des algorithmes de surveillance de cible à l'aide de simulations s'approchant de la réalité. Dans un premier temps, le système HIL utilisé est décrit avant d'entreprendre la présentation des modifications matérielles apportées nécessaires à l'intégration du système vidéo de poursuite de cible. Cette intégration ayant nécessité plusieurs changements au niveau du modèle Simulink simulant le vol d'un UAV, ils sont ensuite décrits dans une section distincte. Par la suite, l'utilisation du simulateur de vol X-Plane pour fournir le champ de vue de la caméra ainsi qu'une vue 3D de la simulation est décrite. Suit une autre section présentant les modifications effectuées au logiciel SerialBridge afin d'assurer la gestion et le contrôle des simulations. Enfin, une section résume à l'aide d'une figure toutes les communications intervenant entre les différents éléments du système HIL.

3.1 Système HIL initial

Le système HIL a été conçu pour qu'il soit possible de faire des simulations de vol avec un UAV en laboratoire. Il utilise un vrai autopilote dont tous les capteurs sont simulés à partir d'un ordinateur exécutant un modèle de UAV. Il est donc possible de simuler de façon réaliste des vols et de tester des algorithmes de surveillance avancés en laboratoire sans toutefois subir la complexité et les risques associés aux vols extérieurs. Le système HIL est présenté à la figure 3.1.

Le montage est composé d'un ordinateur simulant un modèle haute fidélité à six

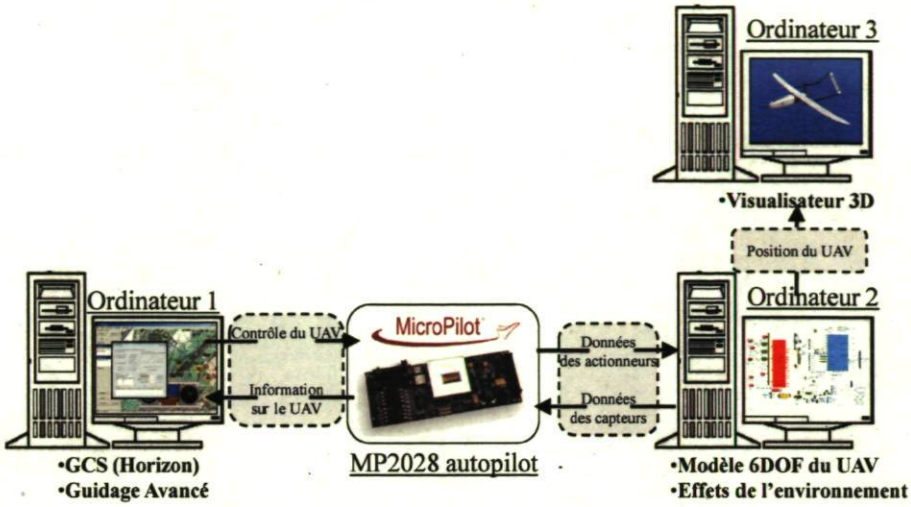


FIGURE 3.1 – Banc d'essais HIL

degrés de liberté (ordinateur 2) qui est connecté à un autopilote commercial. L'ordinateur joue le rôle du UAV et l'autopilote contrôle le modèle au lieu d'un vrai UAV. Le modèle est implanté dans Simulink et exécuté sous xPC Target. xPC Target est un système d'exploitation développé par Mathworks qui permet d'interconnecter des modèles Simulink avec des systèmes physiques dans le but d'obtenir des exécutions en temps réel.

Le modèle, qui provient du « blockset » AeroSim de la compagnie Unmanned Dynamics, a été paramétrisé pour reproduire le comportement du UAV Aerosonde représenté à la figure 3.2. Celui-ci est très connu puisque c'est le premier à avoir traversé l'Atlantique Nord.

FIGURE 3.2 – UAV Aerosonde

L'autopilote utilisé est le MP2028g [28] de la compagnie canadienne Micropilot. Il

est de très faible taille et pèse seulement 28 grammes. Il est composé d'accéléromètres et de gyroscopes sur trois axes, d'un GPS, d'un altimètre ainsi que d'un capteur de vitesse. Il permet de faire des vols complètement autonomes en suivant une liste de positions GPS tout en gardant une vitesse ainsi qu'une altitude désirées.

Les signaux de commande envoyés par l'autopilote aux actionneurs du UAV, à savoir les gouvernails de direction et de profondeur, l'élévateur et la manette des gaz, sont lus par l'ordinateur 2 et convertis en angle et en fraction de manette des gaz avant d'alimenter le modèle. Ce dernier génère ensuite les signaux des différents capteurs et les retourne à l'autopilote. L'ordinateur 1 est utilisé pour exécuter le « Ground Control Station » (GCS) Horizon fourni par Micropilot ainsi que le programme maison SerialBridge qui est utilisé pour tester des algorithmes de guidage avancés. Le principe de fonctionnement de SerialBridge est qu'il intercepte toutes les communications entre le GCS et l'autopilote. Il a donc accès à toutes les informations sur le UAV et il peut aussi envoyer d'autres commandes vers l'autopilote pour le contrôler. Par ailleurs, comme le GCS ne permet qu'une vue en deux dimensions des vols, un troisième ordinateur roulant un visualisateur en trois dimensions (X-Plane [29]) a été intégré pour avoir un meilleur aperçu de l'attitude du UAV (ordinateur 3).

Tableau 3.1 – Capteurs et variables utilisés de l'autopilote

Capteurs	Variables
Trois accéléromètres	Accélération selon les axes X, Y et Z de l'avion
Trois gyros	Angles lacet, tangage et roulis
Capteur de pression différentiel	Vitesse selon l'axe X de l'avion
Capteur de pression statique	Altitude
GPS	Position géodésique et vitesse par rapport au sol

La figure 3.3 montre que l'ordinateur 2 est connecté à l'autopilote par l'entremise de cartes d'acquisition. Pour ce faire, les différents capteurs de l'autopilote ont été retirés et les servomoteurs déconnectés. Les différents capteurs sont listés dans le tableau 3.1. Les signaux des accéléromètres sont envoyés à l'autopilote en signaux « Pulse Width Modulation » (PWM) par la carte de compteurs NI6602 de National Instruments [30]. Cette même carte s'occupe aussi de lire les signaux PWM que l'autopilote envoie à ses servomoteurs. Pour ce qui est de la vitesse, de l'altitude ainsi que des vitesses angulaires, c'est la carte de conversion numérique/analogique NI6702 qui est utilisée afin de reproduire ces signaux pour l'autopilote. Les informations GPS provenant du modèle sont, quant à elles, envoyées par port série à l'autopilote.

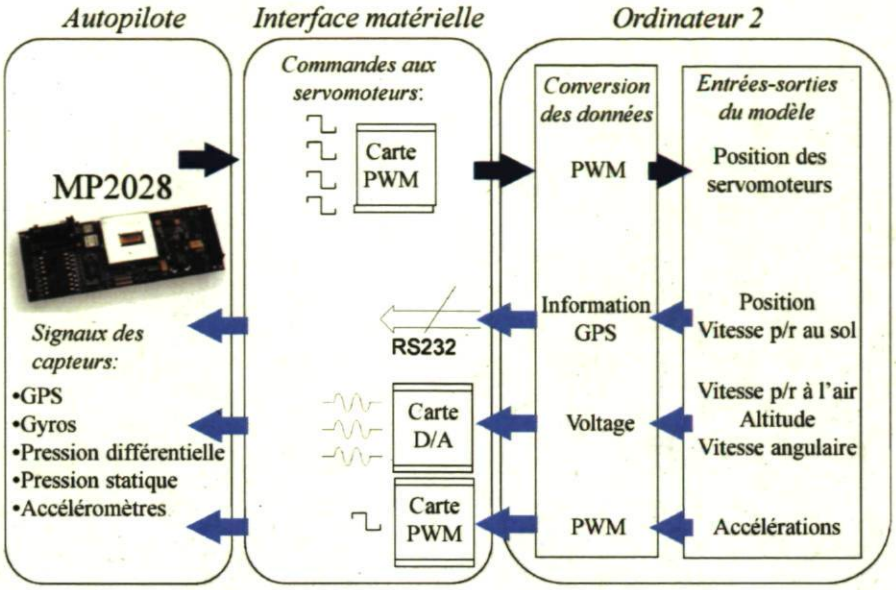


FIGURE 3.3 – Interface entre l'autopilote et le modèle

3.2 Modifications matérielles apportées au système HIL

L'intégration du système vidéo de poursuite de cible ayant pour objectif la recherche et le développement de stratégies de guidage pour la surveillance maritime a nécessité l'ajout de 2 ordinateurs : le système vidéo ainsi qu'un ordinateur simulant une caméra. Une manette à aussi dû être ajoutée afin qu'il soit possible de contrôler les déplacements de la caméra virtuelle. La nouvelle architecture est présentée à la figure 3.4.

Tout d'abord, le système vidéo de poursuite de cible est lui même un ordinateur embarqué ayant Linux comme système d'exploitation. C'est le PVU-Mariner de la compagnie PerceptiVU [31] qu'on peut voir à la figure 3.5. Il est doté de différents algorithmes permettant de poursuivre des véhicules terrestres, des avions, des bateaux et mêmes des personnes. L'unité de traitement principale est un Céléron d'Intel cadencé à 1.0 GHz accompagné de 256 MB de mémoire RAM DDR. Il est aussi doté d'un port Ethernet, de 6 ports USB ainsi que de 4 ports séries. L'ordinateur est normalement couplé à un contrôleur d'unité Pan&Tilt PTU-D46 de la compagnie Directed Perception [32]. L'ordinateur communique avec le contrôleur à l'aide d'un de ses ports séries. Une caméra de la compagnie COHU Inc, modèle 3930, peut être montée sur l'unité Pan&Tilt à l'ordinateur de poursuite de cible. Elle opère dans le domaine visible à l'aide d'un zoom optique 23X et elle est montée dans un habitacle scellé et pressurisé. Elle est contrôlée

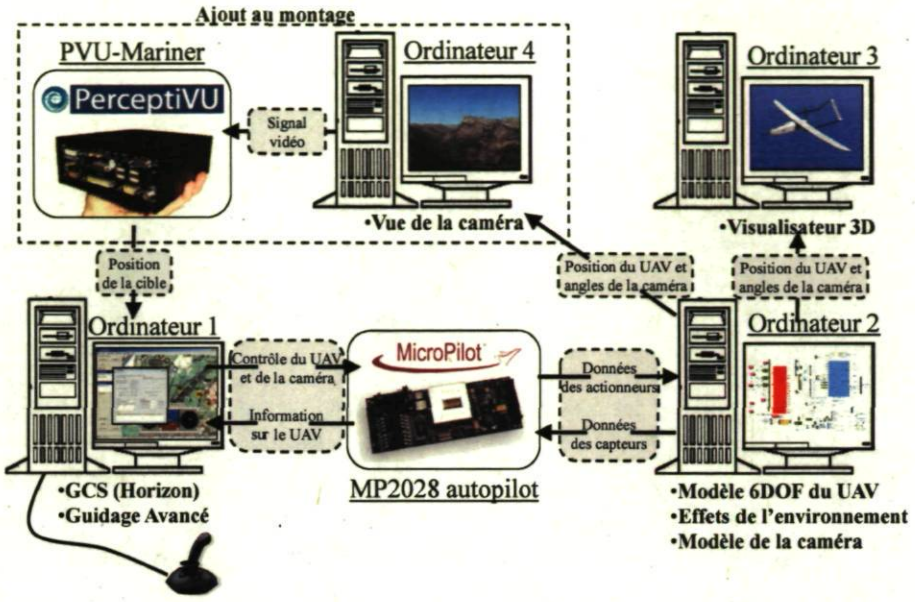


FIGURE 3.4 – Nouveau diagramme du banc d'essais HIL

à partir d'un lien série.

L'ensemble du système est montré à la figure 3.5 et peut se contrôler à partir du clavier de l'ordinateur ou à l'aide d'une manette. On peut donc démarrer ou arrêter la poursuite, déplacer la caméra, ajuster son zoom ainsi que choisir les différentes options des algorithmes de poursuite.

Le second ordinateur qui est ajouté (Ordinateur 4) est celui qui est utilisé pour générer le signal vidéo de la caméra virtuelle. La carte vidéo de cet ordinateur doit avoir une sortie compatible à la norme NTSC pour pouvoir fournir un signal vidéo composite à l'entrée de la carte d'acquisition du système. Il faut donc utiliser un convertisseur s-video à RCA si la carte vidéo n'a que ce type de sortie. Le logiciel 3D utilisé pour fournir ce signal est le même qui était déjà utilisé pour montrer la scène en 3D : X-Plane. Toutefois, ce logiciel ne permettait pas de contrôler des caméras. Un module d'extension a donc été développé pour remplir cette fonction. Les détails de cette implantation se trouvent à la section 3.4.

La manette fournie avec le système vidéo de poursuite de cible permet d'interagir avec le contrôleur des servomoteurs de la caméra. Dans notre cas, comme la caméra est virtuelle et qu'il n'est pas possible d'intercepter les commandes envoyées au système par sa manette, un autre moyen devait être trouvé pour la faire déplacer. L'ajout d'une autre manette reliée à l'ordinateur exécutant le programme maison SerialBridge qui s'occupe de la gestion des simulations s'est avéré une solution pratique et très efficace.

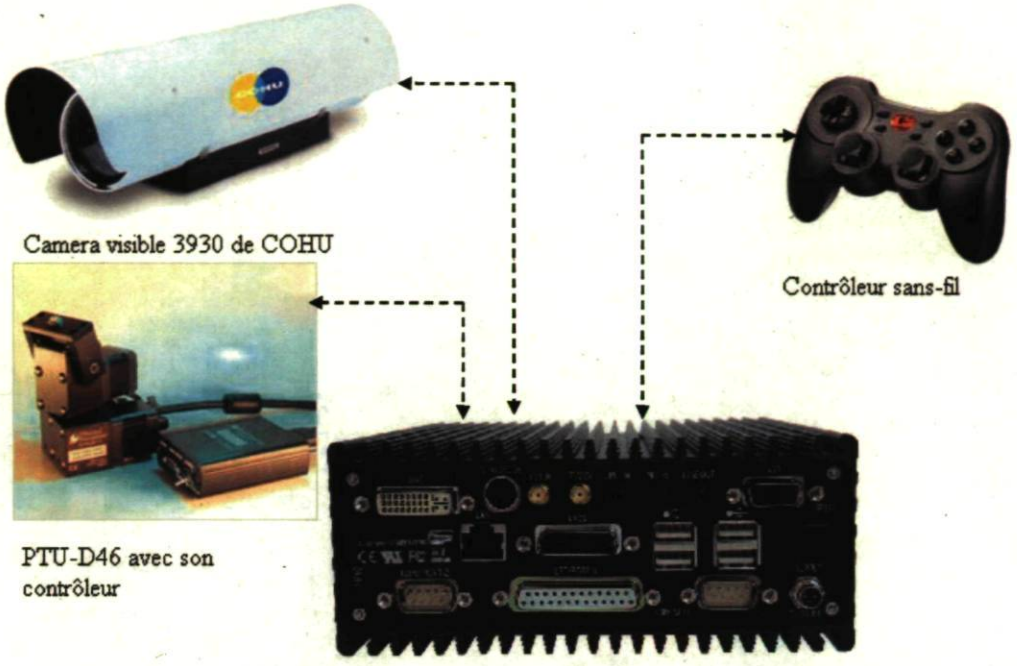


FIGURE 3.5 – Le système vidéo de poursuite de cible PVU-Mariner de la compagnie PerceptiVU

En effet, l'ajout de cette manette a permis de remplacer celle fournie avec le système vidéo en offrant de nouvelles possibilités à l'usager. Celle qui a été utilisée est le modèle SpaceExplorer de la compagnie 3DConnexion [33] et offre 6 degrés de liberté (figure 3.6). Son intégration est détaillée à la section 3.5.2.

FIGURE 3.6 – Manette utilisée pour contrôler le système

3.3 Modifications apportées au modèle Simulink

Pour que les simulations respectent le plus possible la réalité, la dynamique de la caméra réelle a été implantée dans le modèle Simulink du système HIL. La caméra

est représentée à la figure 3.7. Les servomoteurs 5 et 6 de l'autopilote ont été choisis respectivement pour contrôler le panoramique horizontal et vertical. Afin de lire les commandes envoyées à ces nouveaux servomoteurs par l'autopilote, une nouvelle carte de compteurs NI6602 de la compagnie National Instruments a été utilisée.

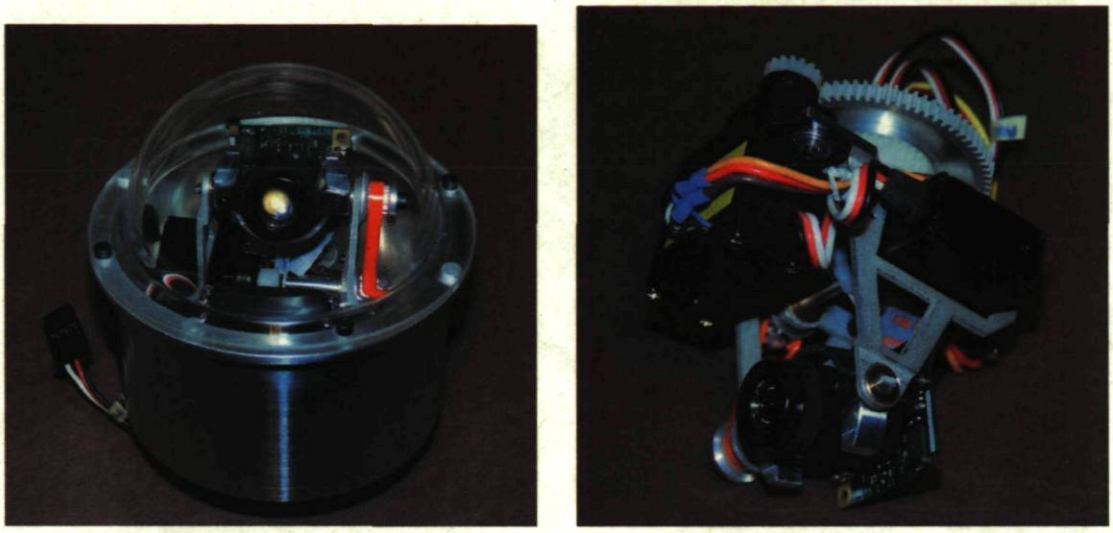


FIGURE 3.7 – Caméra utilisée lors d'essais en vol

3.3.1 Modélisation des servomoteurs panoramiques horizontal et vertical

Lors d'un changement de consigne envoyé à l'autopilote pour le déplacement de la caméra, on peut voir le changement s'effectuer instantanément aux bornes de l'autopilote normalement connectées aux servomoteurs. Toutefois, puisque ceux-ci ont été retirés et remplacés par une carte d'acquisition, il n'est plus possible de voir leur dynamique réelle. Pour palier à ce problème, une identification des servomoteurs a été faite afin de pouvoir les modéliser et ainsi retrouver un déplacement de caméra représentant la réalité.

Une méthode simple pour l'identification d'un procédé est l'analyse de la réponse à l'échelon. Malgré qu'elle soit approximative, cette méthode donne toutefois de très bons résultats. Le signal qui doit être envoyé aux servomoteurs doit être un signal PWM à 50Hz donnant une pulsation qui varie d'une largeur de 1 à 2 ms. 1 ms amène donc le servomoteur à sa position minimale et 2 ms à sa position maximale. Un montage utilisant un contrôleur PWM, le TL494, et différents amplificateurs opérationnels a été réalisé pour effectuer cette tâche [34]. Il reçoit en entrée une tension variant entre 0 et 5V et produit à la sortie le signal pour les servomoteurs.

Pour ce qui est de la lecture de la position en sortie, un capteur de position angulaire a été nécessaire. Comme les deux servomoteurs à identifier sont déjà asservis en position à l'aide d'un potentiomètre, le gain des deux procédés est connu et unitaire. Dans le cas du panoramique horizontal, il a été possible de se connecter directement à ses bornes. Malheureusement, pour ce qui est du panoramique vertical, un autre potentiomètre a dû être utilisé pour jouer le rôle du capteur de position puisque le potentiomètre n'était pas accessible.

L'ensemble du montage se résume à la figure 3.8. Un ordinateur utilisant le logiciel LabView a été utilisé pour contrôler la carte de génération de données analogiques (BNC-2110) de la compagnie Nationnal Instruments ainsi que les cartes d'acquisition (SCXI1102C et PXI6052E) de la même compagnie.

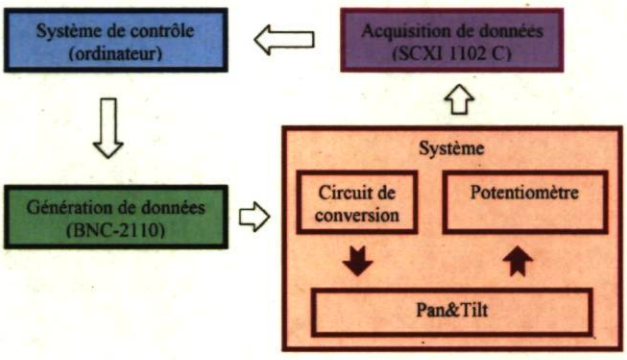


FIGURE 3.8 – Montage utilisé pour faire l'identification des servomoteurs

Afin de s'assurer de la linéarité des servomoteurs à identifier, différents échelons de consigne ont été envoyés aux 2 servomoteurs et une dynamique de premier ordre à été approximée pour chacun d'eux.

Servomoteur panoramique horizontal

Pour de petits échelons de consigne, en deçà de 100° sur une possibilité de près de 360°, aucun délai n'a été noté et une constante de temps de 0.78 seconde a été approximée. Pour de plus grands échelons, une non-linéarité se produit amenant une limitation sur la vitesse de déplacement du servomoteur. Il y a donc un limiteur de pente à ajouter au modèle. La figure 3.9 nous montre bien l'effet de cette limite sur la pente pour un échelon de consigne de 144°. Différents échelons de grandes amplitudes ont permis de découvrir que cette pente limite se trouve à ±85°/s. La fonction de transfert G_P entre la consigne envoyée au servomoteur et l'angle panoramique horizontal de la

caméra est représenté par l'équation suivante :

$$G_P(s) = \frac{1}{1 + 0.78s} \tag{3.1}$$

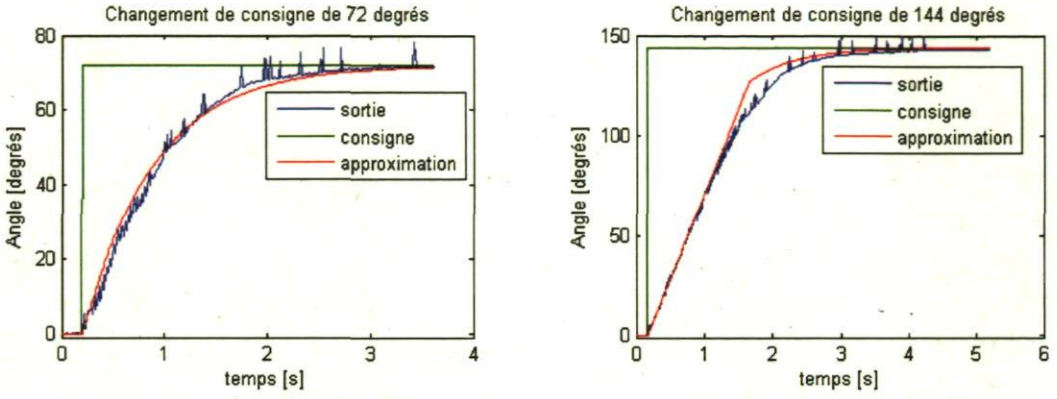


FIGURE 3.9 – Réponse du servomoteur panoramique horizontal suite à des échelons de consigne de 72 et 144 degrés

Servomoteur panoramique vertical

Le servomoteur panoramique vertical est de son côté beaucoup plus rapide. De petits échelons de consigne ont permis de déterminer la constante de temps de 0.033 seconde. Toutefois, malgré son petit temps de réponse, il y a un retard non négligeable de 0.05 seconde. De plus grand échelons de consigne ont aussi permis de déterminer la pente limite de ce servomoteur à $\pm 580°/s$. La fonction de transfert résultante G_T entre la consigne envoyée au servomoteur et l'angle panoramique vertical de la caméra est donnée :

$$G_T(s) = \frac{e^{-0.05s}}{1 + 0.033s} \tag{3.2}$$

La figure 3.10 montre que cette approximation est très près de la réalité.

3.3.2 Intégration de la caméra au modèle Simulink

Maintenant que les deux fonctions de transfert sont connues, il est possible de les intégrer au modèle Simulink couplé à l'autopilote. Les deux fonctions de transfert sont

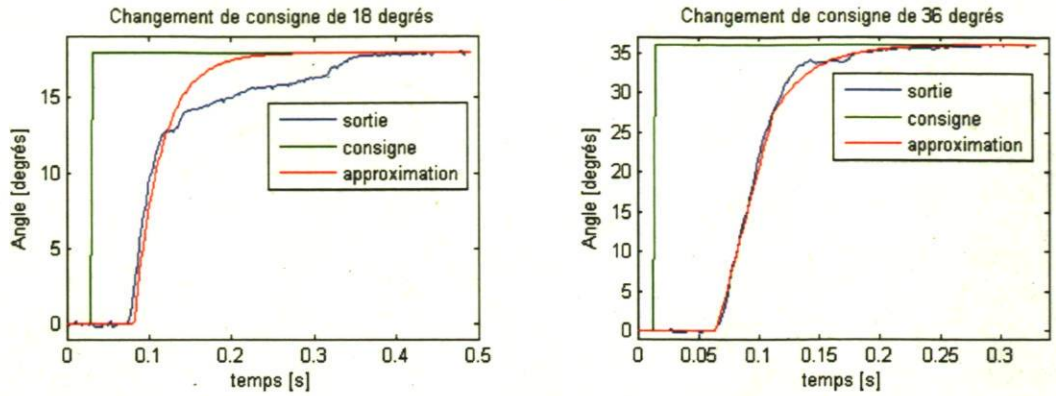


FIGURE 3.10 – Réponse du servomoteur panoramique vertical suite à différents échelons de consigne

ajoutées directement après la lecture des servomoteurs en PWM par la carte d'acquisition et la conversion de ce signal en angle. Les figures 3.11 et 3.12 montrent les différents blocs qui ont été ajoutés.

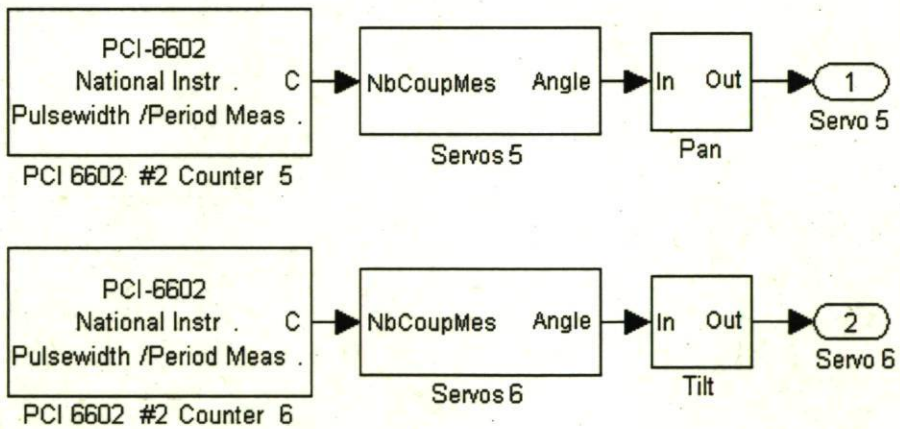


FIGURE 3.11 – Blocs ajoutés pour la lecture des nouveaux servomoteurs

3.3.3 Transfert des données vers X-Plane

Puisque les sorties des servomoteurs panoramiques horizontal et vertical représentent maintenant les mouvements de la caméra réelle, elles peuvent être envoyées directement à la caméra virtuelle de X-Plane. Un bloc *S-Function* de Simulink a été créé afin de construire un paquet UDP contenant la position et l'attitude du UAV ainsi que les deux angles de la caméra. Ce paquet est ensuite envoyé à une fréquence de 50Hz vers l'ordi-

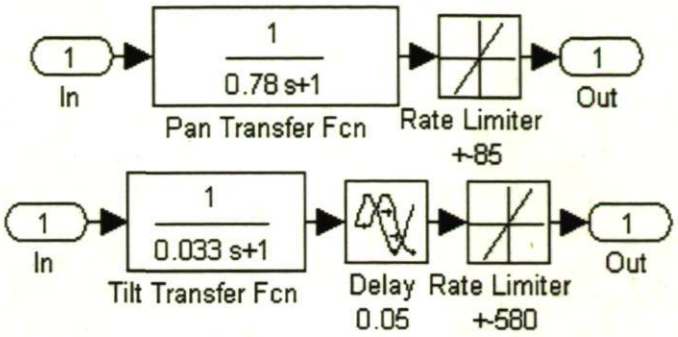


FIGURE 3.12 – Détails des modèles des servomoteurs panoramiques horizontal et vertical

nateur exécutant le logiciel X-Plane, soit une itération sur deux du modèle s'exécutant à 100Hz.

3.4 Utilisation d'X-Plane

Tel que mentionné précédemment, un module d'extension a dû être développé pour qu'il soit possible d'effectuer des tâches que X-Plane ne fait pas par défaut comme le contrôle d'une camera dans le monde 3D. Un module d'extension est un petit programme appelé au démarrage d'un logiciel s'exécutant en parallèle pendant tout le temps d'exécution du programme principal. Il a pour but d'apporter de nouvelles facilités au programme principal.

Un groupe de programmeur a développé un « Software Development Kit » (SDK) [35] avec code source libre en C++ afin qu'il soit possible de développer plus facilement des modules d'extension pour X-Plane. Ce SDK fourni une liste de fonctions et de structures prédéfinies dans le but d'aider des programmeurs à modifier X-Plane.

3.4.1 Création de la caméra virtuelle

Afin de simuler la caméra qui se retrouve sous le ventre du UAV, une caméra a dû être placée dans le monde 3D de X-Plane. Dans le SDK qui a été utilisé, la fonction *UpdateCamera* a servi pour l'affichage dans ce mode. Elle permet de placer une caméra et de l'orienter où on le désire. Cette fonction est appelée 60 fois par seconde et reçoit en argument la position, l'orientation et le zoom de la caméra. Il faut donc s'assurer de

lui fournir des données près de cette fréquence pour que l'affichage soit fluide. La figure 3.13 nous donne un aperçu de la vue de la caméra virtuelle.

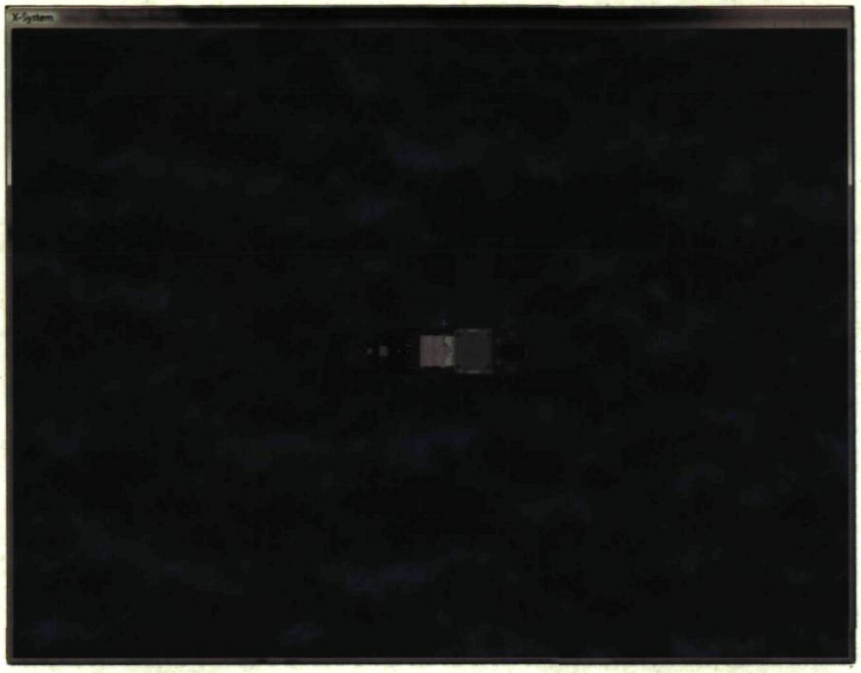


FIGURE 3.13 – Vue de la caméra implantée dans X-Plane

Il est à noter que le passage entre ce mode caméra et le mode standard pour voir l'ensemble de la simulation se fait à l'aide de la touche F8 du clavier.

3.4.2 Modification de la gestion de l'affichage d'X-Plane

En temps normal, X-Plane est utilisé comme simulateur de vol. Les avions ont donc une certaine dynamique et ce, même s'il est utilisé seulement en tant que visualisateur avec les données de vol provenant de paquets Ethernet. Une forme de lissage est créée entre les données qu'on lui transfert, ce qui peut être très pratique lorsque les informations arrivent à faible fréquence pour ne pas que l'affichage soit saccadé. Toutefois, dans le cadre de ce projet, la position de l'avion, et par le fait même de la caméra, est un élément critique qui doit nécessairement être représenté au bon endroit.

Le module d'extension a pu servir de nouveau pour régler ce problème lorsque X-Plane est utilisé pour la caméra. Lorsque le module d'extension reçoit un paquet du modèle Simulink, la fonction *XPLMDisableAIForPlane* du SDK est appelée afin de désactiver le contrôle d'X-Plane sur l'avion choisie en paramètre et de laisser le plein

contrôle de l'affichage au module d'extension. L'affichage doit se faire directement dans les coordonnées d'affichage OPENGL de X-Plane et assez rapidement pour obtenir de la fluidité. C'est à l'intérieur d'une boucle qui est appelé près de 60 fois par seconde que ce fait ce travail à l'aide de fonctions de changement de coordonnées rendues disponibles par le SDK. Il est donc nécessaire que le modèle Simulink du UAV envoie rapidement les informations concernant l'avion et la caméra.

3.4.3 Réception des informations de la caméra

Le module d'extension doit permettre à X-Plane de recevoir les informations concernant la caméra. Pour avoir la plus grande flexibilité possible, la façon choisie pour obtenir ces données a été la réception de paquets Ethernet. Ce choix s'est fait assez facilement sachant que X-Plane doit pouvoir s'exécuter sur son propre ordinateur. Le protocole UDP a été privilégié au dépend du TCP, parce qu'il est moins lourd d'utilisation et que la réception de ces paquets n'est pas critique comme il s'agit seulement d'affichage.

À l'intérieur du module d'extension, une fonction a été créée attendant les paquets UDP servant à la configuration et l'affichage de X-Plane. Un des paquets qu'elle peut recevoir est celui qui est envoyé par le modèle Simulink contenant la position et l'attitude de l'avion ainsi que les deux angles de la caméra. Ce n'est qu'à la réception d'un de ces paquets que la fonction *XPLMDisableAIForPlane* décrite plus haut se fait appelée pour désactiver la dynamique de l'avion.

3.4.4 Utilisation d'X-Plane pour l'affichage de la scène globale 3D

En plus de servir comme caméra virtuelle, X-Plane est utilisé sur un autre ordinateur pour l'affichage du monde 3D. Il est possible de déplacer une caméra dans la scène 3D pour avoir la vue désirée. X-Plane offrait déjà cette possibilité, mais un ajout a tout de même été fait dans le cadre de ce projet au travers du module d'extension pour avoir une meilleure idée de l'orientation de la caméra. Un cône translucide à été placé sous le UAV pour représenter le champ de vue de la caméra. À tout instant, le cône suit les mouvements de la caméra et il est possible de le faire apparaître ou disparaître à l'aide de la touche du clavier F9. Une vue de la scène avec le cône est présentée à la figure 3.14.

FIGURE 3.14 – Vue rendue par X-Plane avec affichage du champ de vue de la caméra

3.5 Modifications apportées au logiciel SerialBridge

Tel que mentionné précédemment, SerialBridge [36] a été développé pour qu'il soit possible de contrôler plus efficacement les UAVs. À ce logiciel, plusieurs nouvelles fonctionnalités ont été apportées en raison de l'intégration du système vidéo de poursuite de cible. Tout d'abord, une nouvelle fenêtre de dialogue a été créée pour remplir les différentes fonctions reliées au système vidéo de poursuite de cible (figure 3.15). Elle est assez simple et permet de voir à tout instant le panoramique horizontal et vertical de la caméra virtuelle, la position en pixel de la cible, le statut du système vidéo ainsi que de commander le départ et l'arrêt de la poursuite. Le logiciel s'occupe aussi de recevoir les informations provenant du système vidéo, de gérer l'intégration d'une manette et de calculer et envoyer les nouvelles commandes à l'autopilote pour ses servomoteurs contrôlant la caméra.

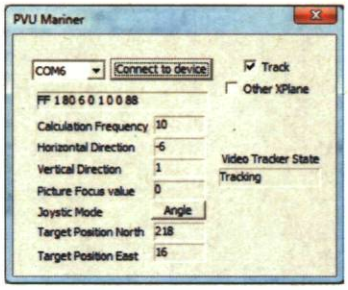


FIGURE 3.15 – Fenêtre ajoutée pour le contrôle du système vidéo de poursuite de cible

3.5.1 Réception de la position en pixel de la cible du système vidéo de poursuite de cible

La position en pixel peut être reçue de 2 façons différentes du système vidéo : d'un port série ou du port Ethernet. Le port série est présentement utilisé puisque que divers problèmes ont été détectés avec l'utilisation du port Ethernet. En effet, le système vidéo de poursuite de cible devenait quelques fois non réactif lors d'utilisations dans ce mode.

Port série

Pour que le système vidéo de poursuite de cible envoie différentes informations dont la position en pixel de la cible par un de ses ports séries, il faut démarrer le programme par la ligne de commande : `grabber -zU`. Le programme envoie alors à 30 Hz du port

série COM1 l'information. Le port série pour la réception doit être ouvert avec les spécifications suivantes : 115 200 bauds, 1 bit d'arrêt, 8 bits de données et pas de bit de parité. Chaque paquet de données est constitué de 9 octets selon la structure donnée dans le tableau 3.2.

Tableau 3.2 – Structure du paquet série

Sync	Command	Data1	Data2	Data3	Data4	Data5	Data6	CheckSum

L'octet de synchronisation est toujours `0xFF`. L'octet de commande, quant à lui, indique le statut du système vidéo. Si les 2 premiers bits sont `00`, il est en attente. `01` indique qu'il est en poursuite et `11` indique qu'il est en ré-acquisition. Les bits numéro 5 ou 6 à 1 indiquent respectivement un focus avant ou arrière et les bits 7 ou 8 à 1 indiquent respectivement un zoom avant ou arrière.

Les octets *Data1* et *Data2* sont respectivement l'octet le plus significatif et le moins significatif de la position en pixel horizontale sur 16 bits signés. Les octets Data3 et Data4 sont quant à eux la position en pixel verticale et les octets Data5 et Data6 la valeur du focus. Un chiffre sur 16 bits signés signifie que le bit le plus significatif indique le signe et les 15 autres la valeur absolue.

Port Ethernet

Comme l'ordinateur dispose d'un port Ethernet, il est aussi possible de lui envoyer des commandes par communication TCP/IP. Il suffit de se connecter à l'adresse IP de l'ordinateur au port 1107. Plusieurs commandes ont été définies, mais seulement quelques unes sont toutefois utiles dans ce projet. Toutes les commandes sont définies en texte. Tout d'abord, la commande *G_TPOS*, (« Get Target Position ») sert à se faire retourner la position en pixel de la cible dans l'image. La réponse est de la forme suivante : *Pan=HorizontalPixelError, Tilt=VerticalPixelError*. Pour commander la poursuite, la commande *S_MODE, ModeData* (« Set Mode ») est utilisée en remplaçant ModeData par 1 ou 0 afin d'activer ou arrêter la poursuite. Le système répond alors par la chaîne de caractère *Mode=ModeData*. De plus, s'il n'y a pas de communication pendant 10 secondes, le système vidéo va fermer automatiquement la communication de son côté. Pour éviter ce problème, il est donc recommandé d'envoyer la commande *S_LIVE* à chaque 5 secondes. Elle ne retourne pas de réponse, mais elle signale au système vidéo qu'on veut garder la communication ouverte.

3.5.2 Intégration de la manette

Pour qu'il soit possible de contrôler la caméra sous l'avion de façon conviviale, une manette à 6 degrés de liberté a été intégrée au système. C'est encore une fois dans le logiciel SerialBridge que s'est fait son intégration. Elle a été facilitée par la trousse de développement logiciel développé par la compagnie 3DConnexion qui fabrique la manette. Dans ce SDK, une fonction est automatiquement appelée lorsqu'un bouton est pesé ou lorsque la molette analogique est déplacée. La fonction appelée pour les boutons reçoit en argument le numéro du bouton. Il n'y a donc aucun de traitement à faire. Pour ce qui est du mouvement de la molette, la fonction appelée reçoit en argument six valeurs proportionnellement aux trois déplacements et aux trois rotations. Toutefois, ces nombres doivent être filtrés car ils sont très bruités. La façon simple implantée est d'utiliser un filtre à moyenne mobile avec une fenêtre de 20 valeurs normalisées entre -1 et 1. Cette technique ajoute un faible déphasage, mais puisque les valeurs arrivent à un taux très élevé, il est transparent pour l'usager.

Pour l'instant, seulement 2 degrés de liberté de la molette analogique sont utilisés pour le contrôle de la caméra. Ce sont les 2 rotations servant à faire bouger les servo-moteurs panoramiques horizontal et vertical et elles sont illustrées sur la figure 3.16.

FIGURE 3.16 – Manette utilisée pour contrôler les simulations de surveillance maritime

Pour ce qui est des boutons, il n'y en a présentement que 4 d'utilisés sur la manette. Le bouton 1 sert à indiquer si on veut que les corrections reçues du vidéo tracker, lorsqu'il est en mode poursuite, soient utilisées pour corriger la caméra virtuelle d'X-Plane. Le bouton 2 bascule le mode de la manette entre *Target* et *Angle* [36]. Le bouton F sert

quant à lui à démarrer ou arrêter la poursuite sur le système vidéo. Il n'est donc plus nécessaire d'utiliser la manette du système vidéo pour effectuer cette action. Lorsque ce bouton et pesé, le paquet TCP/IP *S_MODE* défini en 3.5.1 est créé et envoyé au système vidéo.

Le quatrième bouton (T) est utilisé pour aider l'usager à retrouver la cible. Comme tout se déroule en simulation et que la cible est un objet envoyé à X-Plane, il est toujours possible de savoir où est cet objet. En effet, lorsque qu'X-Plane est mode caméra, il retourne vers SerialBridge la position de la cible qui lui a préalablement été envoyée. Lorsque le bouton est pesé, la caméra s'oriente vers cet objet sans toutefois le suivre. Il suffit par la suite à l'usager de continuer de suivre la cible manuellement et de partir la poursuite s'il le désire. L'orientation de la caméra à envoyer vers l'autopilote lors de l'utilisation du bouton T est donnée par :

$$ {}^{A}_{C}C = {}^{A}_{M}C\,{}^{M}_{C}C = {}^{M}_{A}C^{-1}\,{}^{M}_{C}C = R^{-1}(0, \beta_A, \alpha_A)R(0, \beta_{C_M}, \alpha_{C_M}) \tag{3.3} $$

où β_A et α_A sont les angles d'Euler de l'avion par rapport au monde et les angles β_{C_M},α_{C_M} sont les angles que doit avoir la caméra par rapport au monde. Ils se trouvent à partir du vecteur qui relie le UAV à la cible. La rotation inverse fait passer le vecteur dans le référentiel du monde à celui de l'avion puisque la caméra se commande avec des angles relatifs à l'avion.

3.5.3 Modélisation des servomoteurs

Dans le chapitre 2, il a été fait mention que la position des servomoteurs panoramique horizontal et vertical devait être connue pour retrouver la position de la cible. Toutefois, comme il n'est pas possible d'avoir accès à ces valeurs, des modèles doivent être créés. Cette tâche peut paraître simple puisque les servomoteurs ont déjà été identifié dans la section 3.3.1, mais une nouvelle identification est nécessaire puisque le contrôle des servomoteurs par le biais de l'autopilote ajoute un délai non négligeable. De nouveaux échelons de consigne ont donc été envoyés et on peut voir la réponse des deux servomoteurs à la figure 3.17.

Suite à l'étude de ces réponses, on peut voir que la dynamique est la même qui avait été identifiée plus tôt, mais qu'un délai d'environ 120 ms s'est ajouté. Les nouvelles fonctions de transfert entre les consignes envoyées à l'autopilote et les angles de la caméra sont :

$$ G_P(s) = \frac{e^{-0.12s}}{1 + 0.78s} \tag{3.4} $$

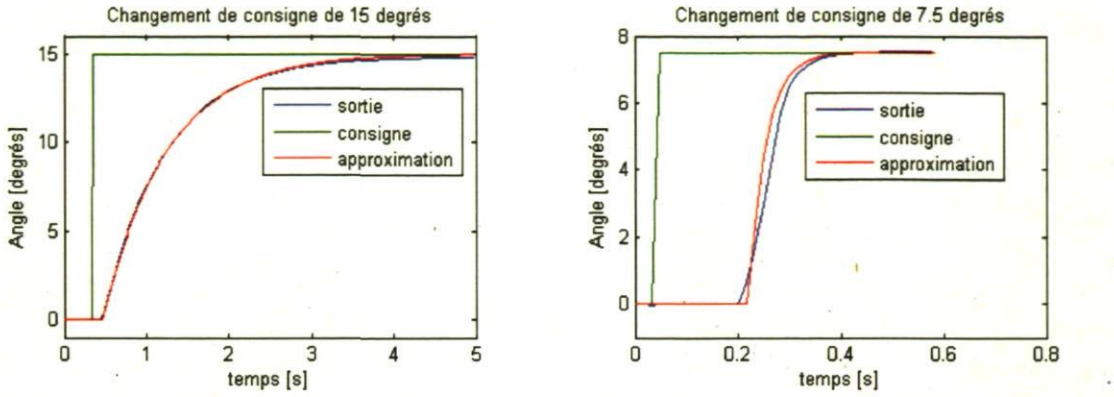


FIGURE 3.17 – Réponses des servomoteurs panoramiques horizontal et vertical à l'échelon

$$G_T(s) = \frac{e^{-0.17s}}{1 + 0.033s} \tag{3.5}$$

Maintenant que les fonctions de transferts sont connues, elles sont discrétisées et mises à jour à une fréquence de 30Hz, soit à la réception de chaque paquet provenant du système vidéo de poursuite de cible. Les équations des deux modèles sont :

$$G_P(z^{-1}) = \frac{(1 - a_P)z^{-4}}{1 - a_P z^{-1}} \tag{3.6}$$

$$G_T(z^{-1}) = \frac{(1 - a_T)z^{-6}}{1 - a_T z^{-1}} \tag{3.7}$$

où $a_P = e^{\frac{-1/30}{0.78}}$ et $a_T = e^{\frac{-1/30}{0.033}}$.

3.5.4 Contrôle de la caméra en mode attente

Lorsque la poursuite n'est pas en fonction, il faut pouvoir contrôler la caméra à partir de la manette afin de pouvoir effectuer la surveillance et pour choisir quelle cible surveiller. La manette contrôle donc le panoramique horizontal et vertical de la caméra en leur ajoutant un certains nombre de degrés proportionnellement à la valeur reçue de la manette une fois normalisée et filtrée (δ). Le gain proportionnel est choisi pour que la vitesse maximale de déplacement soit d'environ 40 degrés par seconde. Les fonctions calculant les angles panoramiques horizontal et vertical α_C et β_C à envoyer à X-Plane sont :

$$\alpha_C(k) = \alpha_C(k-1) + 90\delta_\alpha(k) \tag{3.8}$$

$$\beta_C(k) = \beta_C(k-1) + 10\delta_\beta(k) \tag{3.9}$$

La différence entre les gains s'explique par les temps de réponse différents des deux actionneurs. En effet, la commande envoyée au panoramique horizontal doit être beaucoup plus grande que pour le vertical pour obtenir sensiblement la même vitesse de déplacement de la caméra dans les deux directions.

Une fois qu'une cible potentielle est trouvée et placée dans le centre de la caméra, il est possible d'appuyer sur le bouton F de la manette et ainsi laisser le système suivre la cible automatiquement.

3.5.5 Contrôle de la caméra en mode poursuite

Lorsqu'on passe en mode poursuite, la caméra doit être déplacée de façon automatique afin de garder la cible le plus près du centre de l'image pour qu'il soit possible de la suivre sans jamais la perdre de vue. Lorsque le système vidéo de poursuite de cible est utilisé avec un contrôleur d'unité panoramique horizontal et vertical sur laquelle est montée une vraie caméra, le système s'occupe automatiquement du son déplacement. Toutefois, dans le système HIL, l'autopilote joue le rôle du contrôleur et la caméra est virtuelle. Il n'est donc pas possible d'utiliser le système vidéo dans son mode usuel, mais il est tout de même possible de recueillir la position de la cible en pixels dans l'image calculée par le système vidéo et de s'en servir pour contrôler la caméra en envoyant des consignes à l'autopilote. Le schéma de contrôle complet du système est représenté à la figure 3.18. Cette figure montre les étapes effectuées sur l'ordinateur 1 menant aux consignes envoyées à l'autopilote pour le déplacement de la caméra. Par la suite, les modèles des servomoteurs de la caméra jumelés au modèle Simulink s'exécutant sur l'ordinateur 2 sont mis à jour, l'image générée par la caméra virtuelle sur l'ordinateur 4 change et le système vidéo de poursuite de cible calcul à nouveau la position de la cible.

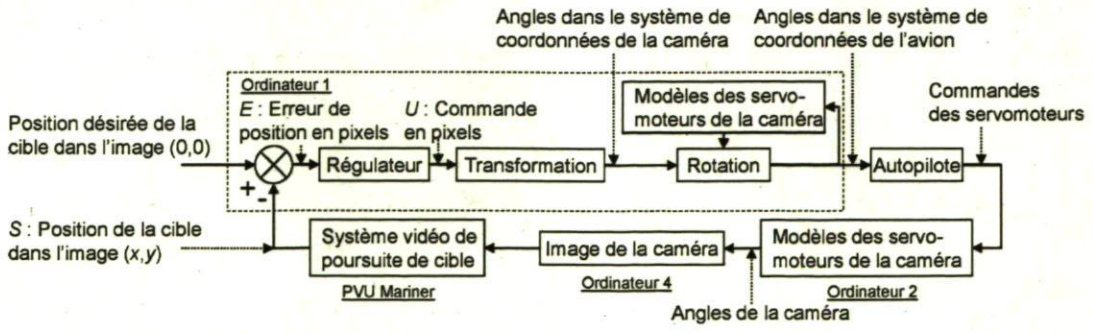


FIGURE 3.18 – Schéma de contrôle de la caméra

Les calculs effectués sur l'ordinateur 1 sont basés sur la théorie amenée au chapitre

2. En effet, le passage de la position de la cible en pixels aux consignes à envoyer à l'autopilote se fait par une suite de deux transformations. Tout d'abord, une transformation est nécessaire afin de connaître à quels angles dans le référentiel de la caméra correspondent la position en pixels horizontale et verticale de la cible dans l'image. Par la suite, les angles trouvés subissent une rotation selon l'orientation de la caméra. L'orientation doit être retrouvée à l'aide d'un modèle puisqu'il n'est pas possible de connaître la position des servomoteurs de l'autopilote. Cette seconde transformation fait passer les angles dans le référentiel de l'avion pour ainsi devenir des consignes à être envoyées à l'autopilote. Suivant la nomenclature donnée au chapitre 2, les deux transformations utilisées pour passer du référentiel de l'image à celui de l'avion peuvent être définies par :

$$\overset{A}{C}C_I^C C = R(0, \beta_C, \alpha_C) R(0, \beta_P, \alpha_P) \tag{3.10}$$

Les angles résultants sont calculés à l'aide des relations 2.5, 2.6 et 2.7.

Implantation d'un régulateur

Pour plus de performance, au lieu d'envoyer directement à l'autopilote la nouvelle consigne, un régulateur non unitaire peut être conçu pour chacun des deux procédés, soit le panoramique horizontal et vertical. Il est donc possible d'accélérer la réponse du panoramique horizontal et de décélérer celle du panoramique vertical pour avoir le plus petit temps de réponse possible sans toutefois amener d'oscillations nuisibles. L'ajout de ce régulateur permet aussi de diminuer les risques de pertes de poursuite causés par les dynamiques respectivement trop lente et trop rapide des servomoteurs panoramiques horizontal et vertical. Une perte de poursuite signifie que la cible est sortie du champ de vue de la caméra et qu'un contrôle manuel est nécessaire pour la retrouver.

Pour concevoir le régulateur, une identification du système complet en boucle ouverte est nécessaire pour chacun des deux servomoteurs. Cette identification se fait en appliquant un échelon sur la commande en pixels U et en observant la réponse provenant du système vidéo de poursuite de cible S sur la position en pixels réelle. L'identification pour le servomoteur panoramique horizontal est montrée à la figure 3.19. Cette figure indique que le procédé est de la forme :

$$\frac{S(s)}{U(s)} = \frac{Ke^{-\theta s}}{s(1+\tau s)} \tag{3.11}$$

Puisque que les blocs transformation, rotation, camera, image et système vidéo de poursuite de cible n'ajoutent pas de retard supplémentaire et ne change pas la

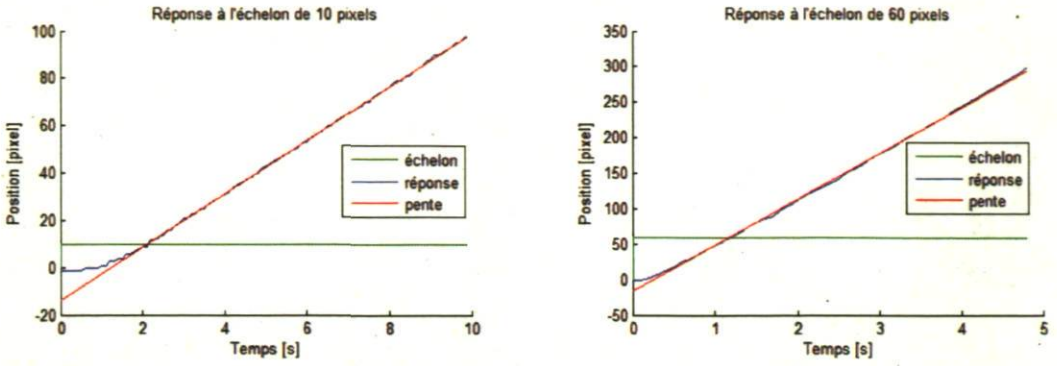


FIGURE 3.19 – Réponse du servomoteur panoramique horizontal à l'échelon

dynamique, le retard θ ainsi que la constante de temps τ restent ceux qui ont été identifiés à la section 3.5.3. Pour ce qui est du gain K, il correspond à la pente normalisée de la réponse en régime permanent qu'on peut voir en rouge. Sur la figure 3.19, les deux identifications donnent respectivement des pentes de 11.2 et 64.8 pixels/s. Une fois normalisée par l'amplitude des échelons de 10 et 60 pixels, un gain identique pour les deux tests de 1.1 s^{-1} est retrouvé.

Les mêmes tests ont étés effectués pour le panoramique vertical et on peut voir la réponse aux échelons à la figure 3.20.

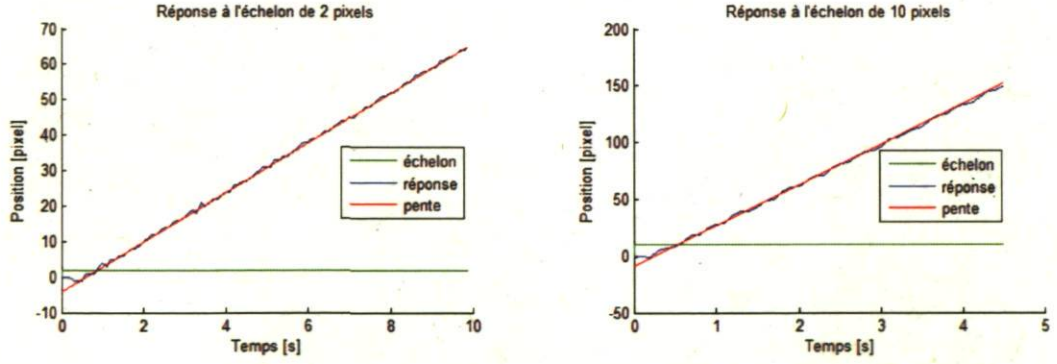


FIGURE 3.20 – Réponse du servomoteur panoramique vertical à l'échelon

Les deux pentes sont respectivement de 6.98 et de 35.5 pixels/s. Une fois normalisé par l'amplitude des échelons de 2 et 10 pixels, on trouve un gain de 3.5 s^{-1}. Les fonctions de transfert G_{pp} et G_{pt} des deux procédés représentant respectivement le déplacement horizontal et vertical de la caméra sont :

$$G_{pp}(s) = \frac{1.1e^{-0.12s}}{s(1+0.78s)} \tag{3.12}$$

$$G_{pt}(s) = \frac{3.5e^{-0.17s}}{s(1+0.033s)} \tag{3.13}$$

Conception du régulateur

Les spécifications recherchées pour le système sont une marge de phase de 65°ainsi qu'une dynamique similaire pour les deux servomoteurs. Des tests préliminaires ont permis de trouver que la dynamique du servomoteur panoramique horizontal était trop lente et celle du servomoteur panoramique vertical trop rapide. En effet, ces dynamiques amenaient des pertes de poursuite. Le régulateur suivant permet de répondre à cette spécification :

$$G_c(s) = K_c \frac{1+\tau_d s}{1+\tau_f s} \tag{3.14}$$

Dans cette dernière équation, τ_d est fixé par annulation des pôles et zéros à la constante de temps du procédé et τ_f est fixé à 0.15 secondes. Pour ce qui est de K_c, il est choisi pour que la marge de phase soit de 65°. Les deux régulateurs deviennent :

$$G_{cp}(s) = 1.2\frac{1+0.78s}{1+0.15s} \tag{3.15}$$

$$G_{ct}(s) = 0.4\frac{1+0.033s}{1+0.15s} \tag{3.16}$$

Les diagrammes de Nichols des deux systèmes sont présentés à la figure 3.21.

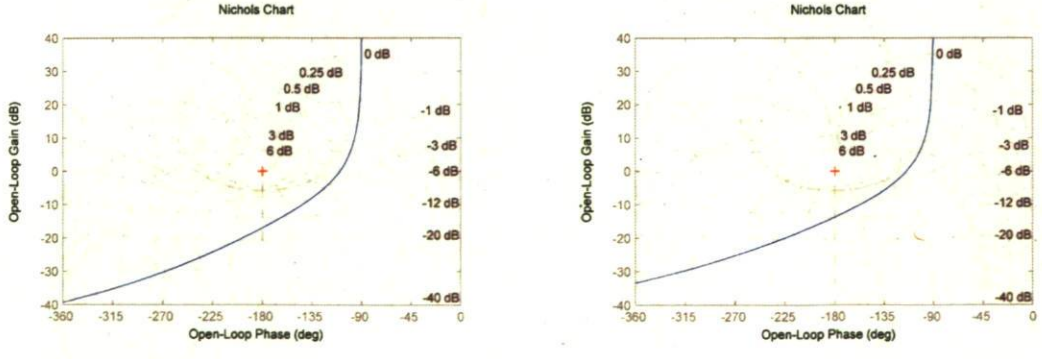


FIGURE 3.21 – Diagrammes de Nichols des procédés avec régulateur pour le panoramique horizontal et vertical

Comme les régulateurs seront implantés dans le programme C++ SerialBridge, ils doivent être discrétisés. L'approche Tustin [37] pour laquelle :

$$s = \frac{2}{T}\frac{1 - z^{-1}}{1 + z^{-1}} \tag{3.17}$$

a été retenue. En remplacement s dans l'équation 3.14 on obtient :

$$G_c(s) = K_c \frac{\left(\frac{T+2\tau_d}{T+2\tau_f}\right) + \left(\frac{T-2\tau_d}{T+2\tau_f}\right) z^{-1}}{1 + \left(\frac{T-2\tau_f}{T+2\tau_f}\right) z^{-1}} \tag{3.18}$$

Sous forme récurrente, l'équation 3.18 devient :

$$u(k) = -\left(\frac{T-2\tau_f}{T+2\tau_f}\right) u(k-1) + K_c\left(\frac{T+2\tau_d}{T+2\tau_f}\right) e(k) + K_c\left(\frac{T-2\tau_d}{T+2\tau_f}\right) e(k-1) \tag{3.19}$$

Comme la position en pixels provient du système vidéo de poursuite de cible à une fréquence de 30 Hz, mais qu'il n'est pas possible de d'envoyer des changements de consigne vers l'autopilote à cette fréquence, T a été fixé à 10^{-1} pour que le contrôle de la caméra se fasse dès la réception d'une correction sur 3. De plus, comme le contrôle se fait à la suite d'une réception de la position de la cible dans l'image, les modèles viennent juste de se rafraîchir. Des tests préliminaires en régulation ont été effectués avec le UAV du système HIL qui se déplace autour de la cible à une altitude 200 mètres. Le parcours suivi par le UAV est illustré à la figure 3.22. Une vue du logiciel de contrôle au sol est montrée avec les 4 points de cheminement placés de façon à faire tourner le UAV autour de la cible décentrée. De cette façon il est possible de tester la régulation sans se soucier du contrôle du UAV. Ces tests ont pu permettre de valider la conception du régulateur puisque le système réussit à garder la cible au centre de la caméra sans jamais la perdre de vue.

3.5.6 Évaluation de la position de la cible

Tel que mentionné dans la section 2.5, la position de la cible se retrouve par la solution de l'équation de la droite formée par la position du UAV et le vecteur suivant l'orientation de la caméra (Eq. 2.30 et Eq. 2.31). Le calcul est assez simple, mais il est très affecté par l'inexactitude des mesures. En effet, cette solution est très précise si la position et l'attitude du UAV sont connues avec une grande précision, mais se dégrade rapidement si les informations provenant de l'autopilote ne sont pas exactes.

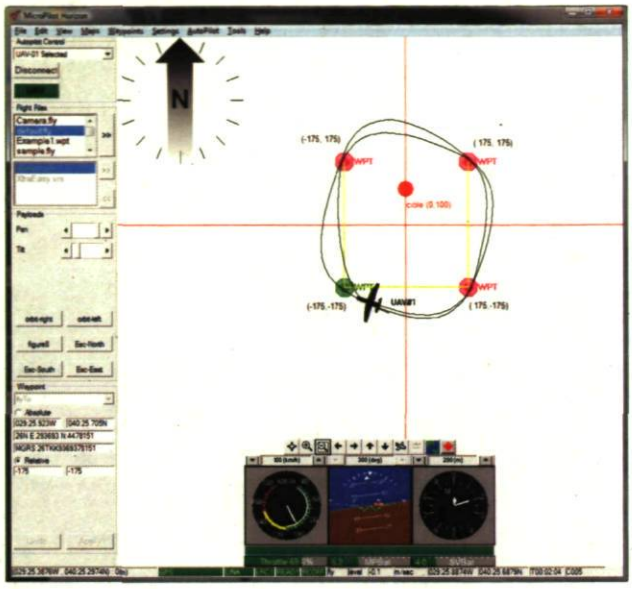


FIGURE 3.22 – Test en régulation

Les informations qui proviennent du UAV arrivent pour la plupart automatiquement à basse fréquence. C'est le cas de la position GPS qui arrive seulement une fois par seconde avec 2 octets de précision. Le roulis et le tangage arrivent quant à eux 5 fois par secondes, ce qui est mieux puisque plus rapide, mais ces données en plus du lacet sont encodées seulement sur 1 octet, ce qui donne une très faible résolution. On peut voir sur la figure 3.23 la différence et le retard entre la position GPS reçue de l'autopilote et la position réelle provenant du modèle Simulink et qui est envoyée à X-Plane pour l'affichage.

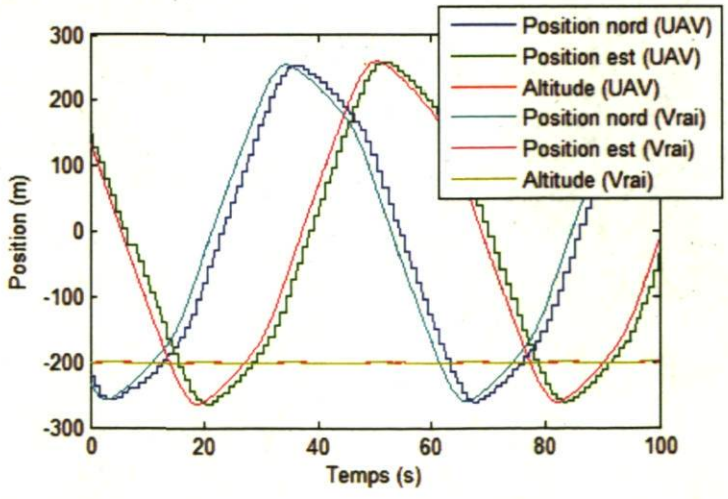


FIGURE 3.23 – Position reçue de l'autopilote et position réelle du UAV

La combinaison de ces facteurs amène une grande erreur sur la position retrouvée. La figure 3.24 indique la position Nord et Est calculée de la cible. Il est à noter que les oscillations à la fréquence d'environ 1 Hz sur le graphique correspondent à l'arrivée des nouvelles informations de l'autopilote.

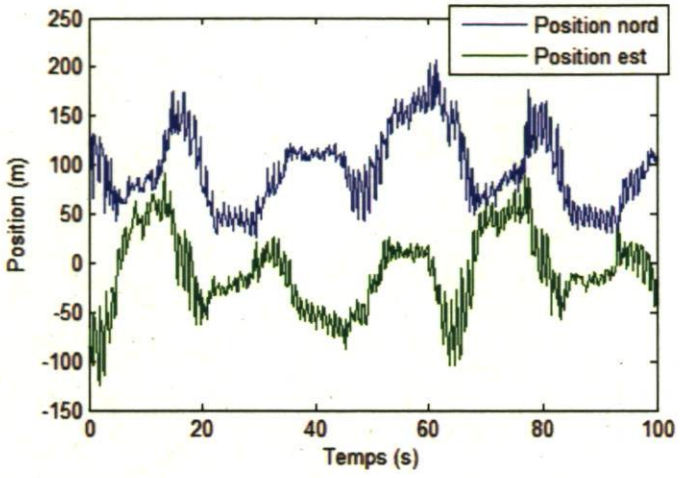


FIGURE 3.24 – Position calculée de la cible avec la réception d'information à 1 Hz

Pour ce test, la position réelle de la cible était (0,100). On peut voir que la moyenne est bien centrée sur ces valeurs, mais l'écart type n'est toutefois pas acceptable. Il est de près de 40 mètres pour les deux coordonnées. Ces tests préliminaires peuvent sembler déplorables, mais il est tout de même possible d'améliorer le système assez facilement. En effet, au lieu d'attendre les valeurs provenant du UAV, il est possible d'y avoir accès plus rapidement ainsi qu'avec une meilleure précision en faisant la demande à l'autopilote. Les positions Nord et Est peuvent nous parvenir 4 fois par seconde comme le permet les GPS les plus performants pour ce type d'autopilote au lieu d'une seule fois par seconde. De plus, il est aussi possible de recevoir l'attitude de façon précise à cette même fréquence.

La figure 3.25 nous montrent la position de la cible calculée cette fois avec les informations du UAV arrivant 4 fois par seconde. On peut voir que ce changement améliore considérablement la recherche de la position de la cible puisque l'écart type des deux coordonnées passe maintenant sous 20 mètres. On peut voir que les oscillations à haute fréquence sont toujours présentes en raison encore une fois de l'arrivée des informations de l'autopilote, mais aussi à cause de la position en pixel de la cible dans l'image qui provient du système vidéo de poursuite de cible. Cette position est en effet assez bruitée puisque le bateau dans l'image couvre un grand nombre de pixel et le système n'indique pas toujours la même partie du bateau. Des tests ont montré qu'un simple filtre de premier ordre pouvait améliorer ces résultats. Il aurait été intéressant

d'approfondir le filtrage en implantant un estimateur de position de la cible présenté par Dobrokhodov et al. [27] ou encore un filtre de Kalman présenté par Monda et al. [38] ou Prévost et al. [14], mais la précision actuelle semble suffisante pour obtenir une performance de poursuite satisfaisante.

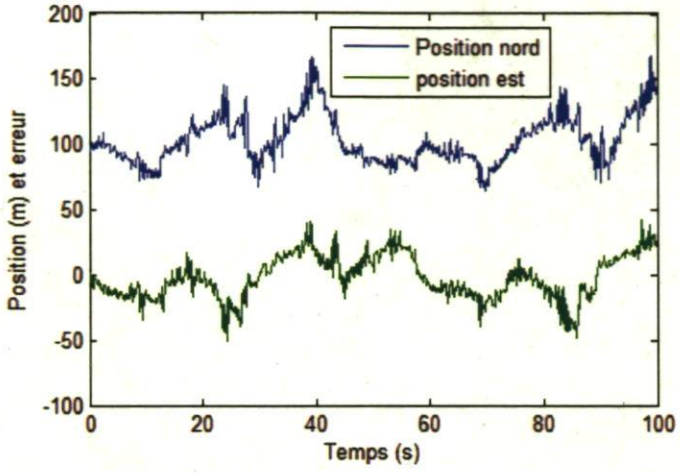


FIGURE 3.25 – Position calculée de la cible avec la réception d'information à 4 Hz

Pour savoir à quel point des données précises améliorent la recherche de la position de la cible, il est possible de refaire le même test, mais cette fois en utilisant directement les valeurs du modèle Simulink qui sont envoyées pour l'affichage vers X-Plane. Le résultat de ce test est montré à la figure 3.26. L'écart type est diminué à environ 5 mètres pour les deux coordonnées. Contrairement aux deux autres tests, cette fois les oscillations hautes fréquences proviennent essentiellement du bruit dans la position en pixel, puisque les informations du UAV sont connues de façon précise à haute fréquence.

Cette performance optimum utilisant les données de l'affichage ne pourra toutefois jamais être atteinte, puisqu'il y aura toujours une petite différence entre les données provenant du modèle Simulink et celles provenant de l'autopilote. Les composantes d'attitude du UAV sont les données les plus affectées, puisque l'algorithme de fusion de données utilisé par l'autopilote pour retrouver son attitude à partir des données de ses accéléromètres et de ses gyroscopes n'est pas parfait. Il introduit une petite erreur qui s'accentue lorsque le UAV tourne et provoque un grand roulis. On peut en voir l'effet par les oscillations à basse fréquence qu'on retrouve dans les graphique précédents.

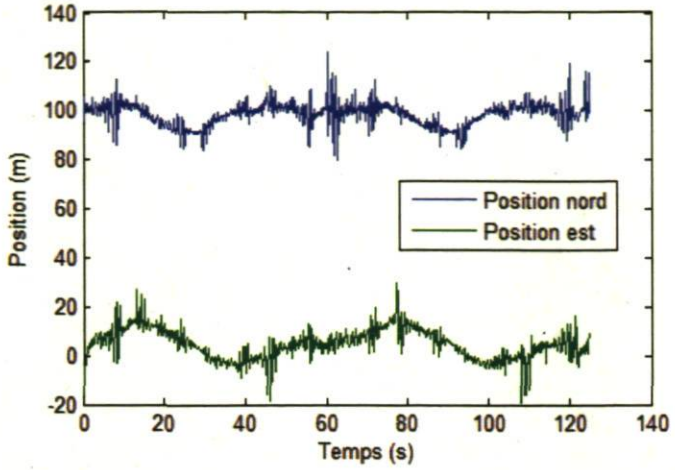


FIGURE 3.26 – Position calculée de la cible en utilisant les données du modèle Simulink

3.6 Communication entre les différents éléments du système HIL

La figure 3.27 résume toutes les communications échangées lors des simulations. Le logiciel SerialBridge envoie beaucoup d'information. Tout d'abord, au système vidéo de poursuite de cible, il doit envoyer une commande TCP/IP *S_LIVE* sur le port 1107 à chaque 5 secondes afin de garder le lien de communication ouvert. Il envoie aussi une commande TCP/IP *S_MODE* sporadiquement lorsque l'usager veut démarrer ou arrêter la poursuite. Aux ordinateurs 3 et 4 roulant X-Plane, il envoie en début de simulation une commande UDP sur le port 49000 afin d'initialiser la cible. Ensuite, il envoie à une fréquence de 10Hz un paquet UDP sur le port 49003 contenant la position de la cible. Il communique aussi avec l'autopilote par le lien série à une fréquence de 1Hz les commandes en orientation du UAV, à une fréquence de 10Hz le déplacement de la caméra et à une fréquence de 4Hz la demande pour des informations de position et d'attitude précises. Enfin, SerialBridge communique avec l'ordinateur 2 roulant le modèle Simulink du UAV sous xPC Target pour la mise en marche du modèle.

Le système vidéo de poursuite de cible, tout comme l'autopilote et l'ordinateur 4, envoie de l'information seulement à un module. Il transfert à l'ordinateur 1 l'état de la poursuite ainsi que la position de la cible dans l'image par son lien série à une fréquence de 30 Hz. L'autopilote envoie quant à lui à une fréquence de 5 Hz de l'information télémétrique standard et à une fréquence de 4Hz de l'information précise par son lien série pour l'affichage des données de vol du UAV dans logiciel de contrôle au sol Horizon ainsiq ue pour les algorithmes de surveillance. De son côté, l'ordinateur 4 affichant la

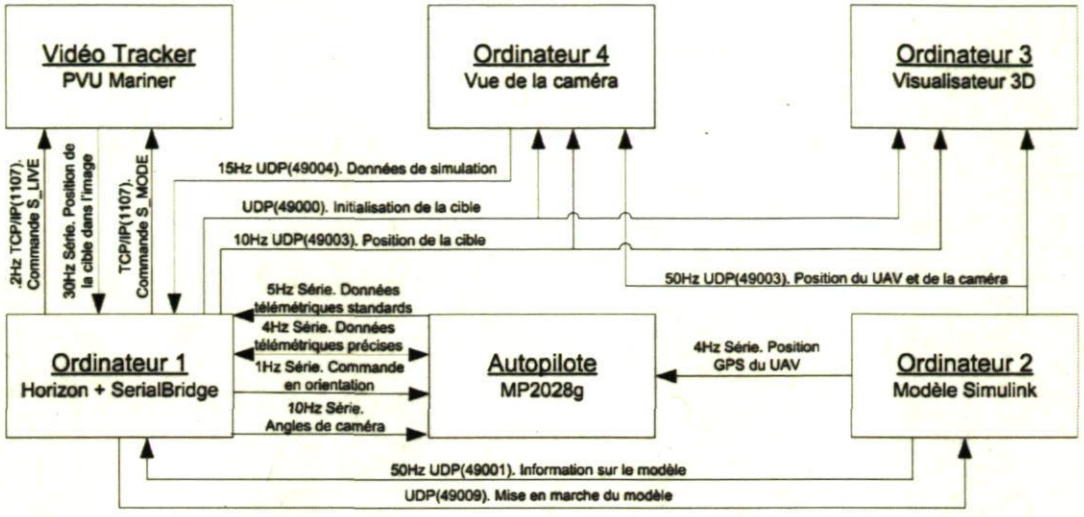


FIGURE 3.27 – Architecture des communications lors de l'évaluation d'algorithmes de surveillance

vue de la caméra envoie de l'information concernant la position et l'attitude du UAV, de la cible ainsi que de la caméra à l'intérieur d'un paquet UDP sur le port 49004 à une fréquence de 15 Hz pour qu'il soit possible de retrouver la cible à l'aide de la manette.

L'ordinateur 2 exécutant le modèle Simulink du UAV sous xPC Target envoie vers les ordinateurs 3 et 4 à une fréquence de 50 Hz un paquet UDP sur le port 49003 contenant la position et l'attitude du UAV ainsi que les angles de la caméra. Il envoie aussi de l'information vers le logiciel SerialBridge concernant le modèle à l'intérieur d'un paquet UDP sur le port 49001 à une fréquence de 50Hz. Ce paquet contient la position du UAV, son attitude, sa vitesse, sa vitesse angulaire ainsi que le vent affectant le modèle. Ces informations peuvent servir à enregistrer les données ou à des fins de tests. Enfin, l'ordinateur 2 simule le GPS de l'autopilote en envoyant les paquets nécessaires par le protocole série à une fréquence de 4Hz.

3.7 Conclusion

Ce chapitre a présenté le système HIL construit afin de simuler de façon réaliste des vols de UAV et de tester des algorithmes de surveillance avancés en laboratoire sans la complexité, les risques et les coûts associés aux vols extérieurs. Dans le cadre de cette maîtrise, un système vidéo de poursuite de cible a été intégré au système HIL dans le but de rechercher et développer des stratégies de guidage pour la surveillance maritime. Cette intégration a nécessité l'ajout de deux ordinateurs : le système vidéo de poursuite

de cible ainsi qu'un ordinateur simulant le champ de vue de la caméra. Afin de garder à tout moment la cible dans son champ de vue, un régulateur a été intégré à la boucle de contrôle de la caméra. Tant que la poursuite n'est pas interrompue, la position de la cible peut être retrouvée. La précision de cette position est toutefois grandement influencée par la précision de la position et de l'attitude du UAV provenant de l'autopilote. C'est pourquoi une demande est faite à l'autopilote à une fréquence de 4Hz afin de connaître ces données de façon plus précise. Cette démarche permet enfin de retrouver la position de la cible de façon juste puisque la moyenne est bonne et assez fidèle puisque l'écart type est en deçà de 20 mètres. Cet écart type est acceptable à des fin de surveillance, puisque les algorithmes de guidage garde normalement le UAV à une distance de la cible bien supérieure suivant la manœvrabilité et la vitesse du UAV [12]. En effet, le UAV du système HIL à un rayon de braquage de 175 mètres à 100 km/h.

Chapitre 4

Algorithmes de base pour la surveillance de cible

Ce chapitre traite des modes d'opération du UAV et introduit les différents algorithmes de contrôle de base testés et analysés en simulation et à l'aide du système HIL. L'évaluation et l'analyse de ces algorithmes se fait au chapitre 5. Il est important de rappeler que l'intégration du système vidéo de poursuite de cible a pour but de retrouver la position de la cible afin d'en permettre la surveillance. La majorité des algorithmes de surveillance n'a besoin que de la position de la cible, mais certains nécessitent plus d'information comme sa vitesse ou son orientation qu'un filtrage sur la position permet de retrouver. À partir de ces informations, les algorithmes permettent de diriger le UAV afin qu'il suive la cible sans jamais la perdre de vue.

4.1 Modes d'opération du UAV

L'autopilote offre deux modes d'opération : le mode complètement autonome (UAV) ainsi que le mode piloté à distance (RPV - « Remotly Piloted Vehicle »). Le mode autonome est défini par un fichier de vol dans lequel se trouve différentes consignes de vitesse et d'altitude ainsi que par une suite de points de cheminement à parcourir. Ce mode est le plus souvent utilisé lors de missions traditionnelles. Le mode RPV permet, quant à lui, de contrôler le UAV par des consignes en vitesse, en altitude ainsi qu'en orientation. Dans ce mode, il est possible d'envoyer des consignes à partir du sol et de contrôler la majorité des aspects de vol du UAV. Ce mode est utilisé en présence d'algorithmes de guidage particuliers.

4.2 Algorithmes de surveillance

Les algorithmes de contrôle étudiés ici ne contrôlent que l'orientation en mode RPV. La vitesse et l'altitude peuvent demeurer constantes. Les 4 premiers utilisent différentes techniques qui amènent le UAV à tourner autour de la cible suivant un cercle de rayon r. Cette solution est très bonne pour surveiller une cible se déplaçant lentement. Le cinquième algorithme est plus simple et oriente directement le UAV vers la cible.

Pour les sections suivantes, la position dans le plan horizontale du UAV est définie par (x_p, y_p) et celle de la cible par (x_t, y_t). La position du UAV par rapport à la cible est quant à elle définie comme :

$$(x_r, y_r) = (x_p - x_t, y_p - y_t) \tag{4.1}$$

4.2.1 Algorithme de cercle selon Quigley et al.

Cet algorithme, basé sur les bifurcations de Hopf super-critiques, est présenté par Quigley et al.[11]. Ces bifurcations produisent des trajectoires spirales qui convergent vers un cercle d'attraction. Dans les sections à venir, cet algorithme est dénoté BHSC afin d'alléger le texte. L'algorithme défini un champ de vecteur de guidage $\mathbf{f}(x_r, y_r)$ pour calculer la vitesse horizontale désirée $[\dot{x_d}, \dot{y_d}]^T$.

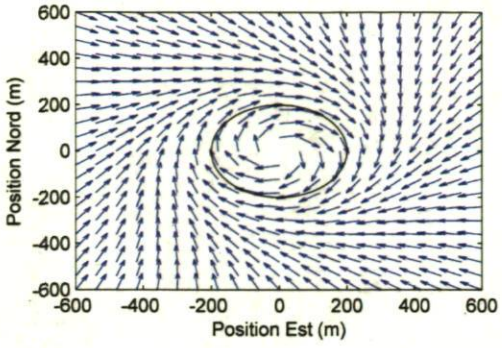
$$\mathbf{f}(x_r, y_r) = \begin{bmatrix} \dot{x_d} \\ \dot{y_d} \end{bmatrix} = \begin{bmatrix} y_r + \frac{x_r}{\mu r^2}(r^2 - x_r^2 - y_r^2) \\ -x_r + \frac{y_r}{\mu r^2}(r^2 - x_r^2 - y_r^2) \end{bmatrix} \tag{4.2}$$

où μ est une tolérance de déviation positive. Une grande tolérance ($\mu > 1$) va adoucir les consignes comparativement à une plus petite. La consigne en orientation ψ est finalement donnée par :

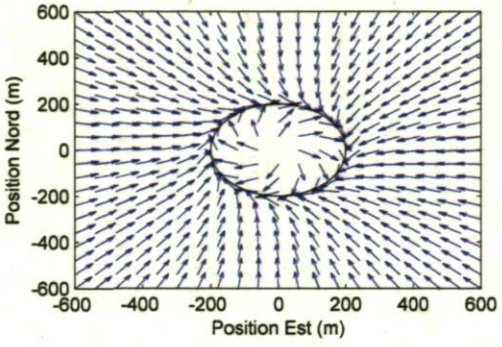
$$\psi = \text{atan2}(\dot{y_d}, \dot{x_d}) \tag{4.3}$$

Dans l'équation 4.3 l'arctangente se calcule à l'aide de la fonction atan2 pour retrouver la consigne en orientation sur l'étendu complète du cercle trigonométrique (voir section 2.3). Cette méthode a la particularité intéressante de diriger le UAV directement vers la cible lorsque celle-ci est éloignée pour s'en approcher le plus rapidement possible.

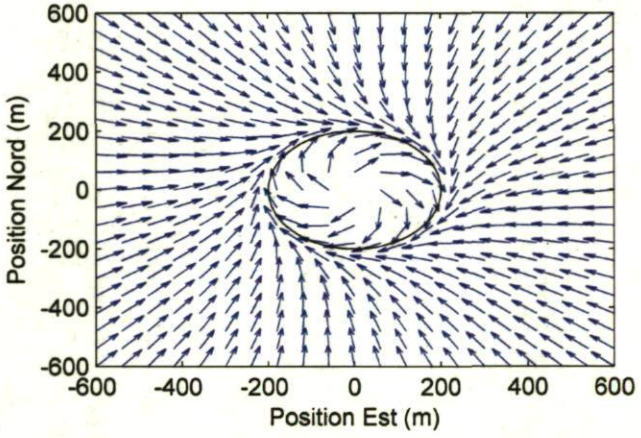
La figure 4.1 montre les consignes en orientation sous forme de vecteurs pour une matrice de points situés autour d'une cible placée en (0,0). On peut voir l'effet de trois facteurs de tolérance μ différents pour un cercle de 200m de rayon.



(a) $\mu = 5$



(b) $\mu = .5$



(c) $\mu = 1$

FIGURE 4.1 – Vecteurs d'orientation de l'algorithme BHSC

4.2.2 Algorithme de cercle selon Rafi et al.

L'algorithme proposé par Rafi et al.[12] est représenté à la figure 4.2 et dénoté ACR. Il n'a seulement qu'un paramètre à ajuster, le rayon du cercle. Contrairement à l'algorithme précédent, celui ci ne fixe pas le sens de rotation du UAV autour de la cible. Il oriente plutôt le UAV dans le sens le plus approprié en choisissant l'angle le plus petit entre ψ_1 et ψ_2 selon :

$$\psi = d + \min(\psi_1, \psi_2) \tag{4.4}$$

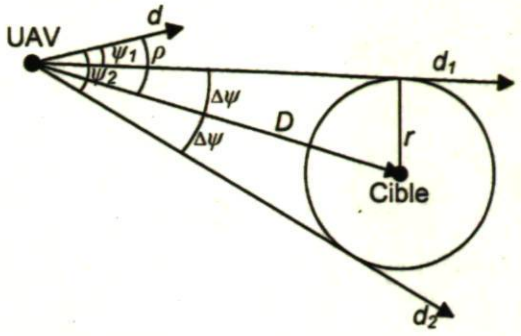


FIGURE 4.2 – Description de la méthode de guidage ACR

Les deux équations suivantes définissent ces deux angles :

$$\psi_1 = \rho - \Delta\psi \tag{4.5}$$

$$\psi_2 = \rho + \Delta\psi \tag{4.6}$$

où ρ est l'angle entre la direction du UAV d et la direction de la cible D. Il se calcule à l'aide du produit scalaire de d et D selon :

$$\rho = \arccos\left(\frac{d \cdot D}{\|d\| \, \|D\|}\right) \tag{4.7}$$

$\Delta\psi$ se calcule quant à lui comme :

$$\Delta\psi = \arcsin\left(\frac{r}{\|D\|}\right) \tag{4.8}$$

La figure 4.3 montre encore une fois la consigne en orientation calculée par l'algorithme pour un cercle de 200m de rayon. Il est possible de voir qu'il tient compte de l'orientation du UAV qui était de 20° pour ce test.

4.2.3 Méthode utilisant les champs de vecteur de Lyapunov

Cette méthode, basée sur les champs de vecteur de guidage de Lyapunov (CVL), est présentée par Frew et al.[8]. Prenons la fonction de Lyapunov $V(x, y) = (\|D\|^2 - r^2)^2$, où $\|D\| = \sqrt{x_r^2 + y_r^2}$ est la distance entre le UAV et la cible et r le rayon du cercle. La dérivée de cette fonction par rapport au temps peut être définie non-positive en choisissant les vitesses relatives du UAV $\dot{x}_d$ et $\dot{y}_d$ selon le champ de vecteur de guidage $\mathbf{f}(x_r, y_r)$ donné par :

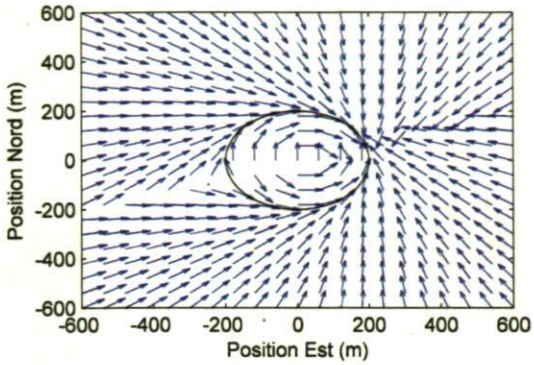


FIGURE 4.3 – Vecteurs d'orientation de l'algorithme ACR pour une orientation du UAV de 20°

$$\mathbf{f}(x_r, y_r) = \begin{bmatrix} \dot{x_d} \\ \dot{y_d} \end{bmatrix} = \frac{-av_0}{\|D\| \left(\|D\|^2 + r^2\right)} \cdot \begin{bmatrix} x_r(\|D\|^2 - r^2) + 2y_r \|D\| r \\ y_r(\|D\|^2 - r^2) - 2x_r \|D\| r \end{bmatrix} \tag{4.9}$$

où a est un facteur de mise à l'échelle et v_0 la vitesse du UAV. La consigne en orientation ψ est calculée à partir de l'équation 4.9 selon :

$$\begin{aligned} \psi &= \text{atan2}\left(\dot{y_d}, \dot{x_d}\right) \\ &= \text{atan2}\left(y_r(\|D\|^2 - r^2) - 2x_r \|D\| r, x_r(\|D\|^2 - r^2) + 2y_r \|D\| r\right) \end{aligned} \tag{4.10}$$

La figure 4.4 montre encore une fois l'orientation en plusieurs points autour du cercle idéal de 200m de rayon.

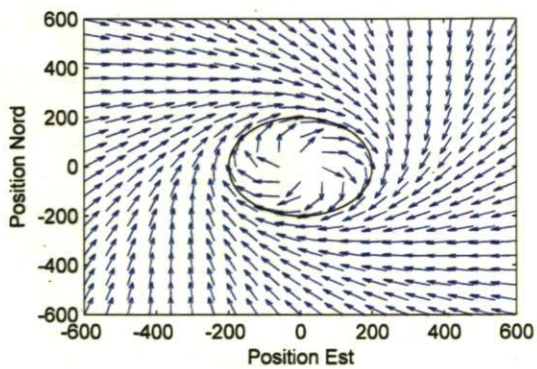


FIGURE 4.4 – Vecteurs d'orientation de l'algorithme CVL

Cible à vitesse constante

Pour une cible à vitesse constante $(\dot{x}_t, \dot{y}_t)$, Frew et al. [8] indiquent qu'il est possible de calculer l'orientation désirée en ajoutant les termes de vitesse de la cible ainsi qu'un facteur d'échelle α_L au calcul de l'orientation pour éviter que le UAV se retrouve à l'intérieur du cercle selon :

$$\psi = \text{atan2}\left(\alpha_L \dot{y}_d + \dot{y}_t, \alpha_L \dot{x}_d + \dot{x}_t\right) \tag{4.11}$$

4.2.4 Comportement du bon timonier

Cette méthode, qui se nomme « Good Helmsman Behavior » en anglais, est proposée par Stoll et al. [9] et représentée à la figure 4.5. Elle est basée sur le fait que l'approche du UAV vers sa trajectoire désirée doit se faire doucement sans la dépasser tout comme un bon timonier. Cette méthode peut s'appliquer pour n'importe quelle trajectoire voulue. Toutefois, dans l'exercice qui nous concerne, puisque le UAV risque d'aller beaucoup plus rapidement que la cible, une trajectoire sous forme de cercle de rayon r autour de la cible a été choisie.

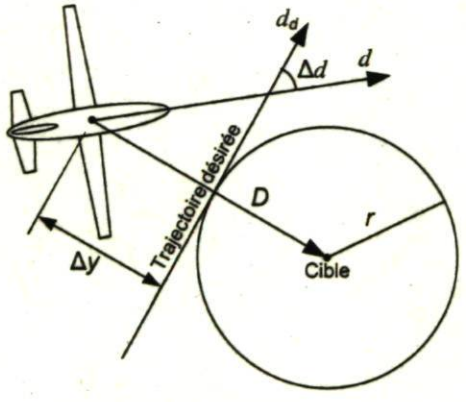


FIGURE 4.5 – Description de la méthode du bon timonier

Cette stratégie peut être réalisée en amenant l'erreur d'angle de trajectoire du UAV $\Delta d = d - d_d$, où d est la trajectoire du UAV et d_d la trajectoire désirée, simultanément à l'écart latéral de trajectoire Δy à 0. Pour ce faire, la consigne en orientation est définie comme $\psi = d_d + d_{\Delta y}$, où $d_{\Delta y}$ est une correction d'orientation fonction de la distance Δy. La trajectoire désirée d_d peut être obtenue en additionnant $\pm 90°$ à la direction du vecteur partant du UAV vers la cible D selon le sens de rotation choisi

(horaire ou anti-horaire). La correction d'orientation $d_{\Delta y}$ est calculée avec une fonction proportionnelle à Δy et saturée à $\pm 45°$. Stoll et al. [9] ont choisi cette limite pour ne pas que les changements d'orientation commandés à l'autopilote soient trop brusques. Si la fonction de saturation est linéaire, elle permet de choisir à partir de quelle distance limite Δy_{lim} la correction maximale de $\pm 45°$ doit être appliquée (figure 4.6). Cette correction est calculée selon :

$$d_{\Delta y} = \frac{\Delta y}{\Delta y_{lim}} \cdot 45° \tag{4.12}$$

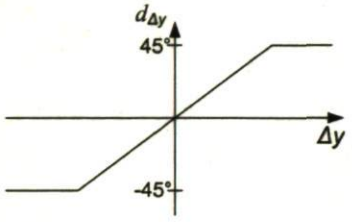


FIGURE 4.6 – Saturation de la correction sur l'orientation due à la distance entre le UAV et la cible

Cette méthode amène le UAV à approcher la trajectoire désirée avec un angle de 45° qui diminue en s'en approchant. Il aussi possible de changer l'agressivité de l'approche du UAV en changeant la distance limite Δy_{lim} où la correction maximale est appliquée.

Pour avoir une meilleure idée de l'algorithme, la figure 4.7 montre encore une fois les consignes en orientation sous forme de vecteurs pour une matrice de points autour d'une cible située en (0,0). Pour cet exemple, le rayon du cercle ainsi que Δy_{lim} ont été fixés à 200 mètres.

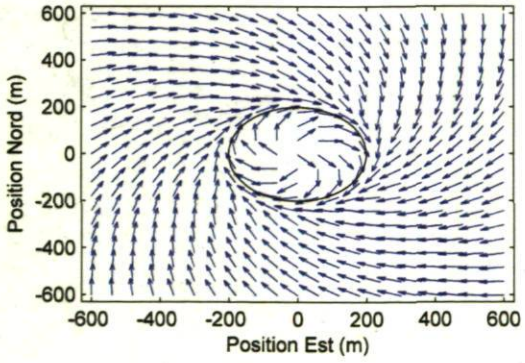


FIGURE 4.7 – Vecteurs d'orientation de l'algorithme du bon timonier

4.2.5 Algorithme de poursuite directe

L'algorithme de contrôle le plus simple est sans contredit de générer des consignes en orientation qui vont diriger le UAV directement vers la cible. Cet algorithme est dénoté APD. En tout point, la consigne en orientation est donnée par :

$$\psi = \text{atan2}(-y_r, -x_r) \tag{4.13}$$

Malgré sa simplicité, cet algorithme a l'avantage d'amener le UAV très près de la cible, ce qui en facilite la surveillance. La figure suivante montre encore une fois les consignes en orientation sous forme de vecteurs pour une matrice de points autour d'une cible située en (0,0).

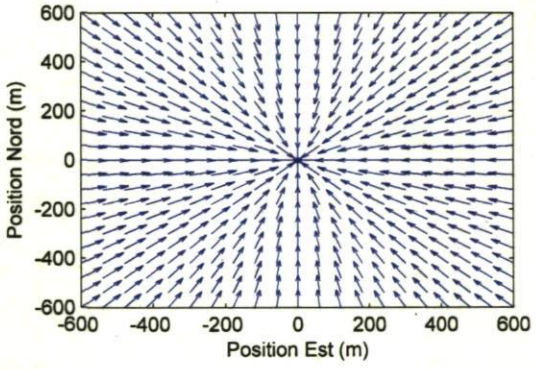


FIGURE 4.8 – Vecteurs d'orientation de l'algorithme de poursuite directe

4.3 Conclusion

Ce chapitre a présenté 5 algorithmes de base pour la surveillance de cible. Les 4 premiers utilisent différentes techniques amenant le UAV à tourner autour d'une cible suivant un cercle de rayon fixe. Ces techniques s'avèrent très efficaces lorsque la cible se déplace à basse vitesse. Elles ont aussi l'avantage de garder la distance constante entre le UAV et la cible. Enfin, une autre technique est présentée où le UAV est dirigé à tout moment directement vers la cible. Cette méthode peut occasionner des problèmes lorsque le UAV passe au dessus de la cible si les mouvements de la caméra sont limités. Toutefois, malgré que la distance maximale entre le UAV et la cible risque d'être plus grande avec cette méthode, elle a l'avantage de faire passer le UAV très près de la cible, ce qui peut en faciliter l'observation.

Chapitre 5

Simulation des algorithmes de surveillance de base

Ce chapitre porte sur la validation, l'analyse et la comparaison des méthodes de surveillance présentées au chapitre précédent à l'aide d'une cible à vitesse constante. Ces tests se déroulent sur le système HIL présenté au chapitre 3. La section 5.1 présente le scénario de simulation ainsi que la description des graphiques de résultats. Ensuite, les sections 5.2 et 5.3 présentent les résultats de simulation. Enfin, les résultats sont commentés et analysés à la section 5.4. Il est à noter que ces résultats ont déjà été présentés en partie par Thériault et al. [39].

5.1 Description du scénario de simulation

Pour qu'il soit possible de comparer les divers algorithmes, les mêmes simulations ont étés faites pour chacun d'eux. La cible a effectué le même parcours pour chaque algorithme et ceux ci ont été utilisés avec les mêmes paramètres lorsque possible. En effet, pour les quatre algorithmes qui forment des cercles autours de la cibles, il ont tous été testés avec le même sens de rotation et le rayon a été fixé à 175 mètres. Cette valeur a été choisie afin de correspondre au rayon de braquage minimum du UAV qui est aussi d'environ 175 mètres. Pour tous les tests, la vitesse du UAV a été fixée à 100 km/h et son altitude à 200 mètres puisqu'aucun des algorithmes ne contrôlent ces paramètres.

Modélisation simplifiée du UAV

Afin de s'assurer de la bonne implantation des algorithmes, des simulations utilisant un modèle de UAV très simplifié modélisant seulement la commande en orientation ont été conduites sous Matlab. Les résultats de ces simulations ont été superposés à celles utilisant le montage HIL pour qu'il soit possible de les comparer facilement. Le modèle de premier ordre G_o entre la commande en orientation envoyée à l'autopilote et l'orientation du UAV a été identifié à l'aide de tests de réponse à l'échelon. Il est défini par :

$$G_o = \frac{1}{(1 + 3.78s)} \tag{5.1}$$

Il est à noter qu'un limiteur de pente de $\pm 10°$ par seconde a été ajouté au modèle afin de refléter le comportement réel du UAV. On peut voir l'ajustement du modèle pour des données d'identification à la figure 5.1.

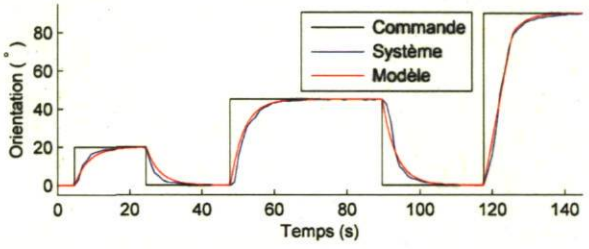


FIGURE 5.1 – Identification de l'orientation du UAV

Ce modèle est utilisé de façon similaire aux simulations utilisant le montage HIL pour avoir des résultats le plus identiques possible. En effet, une nouvelle commande en orientation lui est envoyée une fois par seconde tout comme pour les simulations avec le montage. Cette commande est toutefois calculée en utilisant la position réelle de la cible au lieu de celle estimée par le système vidéo de poursuite de cible afin de connaître le comportement idéal de l'algorithme. Malgré les changements de commande s'effectuant seulement une fois par seconde, le modèle en orientation ainsi qu'en position est mis à jour à une fréquence de 10 Hz pour adoucir l'affichage. La mise à jour du modèle en position se fait à l'aide des équations de mouvement données par :

$$N(k+1) = N(k) + cos(\psi)\Delta t v_p \tag{5.2}$$

$$E(k+1) = E(k) + sin(\psi)\Delta t v_p \tag{5.3}$$

où N et E sont respectivement les positions Nord et Est du UAV, Δt est la période de rafraichissement, ψ est l'orientation du UAV et v_p la vitesse du UAV à chaque instant.

Description des graphiques de résultats

Afin de décrire le contenu des graphiques de résultats, prenons la figure 5.2 en exemple. On peut voir en rouge la position du UAV et en bleu celle de la cible. En vert est représentée la position estimée de la cible à l'aide du système vidéo de poursuite de cible. Cette position est utilisée par les différents algorithmes de surveillance dans leurs calculs afin de trouver la nouvelle commande en orientation à envoyer à l'autopilote. Enfin, en gris est représentée la position du modèle de UAV simulé sous Matlab.

Aux quatre couleurs précédemment définies s'ajoutent deux autres. Tout d'abord, lorsque le système perd la cible du champ de vue de la caméra, la couleur de la courbe représentant la position du UAV passe du rouge au cyan. Dans ce cas, la poursuite a dû être arrêtée et ensuite repartie lorsque la cible à de nouveau été retrouvée de façon manuelle. On peut en voir l'effet sur toutes les simulations avec l'algorithme simple où la cible se déplace à vitesse non nulle. En outre, pour qu'il soit plus facile de connaître la correspondance pour un temps donné entre la position du UAV et celle de la cible, une ligne noir les relies à chaque 12 secondes de simulation.

5.2 Simulations avec une cible à orientation constante

Les tests présentés aux figures 5.2, 5.3, 5.4 et 5.5 ont été faits pour une cible se déplaçant vers l'est à 4 vitesses différentes : 0, 6.94, 13.89 et 20.83 m/s. Ces vitesses correspondent à 0, 25, 50 et 75 % de la vitesse du UAV qui est de 27.78 m/s ou 100 km/h. Afin d'alléger les figures, l'algorithme présenté à la section 4.2 utilisant les bifurcations de Hopf super-critiques est noté BHSC. Celui présenté par Rafi et al. est noté ACR. Celui utilisant les champs de vecteur de Lyapunov est noté CVL. Celui simulant le comportement du bon timonier est noté CBT. Enfin, l'algorithme de poursuite directe est noté APD.

Pour que les résultats soient comparables, le temps de simulation n'a pas été fixé mais plutôt choisi en fonction du patron dessiné par le parcours du UAV. Par exemple, pour les simulations représentées à la figure 5.2, le temps n'est pas le même, mais correspond à cinq tours complets du UAV autour de la cible. Il est à noter que cette section ne présente que les graphiques de résultats pour une cible à orientation constante. L'analyse est faite à la section 5.4.

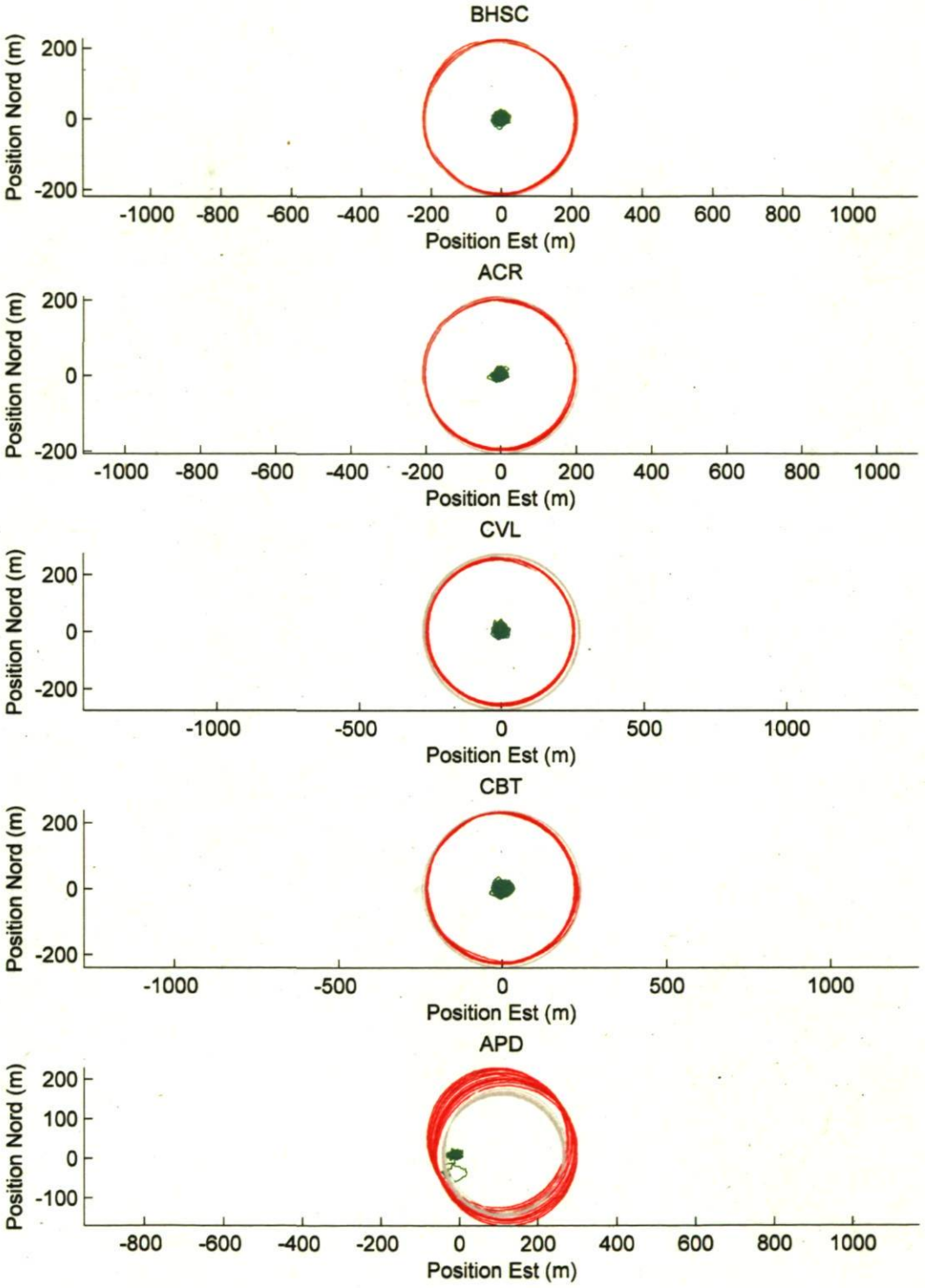


FIGURE 5.2 – Simulations des algorithmes de base avec une cible fixe : UAV –, Cible – ,Cible estimée –, UAV simulé –

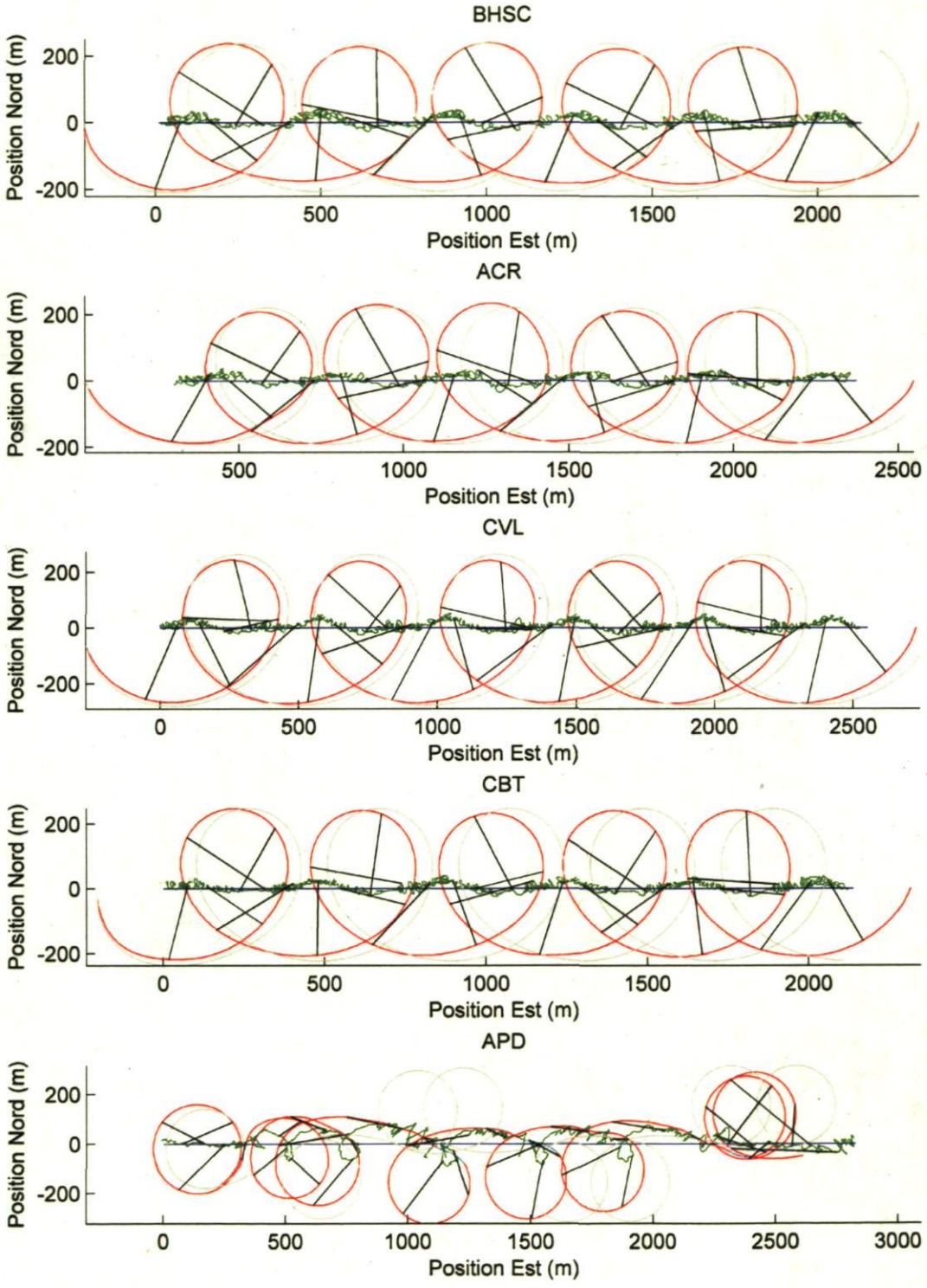


FIGURE 5.3 – Simulations des algorithmes de base avec une cible à orientation constante se déplaçant à 6.94 m/s : UAV –, Cible – ,Cible estimée –, UAV simulé –

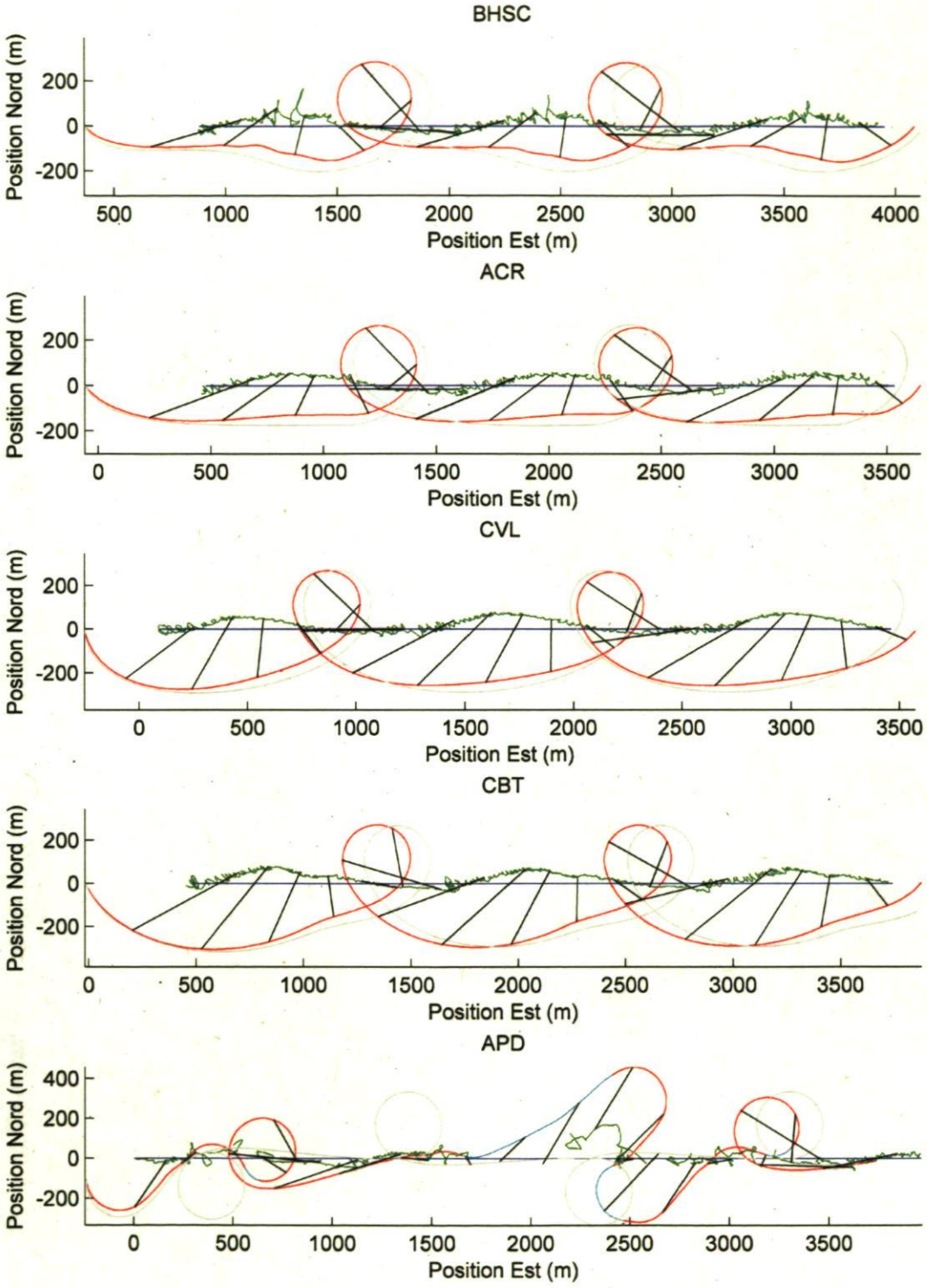


FIGURE 5.4 – Simulations des algorithmes de base avec une cible à orientation constante se déplaçant à 13.89 m/s : UAV –, Cible – ,Cible estimée –, UAV simulé –

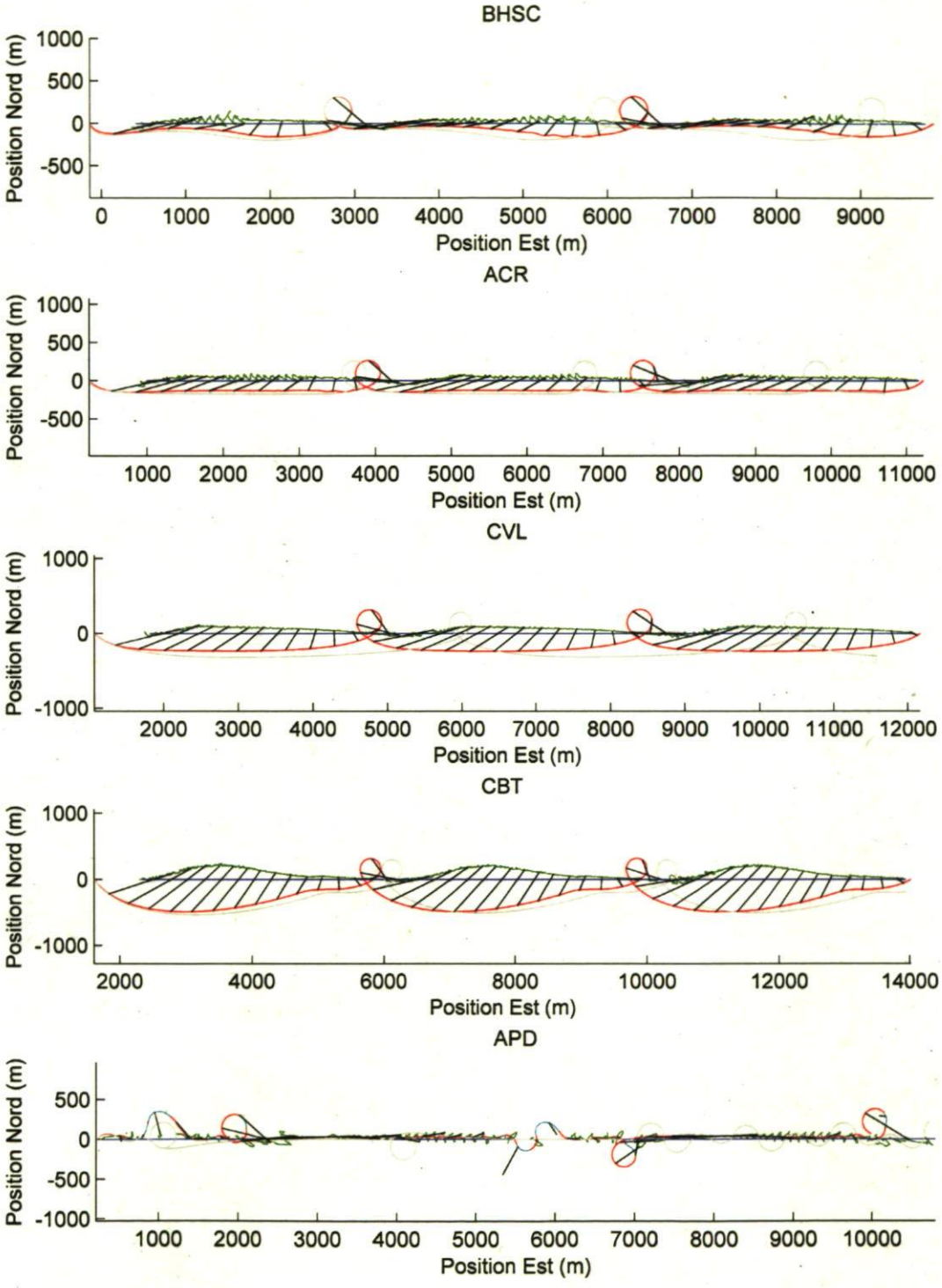


FIGURE 5.5 – Simulations des algorithmes de base avec une cible à orientation constante se déplaçant à 20.83 m/s : UAV –, Cible – ,Cible estimée –, UAV simulé –

5.3 Simulations avec une cible effectuant des virages à 90 degrés

Les tests présentés aux figures 5.6, 5.7 et 5.8 ont été faits pour une cible effectuant des virages à 90° à 3 vitesses différentes : 6.94, 13.89 et 20.83 m/s. Ces vitesses correspondent à 25, 50 et 75% de la vitesse du UAV qui est de 27.78 m/s ou 100 km/h.

La cible se déplace en ligne droite et effectue 4 virages afin de former un patron en 'S'. Le temps en ligne droite diffère entre les différentes vitesses de cible pour que le UAV ait le temps d'effectuer environ le même nombre de passage autour de la cible. Les temps choisis ont été de 54, 84 et 157 secondes respectivement pour les simulations à 6.94, 13.89 et 20.83 m/s. Il est à noter que cette section ne présente que les graphiques de résultats pour une cible effectuant des virages. L'analyse est faite à la section 5.4. De plus, puisque que l'algorithme de poursuite directe a perdu la cible de son champ de vue à plusieurs reprises pour une cible se déplaçant seulement en ligne droite, les tests de cette section n'ont pas été faits pour cet algorithme.

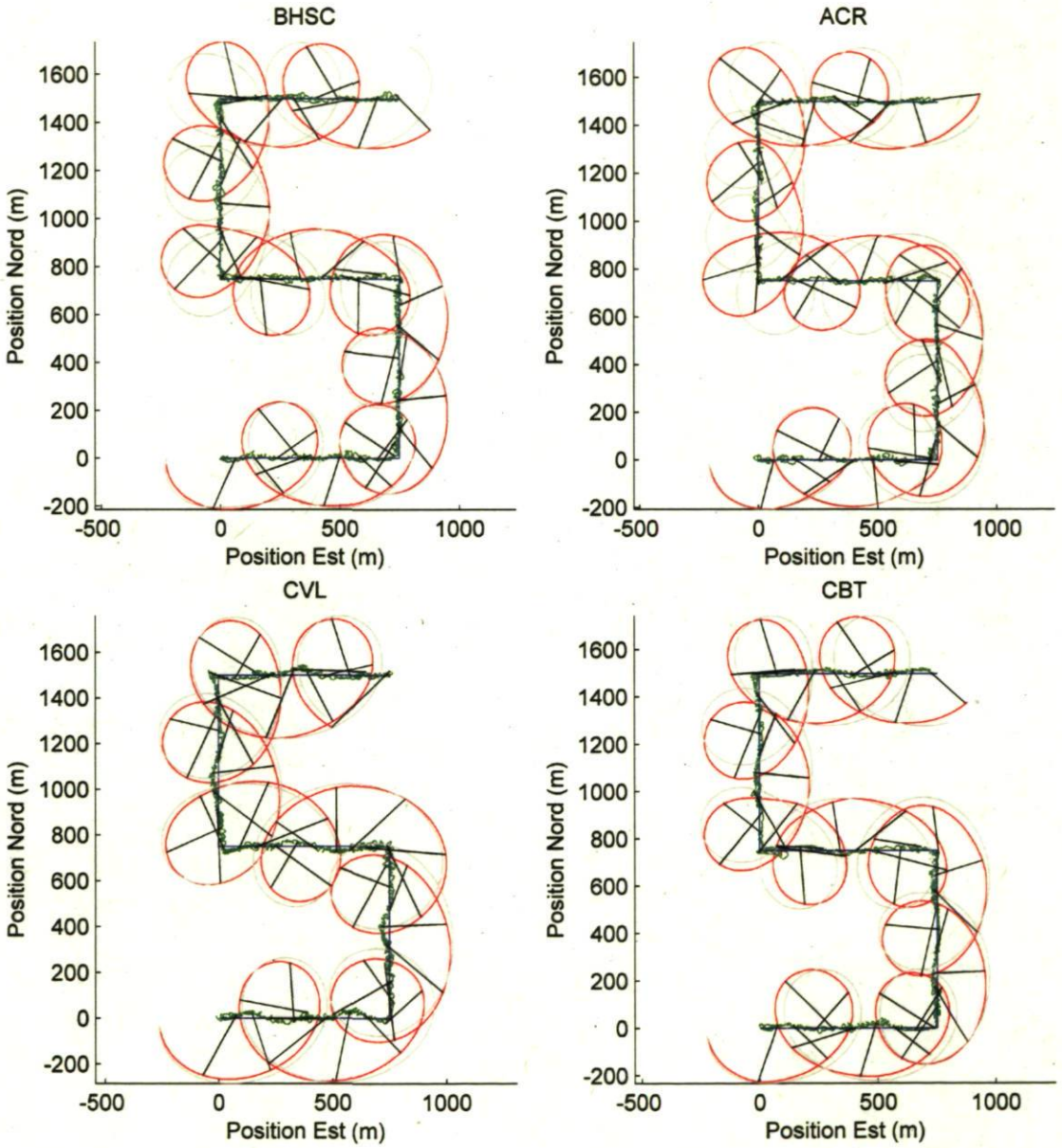


FIGURE 5.6 – Simulations des algorithmes de base avec une cible se déplaçant à 6.94 m/s et effectuant des virages à 90° : UAV –, Cible – ,Cible estimée –, UAV simulé –

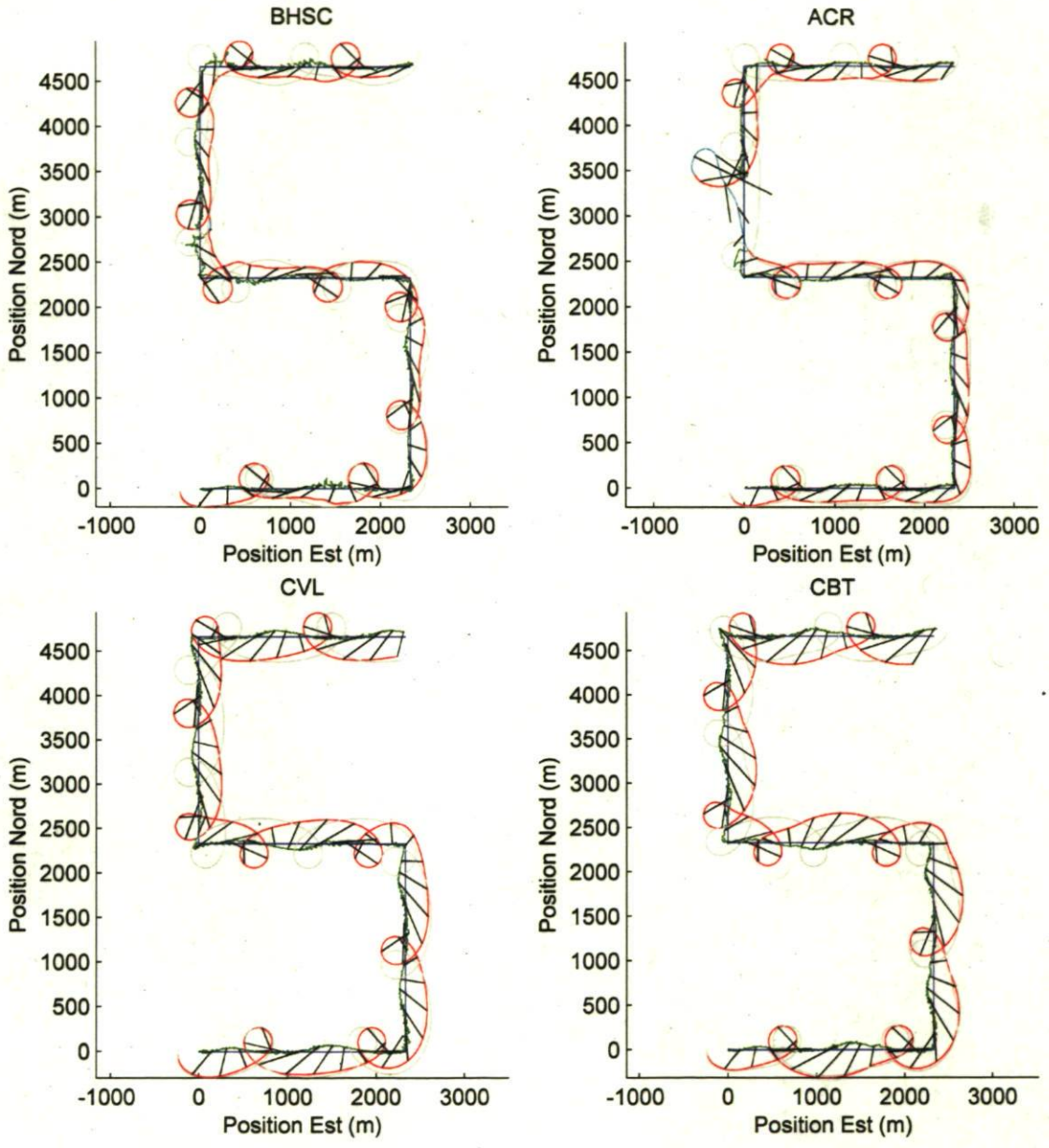


FIGURE 5.7 – Simulations des algorithmes de base avec une cible se déplaçant à 13.89 m/s et effectuant des virages à 90° : UAV –, Cible – ,Cible estimée –, UAV simulé –

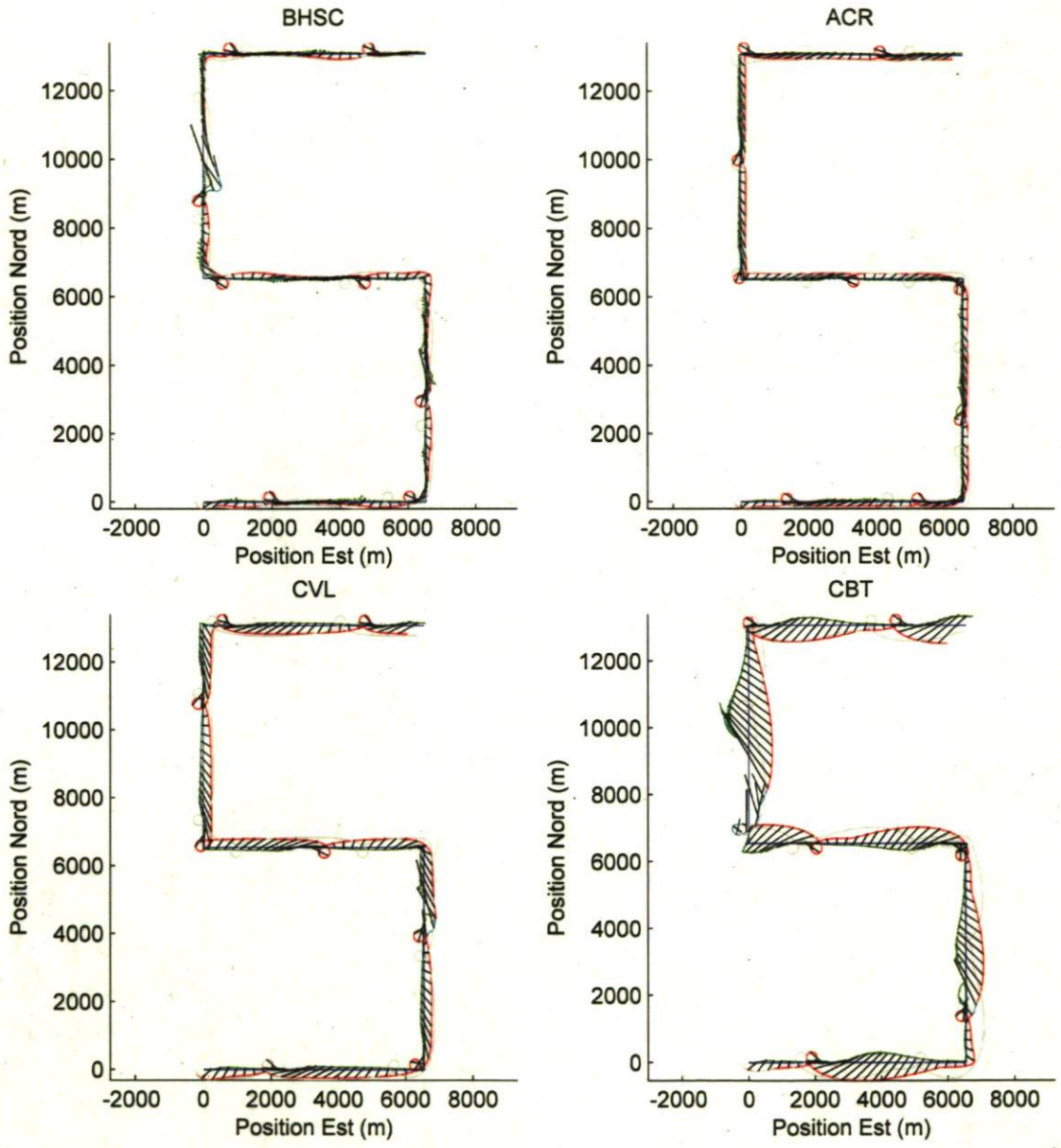


FIGURE 5.8 – Simulations des algorithmes de base avec une cible se déplaçant à 20.83 m/s et effectuant des virages à 90° : UAV –, Cible – ,Cible estimée –, UAV simulé –

5.4 Analyse des résultats

Tout d'abord, il est possible de se rendre compte que la trace du modèle de UAV simulé en gris n'est pas exactement la même que celle en rouge du UAV utilisé avec le système HIL. Différentes raisons peuvent expliquer ce phénomène. Tout d'abord, comme il est mentionné dans la section précédente, la vraie position de la cible est utilisée pour le UAV simulé sous Matlab au lieu de la position estimée à l'aide du système vidéo. Cette pratique provoque quelques différences mineures au niveau de la direction du UAV. Un autre facteur d'erreur est la vitesse du UAV. Malgré que la vitesse demandée du UAV est constante à 100 km/h, il y a toujours une faible variation. La vitesse utilisée pour la simulation est retournée par l'autopilote seulement une fois par seconde avec une précision de $\pm$ 2 km/h. Ce manque de précision sur la vitesse peut entraîner une différence de position non négligeable pour de longues simulations. Le modèle d'orientation utilisé par le UAV simplifié engendre lui aussi une certaine erreur. Il est approximé par une simple fonction de transfert de premier ordre représentée par l'équation 5.1 avec un limiteur de pente. Cette approximation est près de la réalité sans toutefois être parfaite.

Malgré les différences entre les résultats des simulations et des tests avec le montage HIL, les simulations permettent tout de même de voir que le comportement du UAV lors des tests HIL est bien semblable aux algorithmes simulés en situation contrôlée. Plusieurs données ont été enregistrées lors des simulations pour permettre une bonne comparaison des différents algorithmes. Les principales données étudiées sont la position du UAV, de la cible ainsi que la position estimée de la cible. Ces informations permettent d'analyser la proximité du UAV à la cible surveillée, ce qui est un élément majeur en surveillance.

Les tableaux 5.1 et 5.2 présentent différentes données tirées des simulations avec le système HIL. Pour chaque algorithme et chaque vitesse de cible, on y retrouve la moyenne, l'écart type, le minimum ainsi que le maximum de la distance entre le UAV et la cible sur le plan horizontal. On y retrouve aussi la moyenne ainsi que l'écart type de l'erreur d'estimation sur la position de la cible. Enfin, la dernière colonne indique si le système vidéo de poursuite de cible a perdu la cible durant la simulation : 0 signifiant un bon déroulement de simulation et 1 une perte de poursuite.

Tableau 5.1 – Simulations avec une cible à orientation constante

Algorithme	Vitesse de la cible (m/s)	Distance moyenne UAV-Cible (m)	Écart type distance UAV-Cible (m)	Distance minimum UAV-Cible (m)	Distance maximum UAV-Cible (m)	Erreur d'estimation moyenne (m)	Écart type erreur d'estimation (m)	Perte de la poursuite
BHSC	0	216	5	206	231	13	6	0
BHSC	6.94	225	50	162	332	20	9	0
BHSC	13.88	276	134	125	522	42	18	0
BHSC	20.83	348	195	82	737	54	26	0
ACR	0	200	5	189	212	11	5	0
ACR	6.94	218	50	147	324	19	9	0
ACR	13.88	290	145	113	538	43	14	0
ACR	20.83	403	223	45	789	61	36	0
CVL	0	257	4	244	267	14	7	0
CVL	6.94	263	58	171	350	23	11	0
CVL	13.88	313	133	108	517	47	20	0
CVL	20.83	430	220	72	786	88	46	0
CBT	0	228	5	214	239	14	7	0
CBT	6.94	230	42	171	310	20	8	0
CBT	13.88	323	139	117	512	45	20	0
CBT	20.83	496	254	81	804	131	93	0
APD	0	196	79	58	302	14	6	0
APD	6.94	231	114	1	478	52	41	1
APD	13.88	277	167	1	590	44	48	1
APD	20.83	322	247	0	825	61	60	1

Tableau 5.2 – Simulations avec une cible effectuant des virages à 90°

Algorithme	Vitesse de la cible (m/s)	Distance moyenne UAV-Cible (m)	Écart type distance UAV-Cible (m)	Distance minimum UAV-Cible (m)	Distance maximum UAV-Cible (m)	Erreur d'estimation moyenne (m)	Écart type erreur d'estimation (m)	Perte de la poursuite
BHSC	6.94	228	38	168	308	15	8	0
BHSC	13.88	274	127	60	524	36	19	1
BHSC	20.83	389	236	68	1091	55	52	1
ACR	6.94	225	49	156	345	15	8	0
ACR	13.88	288	140	73	669	31	12	1
ACR	20.83	392	207	31	777	53	32	1
CVL	6.94	271	55	174	351	20	10	0
CVL	13.88	328	135	104	530	46	23	0
CVL	20.83	440	218	30	857	86	51	1
CBT	6.94	232	41	167	317	16	7	0
CBT	13.88	336	145	94	529	50	29	0
CBT	20.83	551	305	49	1167	191	197	1

5.4.1 Perte de poursuite

Le facteur le plus déterminant lors de comparaison d'algorithmes est sans contredit la perte de poursuite. En effet, lorsque le système perd de vue la cible, il n'est plus possible de déterminer sa position, le contrôle automatique s'arrête et une intervention humaine est nécessaire pour la retrouver à l'aide de la manette.

Pour ce qui est des simulations avec orientation constante, l'algorithme APD perd la cible de son champ de vue pour tous les essais sauf pour celui où la cible est arrêtée. Ces résultats s'expliquent parce que cet algorithme amène le UAV à passer directement au dessus de la cible et que la caméra est limitée à un mouvement panoramique horizontal de $\pm 160°$. La caméra ne peut être orientée vers l'arrière ou faire de tour sur elle même. Il est donc possible de rejeter cet algorithme puisqu'il ne répond pas aux contraintes imposées par la caméra du montage. Pour cette raison, les simulations avec changements de direction n'ont pas été faites avec l'algorithme APD.

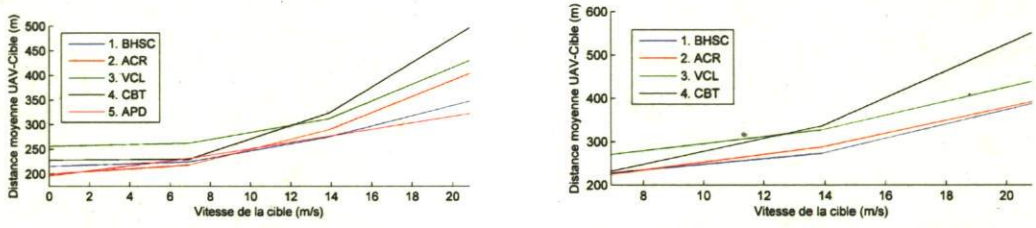
Pour ces secondes simulations, il est plus difficile de tirer une conclusion en analysant la perte de poursuite. En effet, tous les algorithmes ont perdu de vue la cible à 20.83 m/s et deux d'entre eux l'ont perdue à 13.89 m/s (BHSC et ACR). Pour ces simulations, il semble que les perte de poursuite aient été provoquées par la distance trop importante qui séparait le UAV de la cible ou par malchance. En effet, si une cible se déplaçant rapidement effectue un changement brusque de direction qui amène sa trajectoire directement vers le UAV sans que celui-ci ait le temps de l'éviter, il y a un gros risque de perte de poursuite. On peut en voir l'exemple à la figure 5.7 pour l'algorithme ACR. Comme cette situation aurait pu arriver pour tous les algorithmes, ce n'est pas une mesure suffisante pour le rejeter. Par conséquent, d'autres données ont dû être analysées afin de comparer les différentes méthodes.

5.4.2 Distance moyenne

Le graphique présenté à la figure 5.9a présente la distance moyenne entre le UAV et la cible pour chacun des algorithmes pour les 4 différentes vitesses de cible en ligne droite. Cette mesure est très importante puisqu'en surveillance, il importe d'être près de la cible afin de pouvoir analyser son comportement. Tout d'abord, on peut remarquer que l'algorithme APD semble donner de très bons résultats, mais suite à l'analyse faite à la section 5.4.1, cet algorithme doit être rejeté. On peut aussi remarquer que les algorithmes CVL et CBT amènent une plus grande distance que les autres. En effet, ces algorithmes amènent le UAV vers sa trajectoire désirée de façon douce avec de faibles

variations de commande. Les algorithmes BHSC et ACR semblent assez équivalent du point de vue de la distance moyenne. On peut voir que le premier est meilleur pour une cible à haute vitesse tandis que le second domine pour une cible à basse vitesse. La performance à haute vitesse du premier algorithme s'explique par le fait que lorsque la distance séparant le UAV de la cible est assez grande, le UAV s'oriente non pas sur le cercle, mais bien sur la position de la cible. On peut en voir l'effet en comparant la trajectoire des UAVs pour ces deux algorithmes à la figure 5.5.

La figure 5.9b montre cette fois les résultats pour les simulations avec changements de direction. Ces résultats appuient les explication données précédemment puisqu'ils sont presque identiques.

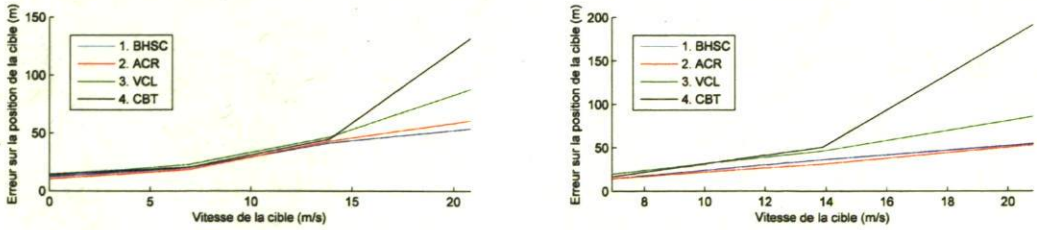


(a) Orientation fixe (b) Changements d'orientation

FIGURE 5.9 – Distance moyenne entre le UAV et la cible

5.4.3 Erreur moyenne sur la position estimée de la cible

Dans un contexte de surveillance, il peut être intéressant d'étudier la position estimée de la cible. En effet, une bonne précision sur cette position peut devenir nécessaire s'il faut l'engager en cas de menace. La figure 5.10 nous montre l'erreur sur cette position estimée.



(a) Orientation fixe (b) Changements d'orientation

FIGURE 5.10 – Erreur sur la position estimée de la cible

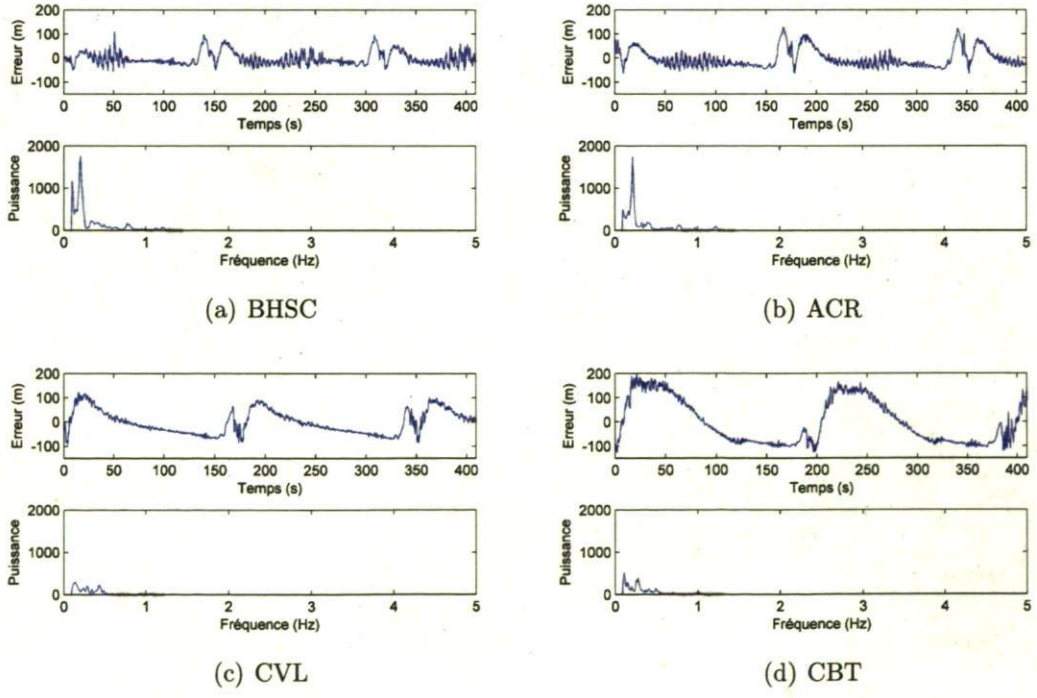
Après analyse des figures, on remarque une très grande corrélation entre ces résultats et ceux de la section 5.4.2. En effet, plus le UAV est loin de la cible, plus sa position

sera difficile à estimer. Cette différence de position est bien visible sur les simulations de la figure 5.5.

5.4.4 Fréquence et amplitude des oscillations sur la position estimée de la cible

Suite à l'observation des résultats de simulation, on peut se rendre compte que les algorithmes amènent différentes oscillations sur la position estimée de la cible. Ces oscillations sont encore plus visibles sur le graphique de la figure 5.5. Ces oscillations peuvent paraître anodines, mais elles entraînent des oscillations dans le contrôle de la caméra pouvant provoquer une perte de la poursuite.

La figure 5.11 présente la densité spectrale de puissance pour les 4 simulations de la figure 5.5. Il est possible de remarquer que les algorithmes BHSC et ACR produisent beaucoup plus de vibrations que les 2 autres. Les oscillations les plus importantes se situent à environ 0.2 Hz. Elles sont produites par les variations fréquentes du roulis du UAV suite aux changements de commande en orientation.



(a) BHSC (b) ACR

(c) CVL (d) CBT

FIGURE 5.11 – Analyse des oscillations par la densité spectrale de puissance

5.5 Conclusion

Suite aux différentes analyses, on remarque que les algorithmes BHSC et ACR donnent des résultats très similaires. Il est possible de faire le même constat pour les algorithmes VCL et CBT. Les premiers réussissent à rester plus près de la cible et produisent une meilleure précision de sa position. Toutefois, ces avantages provoquent une plus grande oscillation au niveau du contrôle de la caméra, ce qui peut entraîner une perte de la poursuite.

Les algorithmes BHSC et ACR seraient donc plus utiles dans les situations où la connaissance de la position de la cible est importante. Dans ce cas, l'algorithme BHSC semble préférable pour les cibles à haute vitesse et le deuxième pour celles à basse vitesse. Par ailleurs, si des simulations plus complexes avaient été testées, l'algorithme ACR aurait probablement pu tirer avantage du fait que son sens de rotation autour de la cible n'est pas fixe, mais plutôt choisi en fonction de la position de la cible.

Les algorithmes VCL et CBT ont quant à eux l'avantage de produire une moins grande oscillation au niveau de la position de la cible entraînant une plus grande stabilité de la caméra et ainsi moins de chance de perdre la poursuite. Ces algorithmes seraient donc plus utiles lorsque la poursuite est critique et où on ne veut pas devoir prendre le contrôle de la caméra en mode manuel. L'algorithme VCL semble par ailleurs préférable à CBT puisqu'il engendre une plus petite distance moyenne à la cible ainsi qu'une meilleure précision sur sa position.

Chapitre 6

Algorithmes avancés pour la surveillance de cible

Ce chapitre traite de deux algorithmes avancés pour la surveillance de cible. Ces algorithmes sont décrits dans un chapitre distinct puisqu'ils diffèrent grandement des autres dans leur utilisation ainsi que dans leur complexité. Le premier algorithme amène le UAV à parcourir une trajectoire sinusoïdale à une distance constante derrière la cible. Il semble très efficace pour les cibles se déplaçant à grande vitesse. Le second algorithme tire profit de la commande prédictive en estimant la position future du UAV ainsi que celle de la cible pour en assurer une surveillance efficace.

6.1 Algorithme de surveillance à trajectoire sinusoïdale

Avant d'implanter cet algorithme sur le système HIL, il a tout d'abord été testé en simulation Matlab. Ces simulations ont permis de constater que l'algorithme devait subir quelques modifications avant de pourvoir être testé sur le montage HIL. Cette section débute en décrivant la méthode originale et en la simulant sous Matlab. Ensuite, deux correctifs améliorant l'algorithme sont présentés pour qu'il soit possible de l'utiliser avec le montage HIL.

6.1.1 Description de la méthode originale

Cet algorithme a été développé par Lee et al. [6]. Contrairement aux algorithmes préalablement étudiés, celui-ci a été développé pour des cibles se déplaçant à des vitesses supérieures au tier de la vitesse du UAV. Le ratio de vitesse est alors $\sigma = v_p/v_t < 3$, où v_p est la vitesse du UAV et v_t la vitesse de la cible. Dans le cas contraire, Lee et al. [6] suggèrent de retourner à un des algorithmes précédents tel que faire des passages successifs au dessus de la cible (méthode APD) ou tourner en rond autour de celle-ci (méthodes BHSC, ACR, CVL ou CBT).

La méthode est présentée à la figure 6.1. Sur celle-ci, on peut voir le UAV représenté par un avion et la cible par un point noir. Les distance D_{s1} et D_{s2} sont égales et définissent la distance horizontale parcourue par les deux entités durant la période de temps T_s qui est arbitrairement choisie.

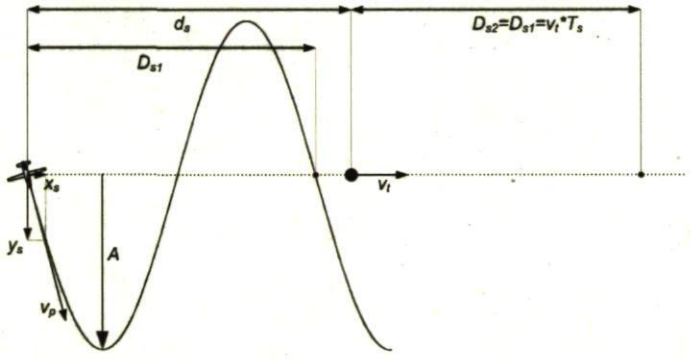


FIGURE 6.1 – Principe de la méthode à trajectoire sinusoïdale

Ce qu'il faut trouver à l'intérieur de cette figure est l'amplitude du sinus nécessaire pour que la distance parcourue par la cible en un temps T_s soit la même que celle parcourue par le UAV dans le même temps. Il est possible de calculer cette amplitude analytiquement à l'aide des relations 6.1 à 6.8 avec $D_s = D_{s1} = D_{s2}$. Tout d'abord, le sinus que fait le UAV peut être représenté comme :

$$y_s = A \sin\left(\frac{2\pi x_s}{D_s}\right) \tag{6.1}$$

Une fois dérivée, l'équation devient :

$$\dot{y_s} = A_m \cos\left(\frac{2\pi x_s}{D_s}\right) \dot{x_s} \tag{6.2}$$

où $A_m = 2\pi A/D_s$. On sait que l'amplitude de la vitesse du UAV v_p est reliée aux

composantes x et y selon :

$$v_p^2 = \dot{x_s}^2 + \dot{y_s}^2 \tag{6.3}$$

Substituer l'équation 6.2 dans 6.3 donne :

$$v_p^2 = \dot{x_s}^2 + A_m^2 \cos^2\left(\frac{2\pi x_s}{D_s}\right)\dot{x_s}^2 \tag{6.4}$$

Après quelques manipulations pour isoler $\dot{x_s}$, on trouve :

$$\dot{x_s} = \frac{v_p}{\sqrt{1 + A_m^2 \cos^2\left(\frac{2\pi x_s}{D_s}\right)}} \tag{6.5}$$

En notant que $\dot{x_s} = dx_s/dt$, on peut transformer l'équation pour mettre en évidence les intégrales :

$$\int_0^{T_s} dt = \frac{1}{v_p}\int_0^{D_s} \sqrt{1 + A_m^2 \cos^2\left(\frac{2\pi x_s}{D_s}\right)}\, dx_s \tag{6.6}$$

On sait que $\int_0^{T_s} dt = T_s$ et $T_s = D_s/v_t$, alors :

$$\frac{D_s}{v_t} = \frac{1}{v_p}\int_0^{D_s} \sqrt{1 + A_m^2 \cos^2\left(\frac{2\pi x_s}{D_s}\right)}\, dx_s \tag{6.7}$$

et puisque le ratio $\sigma = v_p/v_t$,

$$\sigma = \frac{1}{D_s}\int_0^{D_s} \sqrt{1 + A_m^2 \cos^2\left(\frac{2\pi x_s}{D_s}\right)}\, dx_s \tag{6.8}$$

Cette intégrale pourrait être résolue pour ensuite isoler A, mais une intégrale elliptique de deuxième espèce doit être résolue. Ces intégrales, nées de la volonté de calculer la longueur d'un arc d'ellipse, ne sont pas résolubles pas les moyens habituels. Par conséquent, une autre approche a été amenée pour être en mesure de retrouver A en fonction de σ. La démarche est basée sur le fait que le ratio de la longueur du sinus parcouru par le UAV sur la longueur D_s de la ligne droite parcourue par la cible est le même que le ratio des vitesses $\sigma = v_p/v_t$. Il est alors possible trouver ce ratio en calculant la longueur du sinus pour $D_s = 1$, au lieu de d'avoir à calculer l'intégrale. Afin de calculer la longueur d'une période de sinus, il est possible d'utiliser une sommation pour avoir une bonne approximation. Cette sommation peut être représentée, en utilisant l'équation 6.1, par :

$$\sigma = v_p/v_t = \sum_{i=0}^{nb-1} \sqrt{\left(y_s\left(\frac{i+1}{nb}\right) - y_s\left(\frac{i}{nb}\right)\right)^2 + \left(\frac{1}{nb}\right)^2} \tag{6.9}$$

où nb est le nombre de sous division utilisées pour le calcul de la sommation. Plus ce nombre est grand, plus le résultat sera précis.

La figure 6.2 montre un exemple du calcul de la longueur pour $nb = 5$ et $A = 1$. On peut voir sur le graphique une itération de la sommation pour $i = 1$.

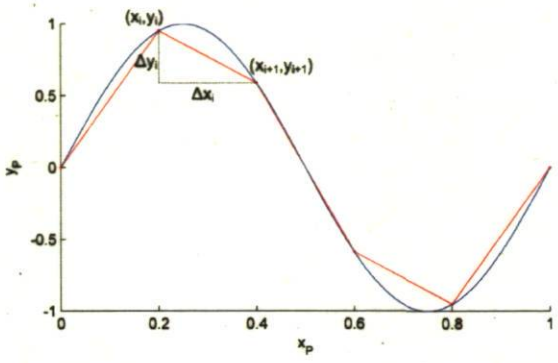


FIGURE 6.2 – Calcul de la longueur du sinus

La figure 6.3 montre σ en fonction du ratio A/D_s trouvé à partir de l'équation 6.9. À partir de cette courbe, il est possible d'en inverser les données pour avoir la courbe du ratio A/D_s en fonction du ratio de vitesse σ. Cette relation représentée à la figure 6.4 permet ensuite de trouver une équation de régression.

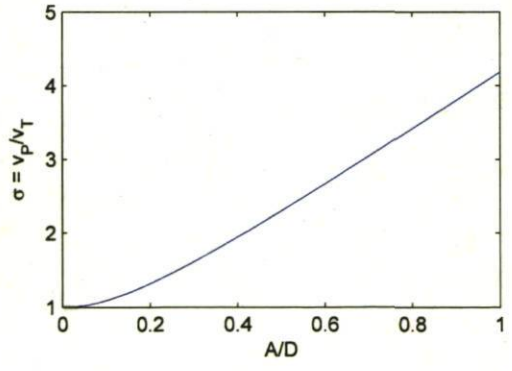


FIGURE 6.3 – σ en fonction de l'amplitude du sinus

Puisque le début de la courbe diffère grandement de la fin, deux équations de régression ont été trouvées afin de bien représenter les données. Soit une de $\sigma = 1$ à 1.09 et une autre pour le reste. Ces deux courbes sont données dans les deux équations suivantes représentées à la figure 6.5.

$$A/D_s = -4320.8\sigma^4 + 18212\sigma^3 - 28783\sigma^2 + 20217\sigma - 5324.7; \sigma = [1 - 1.09] \quad (6.10)$$

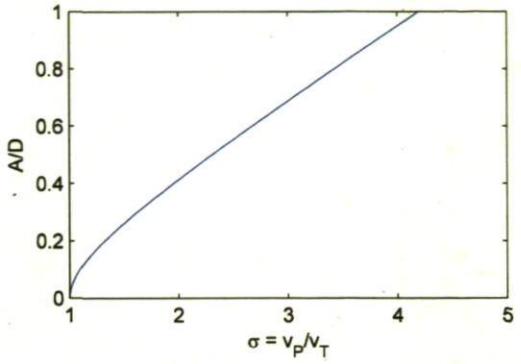
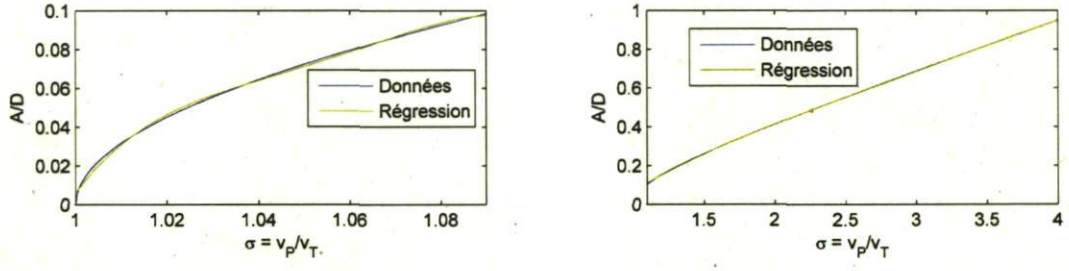


FIGURE 6.4 – Amplitude du sinus en fonction de σ

$$A/D_s = -0.0055183\sigma^4 + 0.066639\sigma^3 - 0.30003\sigma^2 + 0.86452\sigma - 0.55806; \sigma = [1.09 - 4] \tag{6.11}$$



(a) $\sigma = 1 - 1.09$ (b) $\sigma = 1.09 - 4$

FIGURE 6.5 – Courbes de régression représentant le ratio A/D en fonction du ratio de vitesse σ

Pour s'assurer que ces courbes représentent bien la réalité, l'erreur d'approximation est tracée à la figure 6.6. On peut voir que l'erreur n'excède jamais 0.01, ce qui est amplement suffisant pour les besoins de cet algorithme.

Maintenant qu'il est possible de connaître l'amplitude du sinus à accomplir en fonction du ratio des vitesses, l'orientation peut être trouvée à l'aide des relations 6.2 et 6.5 ainsi que par :

$$\psi = arctan\left(\frac{\dot{y}_s}{\dot{x}_s}\right) \tag{6.12}$$

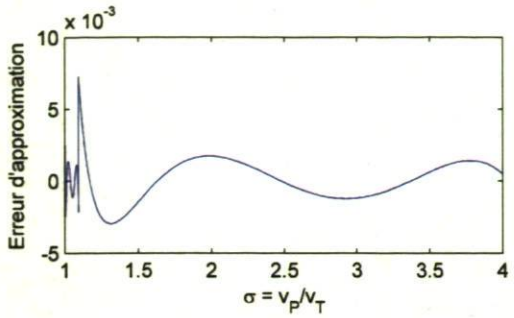


FIGURE 6.6 – Erreur amenée par les courbes de régression

6.1.2 Simulations de la méthode originale

Malgré sa complexité, son utilisation semble tout de même simple puisqu'il n'y a qu'un seul paramètre. Dépendamment du ratio de vitesse $\sigma = v_p/v_t$, le ratio A/D_s est donné. Il faut donc fixer la distance D_s pour enfin connaître l'amplitude du sinus à parcourir. Puisque $A = \sigma D_s$, plus cette distance sera grande, plus l'amplitude du sinus le sera aussi. Il sera alors plus facile pour le UAV d'effectuer les virages, mais cette grande amplitude éloignera le UAV de la cible et compromettra ainsi une bonne surveillance. Il faut donc choisir une distance D_s la plus petite possible. Sachant que le UAV a un rayon de braquage minimum de 175m et qu'il faut qu'il effectue 4 virages à l'intérieur de cette distance, elle ne peut être inférieure à 4*175=700m. En ajoutant une marge de sécurité donnant un peu de latitude au UAV, 1000m semble être un choix judicieux.

Pour s'assurer du bon fonctionnement de l'algorithme avant son utilisation sur le système HIL, l'algorithme a été testé avec Matlab. Les résultats de simulation pour une cible se déplaçant à 23 m/s sont présentés à la figure 6.7.

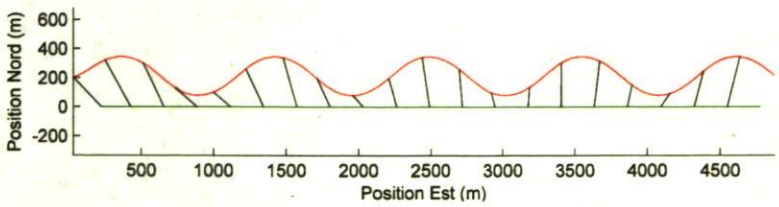


FIGURE 6.7 – Simulation de l'algorithme à trajectoire sinusoïdale

Cette figure présente à première vue 2 principaux problèmes : le UAV ne se déplace pas à la même vitesse linéaire que la cible et sa trajectoire n'est pas centrée sur celle de la cible. En effet, puisque l'algorithme ne tient pas compte de la position du UAV,

rien n'assure que le UAV suivra correctement la cible.

Le problème de vitesse provient de la différence entre la trajectoire effectuée par le UAV et le sinus parfait utilisé pour le calcul. Malgré que les commandes envoyées au UAV trace exactement le sinus, il est très improbable qu'il réussisse à le suivre parfaitement. L'accumulation d'erreurs finit par entraîner une différence non négligeable entre les deux vitesses ainsi qu'une distance grandissante séparant le UAV de sa cible.

Le second problème qui amène le UAV à suivre une trajectoire décentrée par rapport à celle de la cible est causé par le moment où est enclenchée la surveillance. En effet, si le UAV n'est pas directement en arrière de la cible lorsque l'algorithme est engagé, le UAV effectuera une trajectoire sinusoïdale décentrée. Ce problème sera d'autant plus amplifié si la cible effectue des virages.

En raison des ces problèmes, l'algorithme pourrait être rejeté. Toutefois, puisqu'il semble être une bonne idée d'effectuer une trajectoire sinusoïdale pour les cibles à grandes vitesses, les prochaines sections présentent les correctifs apportés pour que l'algorithme donne les résultats attendus.

6.1.3 Ajout d'un contrôleur sur la distance séparant le UAV de la cible

Afin de régler le problème du UAV qui ne va pas à la même vitesse que la cible malgré qu'il effectue une trajectoire sinusoïdale, il est possible de choisir l'amplitude du sinus à parcourir en fonction de la distance séparant le UAV de la cible au lieu de la choisir simplement à partir du ratio de leur vitesse respective σ. Pour ce faire, il faut tout d'abord identifier le procédé décrivant la relation entre l'amplitude du sinus et la distance d_s qu'on retrouve à la figure 6.1. Cette distance est la distance entre le UAV et la cible parallèle à la trajectoire de celle ci.

Calcul des distances entre le UAV et la cible

Les sections suivantes font appel à la distance parallèle et perpendiculaire entre le UAV et la trajectoire de la cible. La figure 6.8 présente ces deux distances.

Pour le calcul de la distance perpendiculaire, il faut d'abord trouver l'équation de la droite passant par le point (x, y) et (x_1, y_1). On sait que l'équation réduite d'une droite

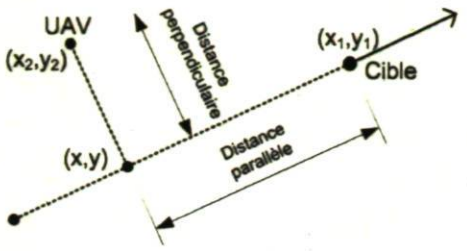


FIGURE 6.8 – Distances entre le UAV et la cible

dans le plan est de la forme : $y = mx + b$. Le terme m est la pente donnée par $tan(\theta)$ où θ est l'orientation de la cible et le terme b est l'ordonnée à l'origine. Comme cette droite passe par le point (x_1, y_1), on trouve que $b = y_1 - tan(\theta)x_1$.

En présence de l'équation réduite d'une droite dans le plan, la distance d_p entre un point et la droite est donnée selon Ouellet [40] par :

$$d_p = \frac{|mx - y + b|}{\sqrt{1 + m^2}} \tag{6.13}$$

Il suffit ensuite de remplacer x et y par la position du UAV (x_2, y_2) pour avoir la distance perpendiculaire. Pour retrouver la distance parallèle, il suffit de faire le même raisonnement en inversant la position du UAV avec celle de la cible.

Identification du procédé reliant l'amplitude du sinus à la distance parallèle entre le UAV et la cible

L'identification s'est faite par des tests de réponse à l'échelon à partir de l'équilibre. En effet, l'amplitude initiale avant le changement de commande permettait au UAV d'aller à la même vitesse que la cible.

Le tableau 6.1 résume pour deux vitesses différentes de cible les tests effectués pour identifier le procédé intégrateur dont l'équation est :

$$G(s) = \frac{K}{s} \tag{6.14}$$

Dans ce tableau, Δu représente l'échelon envoyé sur l'amplitude du sinus, Δt le temps attendu et Δy la variation de distance entre le UAV et la cible. Il est possible de voir que le procédé n'est pas linéaire puisque le gain change en fonction du point d'opération. Toutefois, une approximation en faisant la moyenne permet de le fixer à 0.02.

Tableau 6.1 – Identification du procédé

Vitesse (m/s)	Δu (m)	Δt (s)	Δy (m)	$K = \frac{\Delta y}{\Delta t \Delta u}$
20.83	128	48	106.5	0.017
20.83	78	48	74.3	0.020
20.83	-32	48	-39.7	0.026
20.83	32	48	40.9	0.027
15.83	73	158	113.2	0.010
15.83	-73	158	-127.8	0.011

Conception du régulateur

Maintenant que le procédé est connu, il est possible de faire la conception d'un régulateur proportionnel et intégral donné par :

$$G_c(s) = \frac{K_c(T_I s + 1)}{T_I s} \tag{6.15}$$

En utilisant la méthode proposée par Morari et Zafiriou [41], la fonction de transfert du régulateur résultant est :

$$G_c(s) = \frac{2(100s + 1)}{100s} \tag{6.16}$$

La même simulation que celle de la figure 6.7 a été de nouveau exécutée, mais cette fois en régulant la distance séparant le UAV de la cible. On peut voir les résultats de simulation à la figure 6.9 pour une distance voulue de 175m derrière la cible.

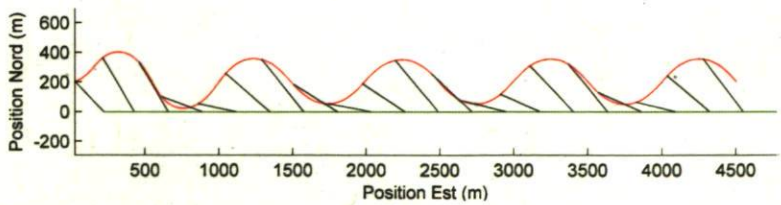


FIGURE 6.9 – Simulation de l'algorithme à trajectoire sinusoïdale avec régulateur

La figure 6.10 présente l'amplitude du sinus manipulée ainsi que la distance séparant le UAV de la cible pour cette même simulation. On voit cette fois que la distance voulue à l'arrière de la cible est respectée malgré les oscillations. Ces oscillations proviennent du mouvement sinusoïdale du UAV. Ce mouvement fait en sorte que la distance parallèle à la trajectoire de la cible varie dans le temps suivant les oscillations du sinus. En effet, lorsque le UAV est aux sommets du sinus, il s'approche de la cible plus rapidement que lorsqu'il se trouve à mi chemin entre deux sommets.

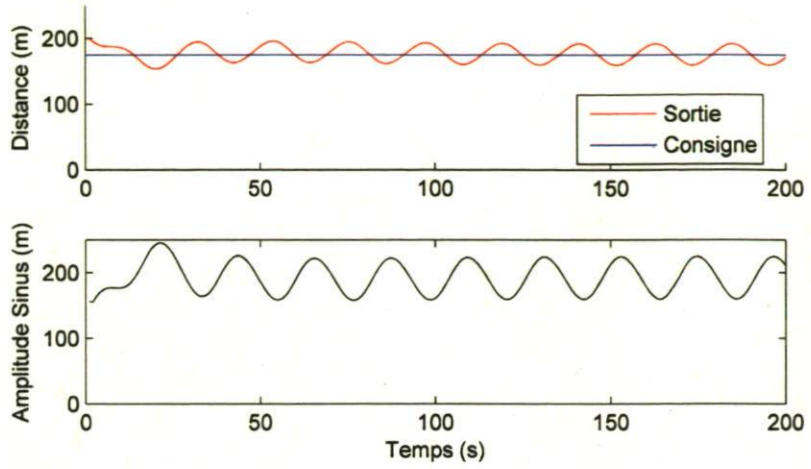


FIGURE 6.10 – Contrôle de la distance parallèle de la simulation à trajectoire sinusoïdale avec régulateur

6.1.4 Ajout d'une correction sur la trajectoire du UAV

Pour ne pas que le sinus reste décentré par rapport à la trajectoire de la cible, il est possible d'ajouter une correction sur l'orientation du UAV. Il est possible de trouver par quelle distance le UAV est décentré en comparant la distance perpendiculaire du UAV détaillée à la section 6.1.3 par rapport à la valeur y_s calculée par l'algorithme. Ces deux valeurs devraient normalement être les mêmes. La figure 6.11 présente ces deux valeurs pour la simulation de la figure 6.10.

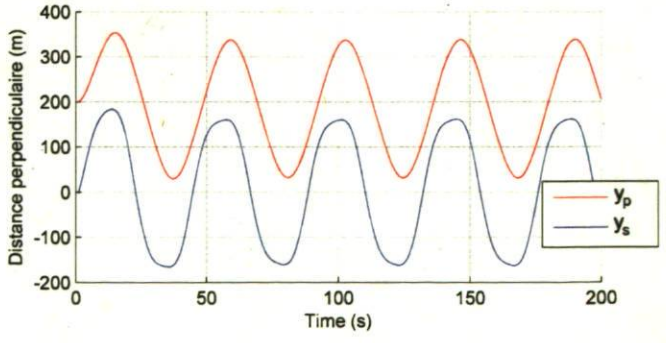


FIGURE 6.11 – Comparaison de la distance perpendiculaire du UAV et de y_s

En regardant les deux courbes, il est possible de se rendre compte qu'il y a un petit décalage entre les deux et qu'elles n'ont pas exactement la même amplitude. Le décalage est principalement dû au temps de réponse du système. En effet, la commande en orientation envoyé au UAV est basé sur y_s, il est donc naturel que le UAV soit

toujours en retard puisque son modèle d'orientation a une constante de temps de 3.78 secondes. Pour connaître quel est le décalage, il est possible de tracer la corrélation entre ces deux courbes. La corrélation présentée à la figure 6.12 est la somme de la différence entre la courbe y_s et la courbe du UAV décalée. À partir de cette figure, on peut voir que la courbe du UAV est en retard de 3 secondes.

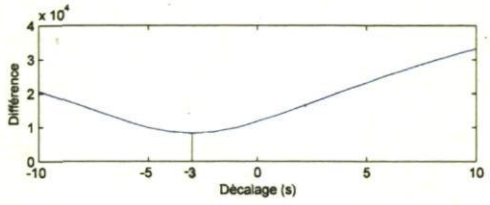


FIGURE 6.12 – Corrélation entre y_s et la distance perpendiculaire du UAV

En plus du décalage, l'amplitude des deux signaux n'est pas exactement la même. On ne peut donc pas simplement soustraire la distance perpendiculaire du UAV de la valeur de y_s pour connaître par quelle distance le UAV est décentré. La courbe de y_s a toujours une plus grande amplitude et le facteur change avec le ratio de vitesse. La figure 6.13 présente les deux signaux pour une simulation où la vitesse de la cible passe linéairement de 50 à 100 km/h.

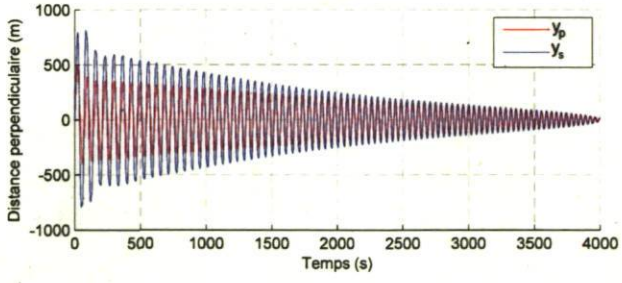


FIGURE 6.13 – Comparaison de la distance perpendiculaire du UAV et de y_s pour différentes vitesses de cible

Ce graphique permet de trouver que le facteur reliant les deux amplitudes suit la courbe suivante :

$$Facteur = -0.052966 * v_t + 2.6587; \tag{6.17}$$

où v_t est la vitesse de la cible.

Une fois ce facteur calculé, on peut finalement trouver par quelle distance le UAV est décentré en soustrayant au y_s de trois itérations précédentes la distance perpendiculaire

du UAV multipliée par le facteur trouvé précédemment. À partir de cette valeur, il est ensuite possible de calculer l'angle de correction θ exposé à la figure 6.14.

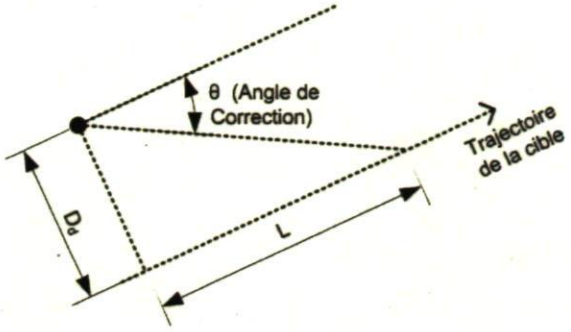


FIGURE 6.14 – Angle de correction de la trajectoire

L'angle θ se trouve par la relation :

$$\theta = atan(D_d/L) \tag{6.18}$$

où D_d est la distance décentrée et L est une longueur choisie de façon à ce que l'angle de correction ne soit pas appliqué brusquement. Plusieurs tests ont permis de fixer cette valeur à 2000m pour obtenir de bons résultats.

Pour une troisième fois, la même simulation est exécutée, mais cette fois en plus du régulateur sur la trajectoire parallèle, une correction est appliquée à la trajectoire du UAV pour la centrer sur celle de la cible. Il est possible de voir à la figure 6.15 que les deux problèmes mentionnés à la section 6.1.2 ont bien été réglés. Le UAV se déplace maintenant derrière la cible et à la même vitesse.

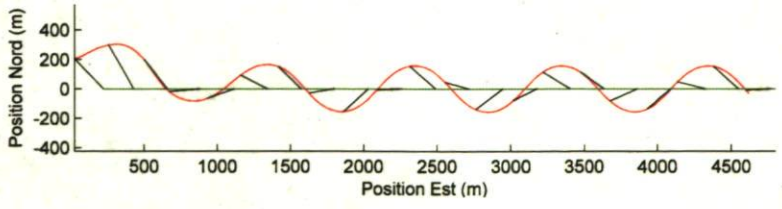


FIGURE 6.15 – Simulation de l'algorithme à trajectoire sinusoïdale amélioré

6.1.5 Utilisation de l'algorithme amélioré

Maintenant que les modifications apportées ont rendu l'algorithme fonctionnel, il est possible de s'interroger sur ses conditions d'utilisation. Puisque cet algorithme amène

le UAV à faire des oscillations derrière la cible, il importe que celle-ci ait une grande vitesse. Lee et al. [6] mentionnent qu'il est possible d'utiliser cet algorithme pour des cibles se déplaçant plus rapidement que le tier de la vitesse du UAV. Toutefois, une telle vitesse demande au UAV une grande amplitude de sinus pouvant amener certains problèmes. Une simulation avec la cible allant au tier de la vitesse du UAV est montrée à la figure 6.16.

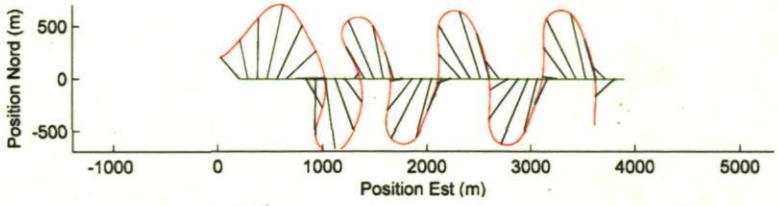


FIGURE 6.16 – Simulation avec la cible allant au tier de la vitesse du UAV

On peut voir que le UAV réussit tout de même à rester derrière la cible, mais la trajectoire ne suit plus la forme sinusoïdale. Le UAV doit même se diriger pendant de courts instants en direction opposée à la cible pour respecter la distance derrière celle-ci. Le problème provient du fait que le UAV n'est plus capable de suivre la trajectoire demandée. En effet, l'amplitude du sinus étant très élevée, le rayon de courbure au sommet du sinus devient bien en deçà du rayon de braquage minimum du UAV.

Pour les différentes formes de sinus demandées en fonction de la vitesse de la cible, il est possible de calculer le rayon de courbure minimum aux sommets du sinus ainsi que le rayon de courbure moyen situé autour de ces sommets selon Gray et al. [42] par :

$$R_c = \frac{\left[1 + \left(\frac{dy_s}{dx_s}\right)^2\right]^{\frac{2}{3}}}{\left|\frac{d^2y_s}{dx_s^2}\right|} \tag{6.19}$$

où R_c est le rayon de courbure et y_s défini selon l'équation 6.1. Le rayon de courbure moyen se calcule en sélectionnant les points du sinus où la vitesse perpendiculaire au déplacement de la cible demandée au UAV est inférieure à 90% du maximum de la vitesse perpendiculaire demandée. On peut voir à la figure 6.17 la sélection des points pour une cible se déplaçant à 10 m/s. Le premier graphique de cette figure montre les points choisis sur le sinus. Le deuxième montre le rayon de courbure calculé pour chaque point. Les deux derniers montrent quant à eux la vitesse perpendiculaire et parallèle demandée en chaque point. La sélection des points s'est faite à partir des données du troisième graphique.

La figure 6.18 montre le rayon de courbure minimum aux sommets du sinus pour différentes vitesses de cible ainsi que la moyenne des rayons de courbure pour les points

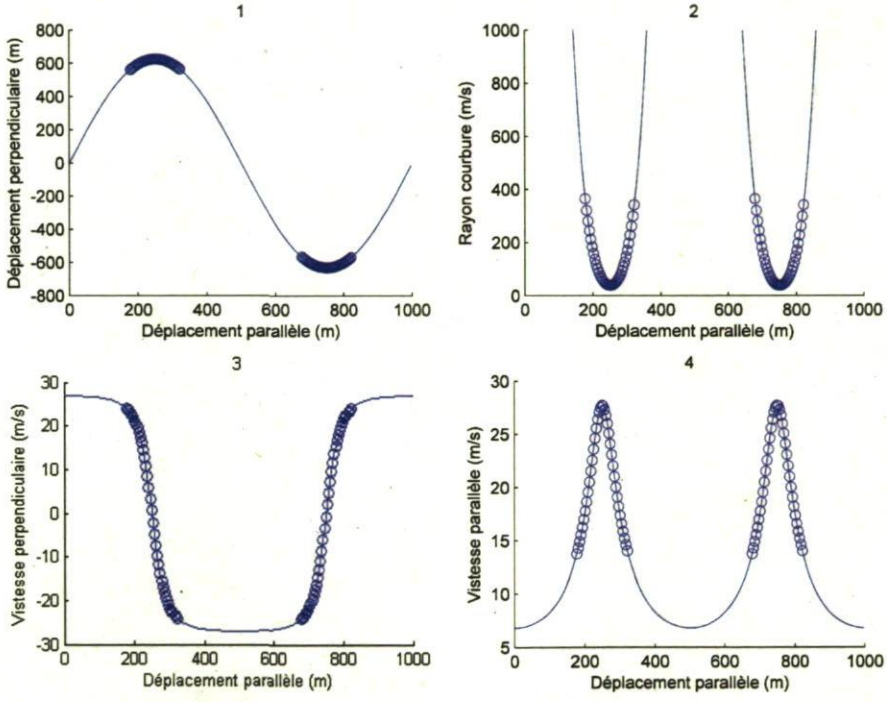


FIGURE 6.17 – Choix des points pour le calcul du rayon de courbure moyen pour une cible se déplaçant à une vitesse de 10 m/s

sélectionnés autour des sommets selon la méthode présentée précédemment. Une ligne correspondant au rayon de courbure minimum de 175 mètres du UAV à été ajoutée.

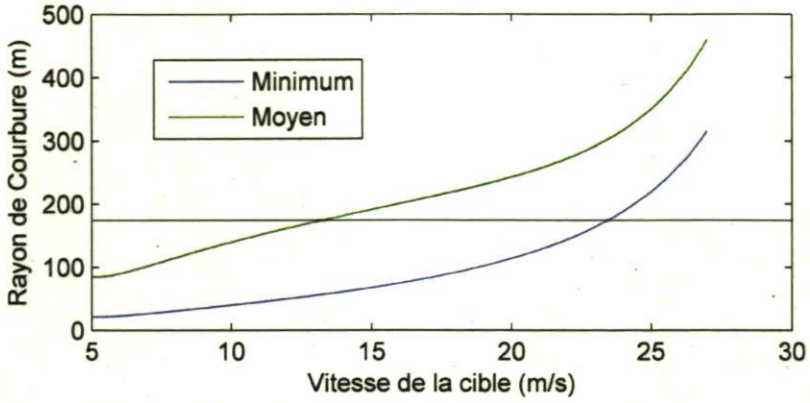


FIGURE 6.18 – Rayon de courbure minimum et moyen du sinus pour différentes vitesses

Il est possible de constater qu'il n'y a qu'au dessus de 23.46 m/s que le rayon de courbure minimum peut être atteint. Pour cette raison, la notion de rayon de courbure moyen a été amenée. En effet, le rayon de courbure minimum ne l'est que pour un seul point au sommet du sinus. La moyenne de ceux situés autour des sommets permet d'avoir une meilleure idée de la courbure demandée.

En examinant la courbure moyenne, on peut voir cette fois qu'il est possible pour le UAV d'atteindre le rayon de courbure à partir de 13.36m/s. Cette vitesse correspond presque qu'exactement à 50% de la vitesse du UAV qui est de 27.78 m/s. Par conséquent, la vitesse minimum de la cible est choisie à 50% de la vitesse du UAV.

Enfin, la figure 6.19 présente une cible se déplaçant à 13.89 m/s et permet de constater que le patron obtenu est très près de la trajectoire sinusoïdale désirée.

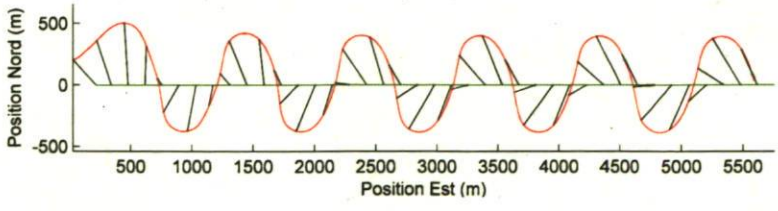


FIGURE 6.19 – Simulation avec la cible allant à 50% de la vitesse du UAV

6.2 Commande prédictive

Prévost et al. [43] propose l'utilisation de la commande prédictive à horizon fuyant pour surveiller une cible dans un environnement dynamique. Il est à noter que pour cette section, comme la notation utilisée en commande prédictive diffère de celle utilisée dans le présent mémoire, la notation choisie est celle de Prévost et al. [43] afin d'en simplifier la correspondance. À chaque intervalle de temps k, l'unité de contrôle de trajectoire (UCT) présentée à la figure 6.20 détermine les consignes optimales pour l'autopilote $\hat{\boldsymbol{u}}_{uv}(k)$ afin que le UAV se dirige vers la cible.

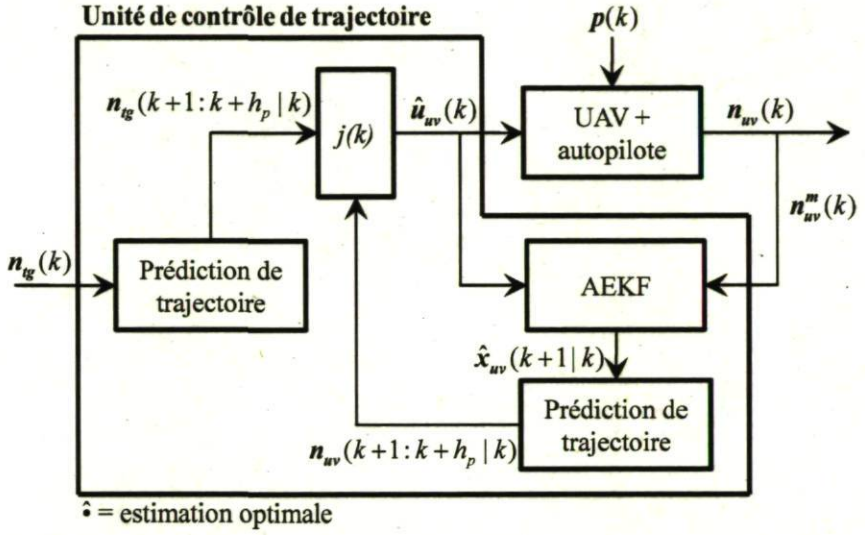


FIGURE 6.20 – Diagramme du contrôleur prédictif

Au coeur de l'UCT se trouve un « Augmented Extended kalman Filter » (AEKF) qui estime les états optimaux du UAV $\hat{\boldsymbol{x}}_{uv}(k+1|k)$ en fonction des consignes de l'autopilote $\hat{\boldsymbol{u}}_{uv}(k)$ ainsi que de la position mesurée du UAV $\boldsymbol{n}^m_{uv}(k)$. Ces états sont ensuite utilisés afin de prévoir la trajectoire du UAV $\boldsymbol{n}_{uv}(k+1:k+h_p|k)$ pour tous les intervalles de temps à l'intérieur de l'horizon de prédiction h_p. La trajectoire de la cible est aussi estimée en supposant son orientation et sa vitesse constantes. Finalement, l'UCT calcule les consignes de l'autopilote qui vont minimiser la fonction d'optimisation $j(k)$ sujette à des contraintes.

6.2.1 Filtre de Kalman étendu

Le modèle du filtre de Kalman utilisé pour l'estimation des états optimaux du UAV $\hat{\boldsymbol{x}}_{uv}(k+1|k)$ est présenté à la figure 6.21. Ces états sont calculés à partir des consignes

de l'autopilote $\hat{\boldsymbol{u}}_{\boldsymbol{uv}}(k)$ ainsi que de la position mesurée du UAV $\boldsymbol{n}^m_{\boldsymbol{uv}}(k)$.

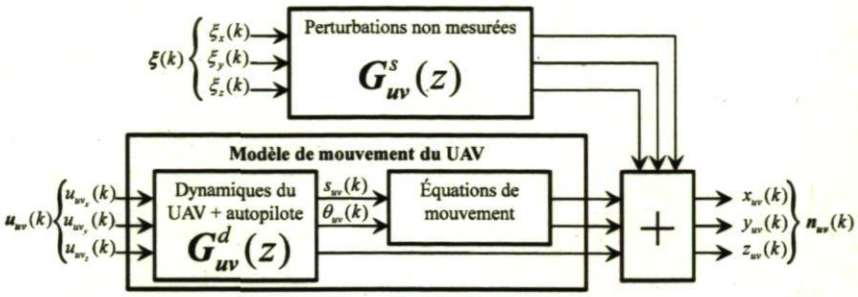


FIGURE 6.21 – Modèle dynamique du UAV

L'équivalent de la figure 6.21 en fonction d'état peut être représenté comme :

$$\boldsymbol{x}_{\boldsymbol{uv}}(k+1) = f(\boldsymbol{x}_{\boldsymbol{uv}}(k), \boldsymbol{u}_{\boldsymbol{uv}}(k), \boldsymbol{\xi}(k)) + \boldsymbol{w}(k) \tag{6.20}$$

$$\boldsymbol{n}^m_{\boldsymbol{uv}}(k) = \boldsymbol{C}\boldsymbol{x}_{\boldsymbol{uv}}(k) + \boldsymbol{v}(k) \tag{6.21}$$

où $\boldsymbol{\xi}(k) = [\xi_x(k)\ \xi_y(k)\ \xi_z(k)]^T$ est un bruit blanc affectant la position du UAV et où $\boldsymbol{w}(k)$ et $\boldsymbol{v}(k)$ sont aussi du bruit blanc représentant respectivement l'incertitude sur le modèle ainsi que sur les mesures.

La dynamique du UAV est représentée par le bloc *Modèle de mouvement du UAV* de la figure 6.21. Les différents modèles exprimant le couplage entre la vitesse du UAV s_{uv}, son orientation θ_{uv} et son altitude z_{uv} sont introduits dans la matrice de fonctions de transfert $\boldsymbol{G}^d_{\boldsymbol{uv}}(z)$. Les trois différentes consignes servent d'entrée à la matrice et les sorties suivent la dynamique modélisée.

Les équations de mouvement suivantes permettent ensuite de connaître la position du UAV dans les axes x et y suite aux nouvelles sorties en vitesse $s_{uv}(k)$ et en orientation $\theta_{uv}(k)$.

$$x(k+1) = x(k) + s_{uv}(k)\cos\theta_{uv}(k) \tag{6.22}$$

$$y(k+1) = y(k) + s_{uv}(k)\sin\theta_{uv}(k) \tag{6.23}$$

Le bloc *Perturbations non mesurées* est ajouté pour compenser l'action de perturbations constantes comme le vent qui viennent affecter la dynamique du UAV. Il est composé d'intégrateurs alimentés par du bruit blanc $\boldsymbol{\xi}(k)$ sur la position mesurée du UAV $\boldsymbol{n}^m_{\boldsymbol{uv}}(k)$ tel que :

$$\boldsymbol{G}_{\boldsymbol{uv}}^{s}(z) = \begin{bmatrix} \left(\frac{z}{z-1}\right)^2 & 0 & 0 \\ 0 & \left(\frac{z}{z-1}\right)^2 & 0 \\ 0 & 0 & \left(\frac{z}{z-1}\right) \end{bmatrix} \tag{6.24}$$

6.2.2 Prédiction de trajectoire

À chaque intervalle de temps, les trajectoires du UAV et de la cible sont prédites. La prédiction optimale sur la trajectoire du UAV minimise la variance sur l'erreur de prédiction de position pour chaque intervalle de temps. De cette façon, la trajectoire du UAV est calculée à partir du modèle dynamique du UAV présenté à la figure 6.21 avec un bruit blanc nul. Comme précédemment mentionné, la trajectoire de la cible est prédite en assumant que sa vitesse, son orientation ainsi que son altitude demeurent constantes.

6.2.3 Fonction objectif et contraintes

À chaque intervalle de temps T, l'UCT calcule les consignes de l'autopilote sur l'horizon de contrôle h_c. Ces consignes sont trouvées en minimisant à l'aide d'une fonction d'optimisation une fonction objectif $j(k)$ qui doit respecter certaines contraintes. Cette fonction objectif est présentée par l'équation :

$$j(k) = \sum_{T=1}^{h_p} \|\boldsymbol{u}_{\boldsymbol{uv}}(k+T|k) - \boldsymbol{u}_{\boldsymbol{tg}}(k+T|k)\|_2 \tag{6.25}$$

Cette équation additionne les distances entre les positions prédites du UAV et de la cible sur tout l'horizon de prédiction h_p. Cette façon de faire tente de faire correspondre l'orientation du UAV à celle de la cible au temps d'interception.

Afin que les consignes trouvées soient réalisables, des contraintes sont ajoutées pour limiter les consignes ainsi que leurs incréments conformément aux équations suivantes :

$$\Delta \boldsymbol{u}_{\boldsymbol{uv}}^{min} \leq \Delta \boldsymbol{u}_{\boldsymbol{uv}}(0 : h_c - 1|k) \leq \Delta \boldsymbol{u}_{\boldsymbol{uv}}^{max} \tag{6.26}$$

$$\boldsymbol{u}_{\boldsymbol{uv}}^{min} \leq \boldsymbol{u}_{\boldsymbol{uv}}(0 : h_c - 1|k) \leq \boldsymbol{u}_{\boldsymbol{uv}}^{max} \tag{6.27}$$

6.2.4 Simulation de l'algorithme

La figure 6.22 présente les résultats de simulation avec Matlab pour une cible se déplaçant à 50 km/h ou 50% de la vitesse du UAV.

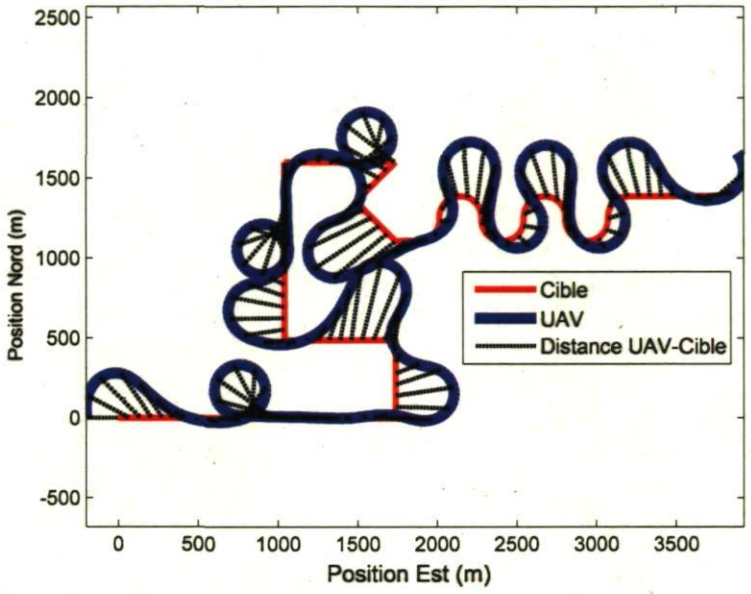


FIGURE 6.22 – Simulation en utilisant la commande prédictive

Cette simulation semble s'être bien déroulée, mais elle comporte toutefois deux problèmes empêchant que cet algorithme soit utilisé sur le système HIL. Tout d'abord, le UAV se dirige directement vers la cible. Il a été démontré à l'aide des résultats de simulation de l'algorithme de poursuite directe qu'un algorithme amenant le UAV directement vers la cible va forcément provoquer une perte de poursuite. Le second problème provient du fait que rien n'assure que la caméra pourra garder la cible dans son champ de vue puisque que l'algorithme ne tient pas compte des limites du mouvement panoramique horizontal (pan) de $\pm 160°$.

Malgré que l'algorithme tel que présenté ne peut être simulé sur le système HIL, la commande prédictive qu'il utilise permet toutefois d'ajouter des contraintes pour tenir compte des limites de la caméra. Il est donc possible d'ajouter une contrainte permettant de garder une certaine distance minimum entre le UAV et la cible, forçant ainsi le UAV à tourner autour de la cible. Il est aussi possible d'ajouter une contrainte empêchant que l'angle entre le vecteur pointant dans l'orientation du UAV et le vecteur partant du UAV vers la cible sorte des limites de $\pm 160°$ imposées par la caméra.

Bien qu'il n'est pas été prêt lors des simulations sur le système HIL, l'algorithme a tout de même été modifié afin de tenir compte de ces deux contraintes. La figure 6.23 présente les résultats de la même simulation avec le nouvel algorithme.

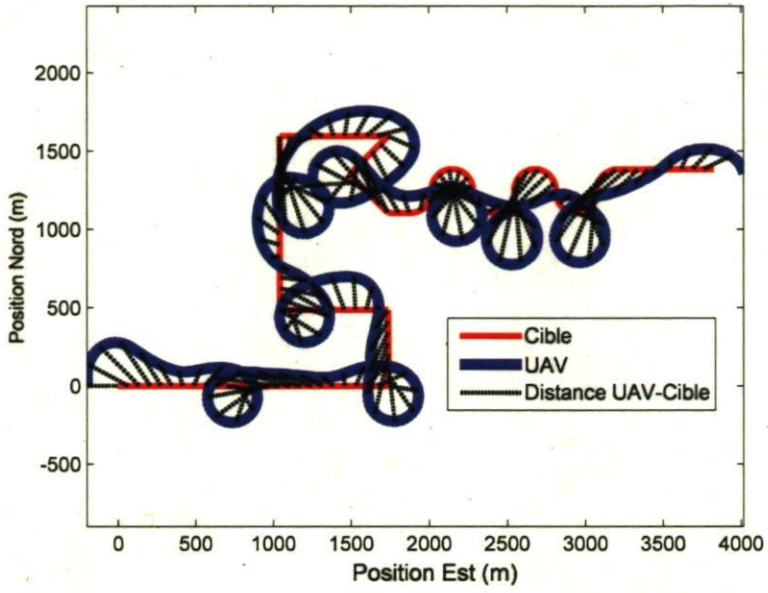


FIGURE 6.23 – Simulation en utilisant la commande prédictive avec contraintes

Pour s'assurer que l'algorithme respecte les contraintes imposées, la figure 6.24 présente l'angle entre la direction du UAV et la cible. Cet angle demeure à tout instant à l'intérieur des limites de $\pm 160°$, ce qui démontre la fonctionnalité de l'algorithme en pure simulation. Cet algorithme n'a pu être testé sur le système HIL compte tenu de sa complexité et du manque de temps pour en faire l'intégration. Toutefois, puisque les simulations du chapitre 5 ont permis de constater que les résultats obtenu sur le système HIL étaient très près des simulations Matlab pures, il est possible de croire que cet algorithme aurait aussi donné de bons résultats sur le système HIL.

6.3 Conclusion

Ce chapitre a présenté deux algorithmes avancés. Le premier amène le UAV à effectuer une trajectoire sinusoïdale derrière la cible. Cette technique a été développée pour des vitesse de cible supérieure à 50% de la vitesse du UAV. Elle a l'avantage de fixer la distance entre la cible et le UAV et de la garder presque constante. Cette méthode a toutefois dû être adaptée pour devenir fonctionnelle puisqu'elle présentait

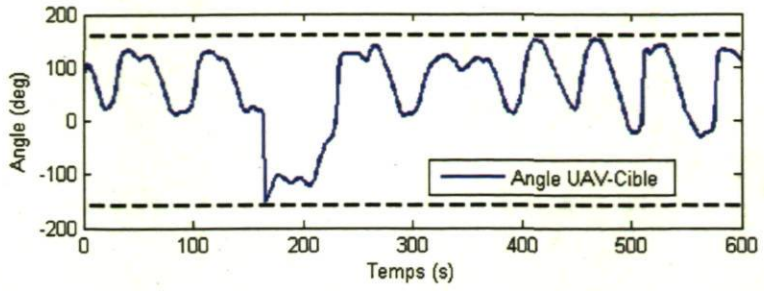


FIGURE 6.24 – Angle entre la ldirection du UAV et la cible pour la simulation utilisant la commande prédictive avec contraintes

deux principaux problèmes. Tout d'abord, rien n'assurait que la distance séparant le UAV de la cible aller demeurer constante. Un régulateur a été intégré à l'algorithme à ce propos. De plus, une correction de trajectoire a été ajoutée puisque la trajectoire du UAV ne suivait pas nécessairement celle de la cible. Le deuxième algorithme utilise la commande prédictive à horizon fuyant avec contraintes. Cet algorithme nécessite encore plus de puissance de calcul puisqu'il utilise des fonctions d'optimisation. Il est très efficace puisqu'il utilise des prédictions sur la trajectoire du UAV et de la cible et il est possible d'y ajouter des contraintes dépendamment de la simulation. Par exemple, il a été possible d'y ajouter les contraintes du système de caméra sans difficulté. De plus, cet algorithme permet de manipuler les consignes en altitude et en vitesse ajoutant une souplesse suplémentaire.

Chapitre 7

Comparaison des algorithmes de surveillance

À l'aide d'un scénario de déplacement de cible et du montage HIL, ce chapitre fait une comparaison des performances entre les algorithmes de surveillance de base et ceux plus complexes. Comme les différents algorithmes de surveillance de base ont donné des résultats assez similaires au chapitre 5, seulement l'algorithme BHSC qui a donné la meilleure performance au niveau de la distance moyenne séparant le UAV de la cible est simulé ici. Pour ce qui est des algorithmes avancés, seulement l'algorithme à trajectoire sinusoïdale est étudié puisqu'il n'a pas été possible de testé l'algorithme à commande prédictive sur le montage HIL. Enfin, puisque le scénario choisi comprend une section à haute et à basse vitesse, une analyse des résultats de simulation permet de faire ressortir les forces et faiblesse des deux algorithmes simulés ainsi que les lignes directrices d'une stratégie de guidage efficace.

7.1 Scénario de test

Pour qu'il soit possible de comparer les différents algorithmes de surveillance, un scénario de déplacement de la cible présenté à la figure 7.1 a été créé où la cible effectue des virages ainsi que des changements de vitesse. La cible se déplace en ligne droite pendant 200 secondes entre les points A et C. Sa vitesse initiale est de 20.83 m/s. Elle accélère linéairement jusqu'à mi-chemin (B) où elle atteint 25 m/s puis elle décélère pour retourner à 20.83 m/s. Elle effectue ensuite deux virages successifs de 90° dans le sens anti-horaire suivis d'une ligne droite de 50 secondes entre D et E. Elle ralentit

durant 50 secondes jusqu'à 13.89 m/s, puis elle entreprend ensuite des virages adoucis de ±60° durant 200 secondes entre F et G.

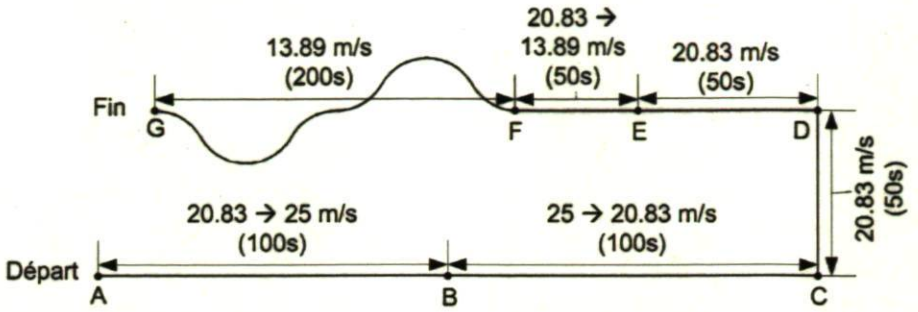


FIGURE 7.1 – Parcours effectué par la cible pour le scénario étudié

7.2 Résultats

La figure 7.2 présente les résultats de simulation HIL de l'algorithme à trajectoire sinusoïdale pour le scénario de test décrit précédemment. Le début de la simulation se déroule très bien. L'algorithme n'a aucune difficulté à ajuster l'amplitude du sinus afin de faire correspondre la vitesse de déplacement du UAV à celle de la cible. Les 2 virages ne causent pas non plus de problème, mais la section à basse vitesse amène une perte de poursuite. Malgré une intervention manuelle pour reprendre la poursuite, la faible vitesse de la cible ainsi que les virages qu'elle effectue ne permettent pas à l'algorithme de récupérer et de revenir fonctionnel. La distance moyenne entre le UAV et la cible pour cette simulation est de 373 mètres.

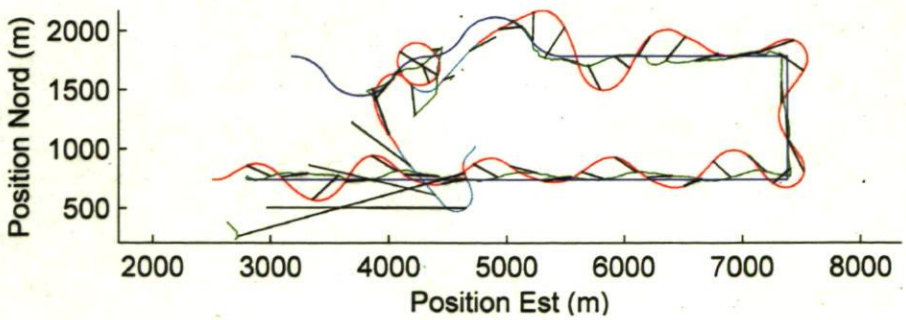


FIGURE 7.2 – Résultats de simulation de la méthode à trajectoire sinusoïdale : UAV –, Cible – ,Cible estimée –

Puisque les algorithmes de base ont été étudiés et comparés au chapitre 5, seulement un seul est simulé ici. L'algorithme BHSC a été choisi puisqu'il a offert les meilleures

performances lors des tests précédents basées sur la distance moyenne séparant le UAV de la cible. On peut voir les résultats de la simulation sur la figure 7.3. Contrairement à l'algorithme à trajectoire sinusoïdale, celui-ci semble avoir bien fonctionné durant les 550 secondes de simulation puisqu'aucune perte de poursuite est survenue. Toutefois, la première boucle autour de la cible durant la ligne droite à haute vitesse a amené le UAV à s'éloigner considérablement de la cible. La distance moyenne entre les deux entités reste tout de même meilleure que pour la simulation de l'algorithme à trajectoire sinusoïdale, soit 342 mètres.

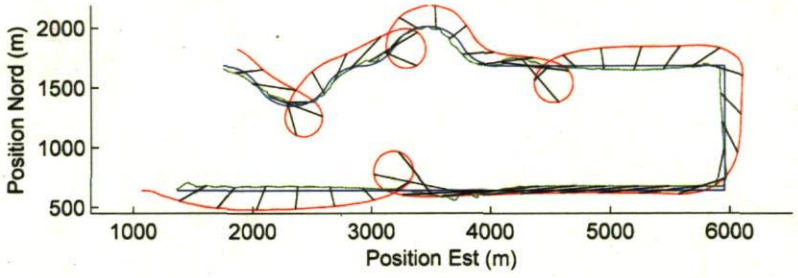


FIGURE 7.3 – Résultats de simulation de la méthode BHSC : UAV –, Cible – ,Cible estimée –

Malgré qu'il n'a pas été possible de tester l'algorithme à commande prédictive sur le système HIL, la même simulation a tout de même été exécutée sous Matlab. La figure 7.4 présente les résultats. Il est possible de voir que cet algorithme est très prometteur, puisqu'il produit une distance moyenne entre le UAV et la cible de seulement 259 mètres.

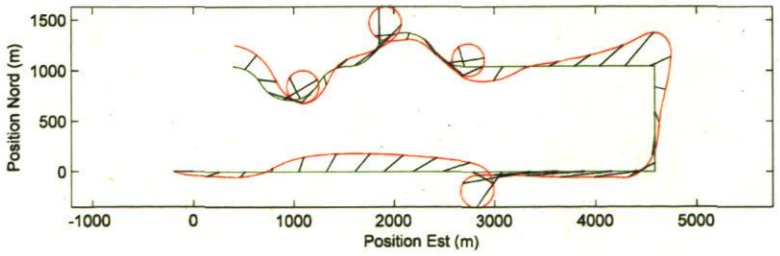


FIGURE 7.4 – Résultats de simulation de la méthode à commande prédictive : UAV –, Cible –

7.3 Analyse

Puisque le scénario de test est composé d'une section à haute vitesse ainsi que d'une à basse vitesse, il permet de bien faire ressortir les avantages et les inconvénients des deux algorithmes étudiés sur le système HIL. L'algorithme à trajectoire sinusoïdale est incapable de fonctionner lorsque la cible se déplace lentement, tandis qu'il excelle à grande vitesse. Pour ce qui est de l'algorithme BHSC, il est parfait pour de petites vitesses, mais il peut amener une perte de poursuite à grande vitesse. En effet, la distance séparant le UAV de la cible devient très grande après que le UAV l'ait contournée. Le même phénomène peut se produire pour l'algorithme à commande prédictive, puisqu'il amène lui aussi le UAV à contourner la cible.

Pour mieux comparer la section à haute vitesse des deux algorithmes, elles ont été extraites des résultats de simulation pour être étudiées indépendamment. La figure 7.5 présente la section à grande vitesse pour l'algorithme à trajectoire sinusoïdale. Le UAV reste à tout moment près de la cible, ce qui occasionne une distance moyenne entre le UAV et la cible de seulement 258 mètres.

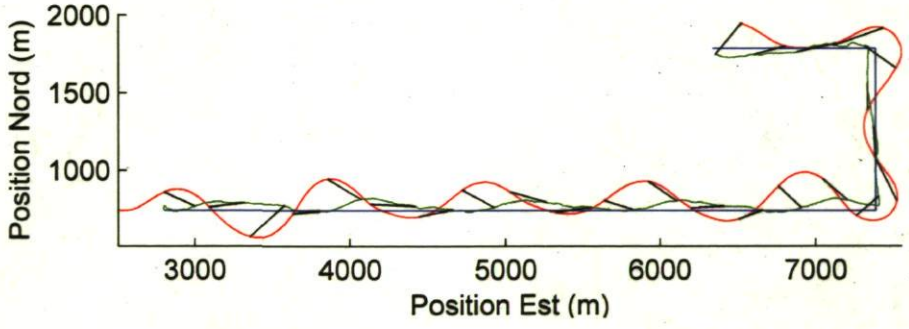


FIGURE 7.5 – Section à haute vitesse de l'algorithme à trajectoire sinusoïdale : UAV –, Cible – ,Cible estimée –

La figure 7.6 montre cette fois la section à grande vitesse pour l'algorithme BHSC. La distance moyenne est cette fois de 399 mètres. Puisque cette distance est beaucoup plus élevée que celle obtenue avec l'algorithme à trajectoire sinusoïdale, il est difficile de se contenter de cet algorithme pour un scénario diversifié. Une utilisation simultanée des deux algorithmes semble donc un choix éclairé. L'algorithme de base pour les sections à basse vitesse ou avec beaucoup de virages et l'algorithme avancé pour les sections à grande vitesse.

La figure 7.7 montre encore une fois le même scénario, mais cette fois avec un changement d'algorithme arrivant après 300 secondes. On peut voir maintenant les

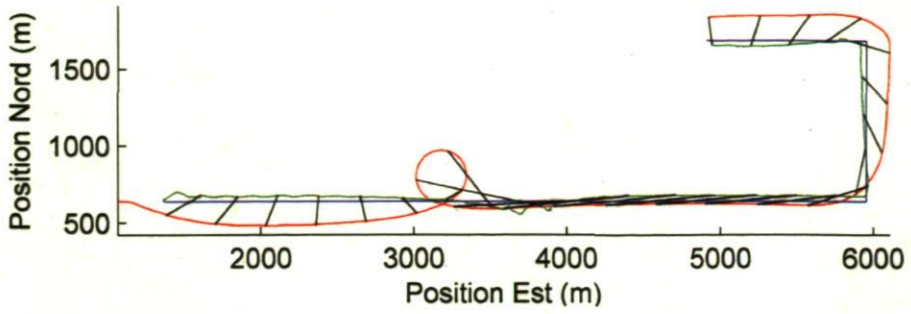


FIGURE 7.6 – Section à haute vitesse de l'algorithme BHSC : UAV –, Cible – ,Cible estimée –

avantages des deux algorithmes, ce qui diminue grandement la distance moyenne à 263 mètres.

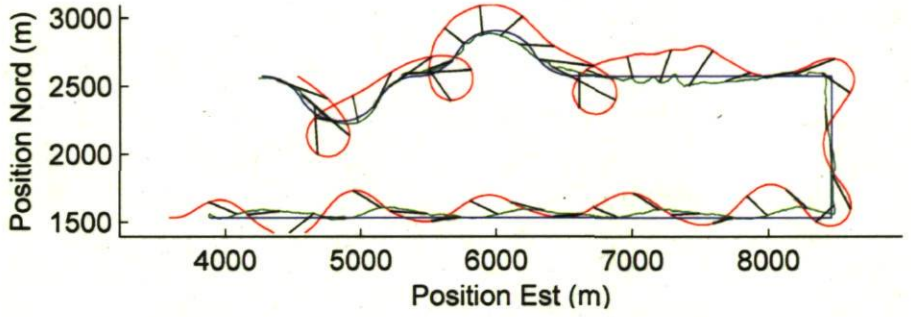


FIGURE 7.7 – Simulation avec une utilisation de l'algorithme à trajectoire sinusoïdale et l'algorithme BHSC : UAV –, Cible – ,Cible estimée –

7.4 Conclusion

Suite aux tests effectués dans ce chapitre, il semble qu'il n'y ait pas d'algorithme magique qui offre les meilleures performances dans toutes les circonstances. Pour une cible à grande vitesse, les algorithmes de base et l'algorithme à commande prédictive risquent d'entraîner une perte de poursuite causée par la grande distance séparant le UAV de la cible lorsque le UAV la contourne, tandis que l'algorithme à trajectoire sinusoïdale ne peut fonctionner pour des cibles à basse vitesse. Toutefois, un choix d'algorithmes qui dépend de la vitesse de la cible peut s'avérer une très bonne solution. Chaque algorithme peut alors être utilisé dans ses conditions optimales pour ainsi rester à tout moment le plus près de la cible possible tout en évitant des pertes de poursuite. Il est alors possible de tirer profit d'un algorithme comme celui qui commande des trajectoires sinusoïdales lorsque la cible se déplace à haute vitesse et d'utiliser un algorithme plus simple lorsque

la cible se déplace lentement. Au lieu de faire le choix des algorithmes manuellement, il aurait été intéressant de développer un système qui l'aurait fait automatiquement en fonction de la dynamique de la cible. L'utilisateur n'aurait alors pas à se questionner à ce sujet et les interventions humaines seraient encore plus diminuées.

Chapitre 8

Conclusion

8.1 Commentaires généraux

Ce mémoire a présenté le système « hardware in the loop » construit autour d'un autopilote de UAV et d'un système vidéo de poursuite de cible commercial dans le but d'évaluer des algorithmes de surveillance et de poursuite de cible. Le système permet de tester les algorithmes de surveillance sans le risque, le coût et la complexité associés avec les vols extérieurs. Dans un premier temps, les mathématiques nécessaires à l'évaluation de la position de la cible ont été présentées. Ensuite, le système HIL original a été décrit en expliquant clairement les différentes interactions entre l'autopilote MP2028 de la compagnie Micropilot, l'ordinateur exécutant le modèle de UAV à six degrés de liberté et l'ordinateur exécutant le logiciel de contrôle au sol Horizon. Les modifications matérielles et logicielles ayant permis l'intégration du système vidéo de poursuite de cible ont aussi été présentées. En plus de l'ordinateur du système vidéo, un autre ordinateur à été utilisé afin de générer le champ de vue de la caméra virtuelle par le biais d'un module d'extension ajouté à l'environnement 3D X-Plane. Puisque la caméra est virtuelle, des modèles de ses servomoteurs ont été identifiés par des tests de réponse à l'échelon et intégrés au modèle à six degrés de liberté pour assurer un déplacement conforme de la caméra. Par ailleurs, les modifications faites au logiciel de gestion de simulation SerialBridge ont été présentées ainsi que la manette qui lui a été intégrée pour contrôler la caméra virtuelle et le système vidéo de poursuite de cible. Enfin, à l'aide de simulations, il a été montré que le système était fonctionnel et qu'il permettait l'évaluation systématique des algorithmes de surveillance tout en amenant du réalisme aux expérimentations.

Plusieurs algorithmes de base amenant le UAV à tourner autour d'une cible ont été

testés sur le système HIL et comparés. Les algorithmes BHSC et ACR ont l'avantage de garder le UAV près de la cible produisant ainsi une meilleure estimation de sa position. Les algorithmes VCL et CBT amènent quant à eux moins d'oscillations au niveau du contrôle de la caméra entraînant une plus grande stabilité de la caméra et par le fait même une diminution du risque de perte de poursuite. L'algorithme APD amène de son côté le UAV à passer directement au dessus de la cible. Malgré le fait que cet algorithme entraîne la plus petite distance entre le UAV et la cible en plus d'être très simple, il n'est pas possible de l'utiliser sur le montage HIL puisqu'il provoque des pertes de poursuite causées par les limitations du système de caméra. Ensuite, deux algorithmes plus complexes ont été présentés. Le premier amenant le UAV à effectuer une trajectoire sinusoïdale derrière la cible a été modifié afin qu'il puisse être testé sur ce même système. Le second utilisant la commande prédictive n'a pu être testé sur le système HIL, mais offrait tout de même de bons résultats en simulation. Enfin, une comparaison des algorithmes a été effectuée à l'aide d'un scénario impliquant une cible à vitesse et direction variables. Elle a pu permettre de faire ressortir les avantages deux catégories d'algorithmes. Les algorithmes amenant le UAV à tourner autour de la cible offrent de meilleures performances pour une cible à basse vitesse. Celui amenant le UAV à osciller derrière la cible offre quant à lui son plein potentiel pour une cible qui se déplace rapidement. De ces observations résulte qu'il ne faut pas trancher entre les deux catégories, mais plutôt les utiliser de façon hybride en fonction de la dynamique de la cible.

8.2 Travaux futurs

Il y a beaucoup de travail qui pourrait être fait dans le futur sur ce système. Tout d'abord, l'algorithme à commande prédictive et contraintes présenté dans ce mémoire est très prometteur en simulation pure, mais des simulations sur le système HIL valideraient l'efficacité de cet algorithme. De plus, beaucoup de paramètres ont été fixés lors des différents tests. Il aurait été intéressant de faire varier des paramètres comme la vitesse du UAV, son altitude, le rayon du cercle pour les algorithmes de bases ou même le zoom de la caméra en fonction de la vitesse de la cible. Par exemple, lorsque le UAV se retrouve éloigné de la cible, le UAV pourrait tirer avantage d'une vitesse plus élevée et d'un plus grand zoom permettant de mieux détecter la cible. Un système pourrait aussi être développé pour faire le choix des algorithmes en temps réel en fonction du déplacement de la cible. Présentement, l'usager doit faire ce choix manuellement à l'aide de l'interface graphique mise à sa disposition. Par ailleurs, le modèle de UAV utilisé présentement est l'Aerosonde et l'autopilote est réglé pour garder le UAV dans une enveloppe de vol très sécuritaire pour ce modèle. Il serait intéressant de tester des

modèles de UAV plus manœuvrables et d'ajuster les boucles de contrôle de l'autopilote en conséquence. Le rayon de courbure du UAV pourrait alors être diminuer grandement et ainsi améliorer la surveillance. En outre, la caméra modélisée présentement sur le système ajoute beaucoup de limitations. L'utilisation d'une caméra contrôlée en vitesse n'offrant pas de limitation au niveau du déplacement panoramique horizontal permettrait de tester un plus grand éventail d'algorithmes. Enfin, d'autres perturbations pourraient être ajoutées tel que du vent pour rendre le système encore plus réaliste avant d'effectuer de vraies évaluations d'algorithmes de surveillance à l'extérieur.

Bibliographie

[1] Ryan A. and Hedrick J.K., *A mode-switching path planner for UAV-assisted search and rescue*, IEEE Conference on Decision and Control, Seville, Spain, 1471-1476, 2005.

[2] Girard A.R., Howell A.S. and Hedrick J.K., *Border Patrol and Surveillance Missions using Multiple Unmanned Air Vehicles*, IEEE Conference on Decision and Control, Atlantis, Paradise Island, Bahamas, 620-625, 2004.

[3] Frew E.W. and Lawrence D.A., *Cooperative Stand-off Tracking of Moving Targets by a Team of Autonomous Aircraft*, AIAA Guidance, Navigation and Control Conference, San Francisco, California, 7, 4885-4895, 2005.

[4] Klein D.J. and Morgansen K.A., *Controlled Collective Motion for Trajectory Tracking*, American Control Conference, Minneapolis, USA, 5269-5275, 2006.

[5] Wise R.A. and Rysdyk R.T., *UAV Coordination for Autonomous Target Tracking*, AIAA Guidance, Navigation and Control Conference, Keystone, Colorado, 3210-3231, 2006.

[6] Lee J., Huang R., Vaughn A., Xiao X., Hedrick J.K., Zennaro M. and Sengupta R., *Strategies of Path-Planning for a UAV to Track a Ground Vehicle*, Autonomous Intelligent Networks and Systems Conference, Menlo Park, California, 2003.

[7] Husby C.R., *Path Generation Tactics for a UAV Following a Moving Target*, Master's thesis, Air Force Institute of Technology, Wright Patterson Air Force Base, Ohio, 2005.

[8] Frew E.W. Lawrence D.A., Dixon C., Elston J. and Pisano W.J., *Lyapunov Guidance Vector Fields for Unmanned Aircraft Applications*, American Control Conference, New York City, USA , 371-376, 2007.

[9] Stolle S. and Rysdyk, R., *Flight Path Following Guidance for Unmanned Air Vehicles With Pan-Tilt Camera for Target Observation*, Digital Avionics Systems Conference, Indianapolis, USA, 2003.

[10] Rysdyk R., *UAV Path Following for Constant Line-of-Sight*, 2nd AIAA Unmanned Unlimited Systems,Technologies, and Operations Aerospace, Land and Sea Conference, Paper 6626, San-Diego, USA, 2003.

[11] Quigley M., Goodrich M.A., Griffiths S., Eldredge A. and R.W. Beard, *Target Acquisition, Localization, and Surveillance Using a Fixed-Wing Mini-UAV and Gimbaled Camera*, IEEE International Conference on Robotics and Automation, Barcelona, Spain, 2600-2606, 2005.

[12] Rafi F., Khan S. M. , Shafiq K. and Shah M., *Autonomous target following by unmanned aerial vehicles*, SPIE Defense and Security Symposium, Orlando, Florida, 2006.

[13] Chen H., Chang K.C. and Agate C.S., *A dynamic path planning algorithm for UAV tracking*, SPIE Signal Processing, Sensor Fusion, and Target Recognition XVIII, Orlando, Florida, Vol. 7336, P. 73360B, 2006.

[14] Prévost C.G., Desbiens A. and Gagnon, E., *Extended Kalman Filter for State Estimation and Trajectory Prediction of a Moving Object Detected by an Unmanned Aerial Vehicle*, American Control Conference, New York, USA, 1805-1810, 2007.

[15] Li L. and Ding J., *Ground Moving Target Tracking Control System Design for UAV Surveillance*, 2007 IEEE International Conference on Automation and Logistics, Jinan, China, 1458-63, 2007.

[16] Barber D.B., Redding J.D., McLain T.W., Beard R.W. and Taylor, C.N., *Vision-based target geo-location using a fixed-wing miniature air vehicle*, Journal of Intelligent and Robotic Systems, Vol. 47, No. 4, 361–382, 2006.

[17] Gibbins D., Roberts P. and Swierkowski L., *A video geo-location and image enhancement tool for small unmanned air vehicles (UAVs)*, Intelligent Sensors, Sensor Networks and Information Processing Conference, Melbourne, Australia, 469-473 2004.

[18] Sohn S., Lee B., Kim J. and Kee C., *Vision-based real-time target localization for single-antenna GPS-guided UAV*, IEEE Transactions on Aerospace and Electronic Systems, Vol. 44, No. 4, 1391-1401, 2008.

[19] Whitacre W., Campbell M., Wheeler M. and Stevenson D., *Flight Results from Tracking Ground Targets Using SeaScan UAVs with Gimballing Cameras*, American Control Conference, New York, USA, 377-383, 2007.

[20] Gans N.R., Dixon W.E., Lind R. and Kurdila A., *A hardware in the loop simulation platform for vision-based control of unmanned air vehicles*, Mechatronics, 1043-1056, 2009.

[21] Shixianjun, jiakun S. and Hongxing L., *Hardware-in-the-loop Simulation Framework Design For a UAV Embedded Control System*, 2006 Chinese Control Conference, Harbin, China, 2006.

[22] Manai M., Gagnon E. and Desbiens A., *Identification of a UAV and Design of a Hardware-in-the-Loop System for Nonlinear Control Purposes*, AIAA Guidance, Navigation, and Control Conference, San Francisco, California, 8, 6441-6446, 2006.

[23] Boivin E., Desbiens A. and Gagnon E., *Cooperative Predictive Control for Collision Avoidance with Unknown Ellipsoid Obstacles*, SIAM Journal on Control and Optimization, 2006.

[24] Bélanger J., Desbiens A. and Gagnon E., *UAV Guidance with Control of Arrival Time*, American Control Conference, New York, USA, 4488-4493, 2007.

[25] Forsyth D.A., and Ponce J., *Computer Vision : A Modern Approach*, Prentice Hall, 2003.

[26] Craig, J.J., *Introduction to Robotics : Mechanics & Control*, Addison-Wesley, 1986.

[27] Dobrokhodov V.N., Kaminer I.I., Jones K.D. and Ghabcheloo R., *Vision-Based Tracking and Motion Estimation for Moving Targets Using Small UAVs*, American Control Conference, Minesota, USA, 2006.

[28] MP2028 Autopilot, `http://www.micropilot.com/products-mp2028g.htm`, juillet 2009.

[29] X-Plane, `http://www.x-plane.com`, juillet 2009.

[30] National Instruments, `http://www.ni.com/`, juillet 2009.

[31] PerceptiVU Tracking System, `http://www.perceptivu.com/PerceptiVU.html`, juillet 2009.

[32] Directed Perception, `http://www.dperception.com`, juillet 2009.

[33] 3D Connexion, `http://www.3dconnexion.com/`, juillet 2009.

[34] Zoso N. and Wong F.C., *Stabilisation d'une caméra sur UAV par méthode des gyroscopes*, DRDC Valcartier TR 2007-440, Juillet 2007.

[35] X-Plane SDK, `http://www.xsquawkbox.net/xpsdk`, juillet 2009.

[36] Theriault O., *SerialBridge - Manuel d'utilisation*, DRDC Valcartier CR 2009-143, Août 2009.

[37] Corriou J-P., *Process Control : Theory and Applications*, Springer, 2004.

[38] Monda M.J., Woolsey C.A. and Konda R.C., *Ground Target Localization and Tracking in a Riverine Environment from a UAV with a Gimbaled Camera*, AIAA Guidance, Navigation, and Control Conference, Hilton Head, USA, 3788-3801, 2007.

[39] Thériault O., Poulin E., Gagnon E., Wong F. and Desbiens A., *Tracking Algorithm Evaluation with a Integrated Video Tracker and UAV HIL Facility*, AIAA Guidance, Navigation, and Control Conference, 2009-6265, Chicago, 2009.

[40] Ouellet G., *Algèbre Linéaire - Vecteur et géométrie*, Le Griffon d'argile, 1994.

[41] Morari M. and Zafiriou E., *Robust Process Control*, Prentice Hall, 1989.

[42] Gray A., Abbena E. and Salamon S., *Modern Differential Geometry of Curves and Surfaces with Mathematica 3rd Edition*, Chapman & Hall/CRC, 2006.

[43] Prevost C., Theriault O., Desbiens A. and Poulin E., *Receding Horizon Model-Based Predictive Control for Dynamic Target Tracking; a Comparative Study*, AIAA Guidance, Navigation and Control, Chicago, USA, 2009.