**ERM AUTOMATISMES INDUSTRIELS**

280 Rue Edouard Daladier 84200 CARPENTRAS

Tél : 04 90 60 05 68 - Fax : 04 90 60 66 26

Site : <http://www.erm-automatismes.com/>

E-Mail : [Contact@erm-automatismes.com](mailto:Contact@erm-automatismes.com)



[1 Principe 1](#_Toc457812800)

[1.1 Objectif, principe général 1](#_Toc457812801)

[1.2 Environnement recréé 1](#_Toc457812802)

[2 Structure de la chaine d’asservissement 2](#_Toc457812803)

[2.1 Consigne/sortie 3](#_Toc457812804)

[2.2 Blocs Nacelle + GYRO 4](#_Toc457812805)

[2.3 Chaine de retour 4](#_Toc457812806)

[2.4 Elaboration de la consigne 7](#_Toc457812807)

[2.5 Déplacement de la cible 9](#_Toc457812808)

[3 Utilisation de l’interface vidéo 10](#_Toc457812809)

[3.1 Technique de réglage des seuils 10](#_Toc457812810)

[3.2 Fonction initialisation 10](#_Toc457812811)

[3.3 Bouton viser 10](#_Toc457812812)

[3.4 Bouton suivi 11](#_Toc457812813)

[4 Programmation en python par l’élève : mode personnalisé 12](#_Toc457812814)

[4.1 Structure de la boucle générale 13](#_Toc457812815)

[4.2 Programmation de la partie reconnaissance du centre 14](#_Toc457812816)

[4.3 Programmation de la partie Elaboration de la consigne 14](#_Toc457812817)

[5 Modification des paramètres externalisés. 15](#_Toc457812818)

**Dossier ressource**

**Nacelle de drone avec video**



Nacelle de drone en mode video.docx

# Principe



nacelle

drone

camera

## Objectif, principe général

La fonctionnalité « **suivi de cible** » permet de placer systématiquement et automatiquement un sujet à filmer (appelé cible) au centre de l’image.

Le retour d’image fourni par la caméra permet de connaitre les mouvements de la cible dans l’image et d’en déduire les déplacements à imposer à la nacelle afin de recentrer automatiquement la cible sur l’image.

## Environnement recréé

Pour travailler en mode vidéo avec la nacelle, il faut disposer :

* Du système « nacelle de drone instrumentée »
* D’un ordinateur avec le Viewer video de la nacelle
* D’un second écran qui permettra de projeter la cible

Le suivi se fait dans une seule direction en configuration classique. C’est d’ailleurs la configuration qui correspond à l’utilisation réelle du système drone + nacelle sans son 3eme axe.

*Remarque* : le suivi peut aussi s’effectuer dans les 2 directions, mais dans un mode boucle ouverte uniquement.

# Structure de la chaine d’asservissement

On se place dans la configuration classique de suivi dans une direction.

Cible mobile

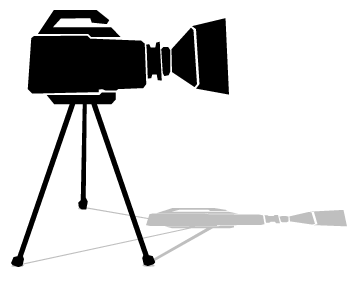
M

**Roulis**

Tension Phase 1

Tension Phase 1

Tension Phase 1



**y**

**Image vue par la caméra**

**θN**

α

**yN**

C

**y**

**θC**

**y**c

A

Notations

A : point situé à l’intersection de l’axe de tangage et de l’axe de la caméra

C : point situé au centre de la cible

|  |  |  |
| --- | --- | --- |
|  | Axe de la caméra (ligne de visée de la caméra) | |
| ) | Axe défini par la droite (AC), droite « nacelle-cible » | |
|  |  |  |
|  | Angle entre l’axe de la caméra et la droite (AC) | (°) |
|  | Consigne d’angle  : nulle dans le cas d’un suivi | (°) |
|  | Image de l’angle , issue de la mesure par traitement d’image | (°) |
|  |  |  |
| Y | Position réelle de la cible dans l’image caméra, en pixel de l’image.caméra | (pixel) |
| Ymes | Position mesurée de la cible dans l’image caméra, position obtenue par traitement d’image, en pixels de l’image caméra | (pixel) |
|  |  |  |
| **θN =** | Position angulaire de l’axe de la caméra | (°) |
| **θN\_mes** | Position angulaire de l’axe caméra mesurée par la plateforme inertielle par rapport à l’horizontale | (°) |
| **θcons** | Consigne d’angle **θN**, envoyée à la nacelle pour corriger l’écart angulaire | (°) |
|  |  |  |
| **θc=** | Position de la cible donnée en angle (angle entre l’horizontale et la droite (AC) | (°) |
| **Yc** | Position de la cible dans l’écran de projection, en pixel de l’écran. | (pixel) |

Schéma-bloc

αcons= 0  
(°)

PID

Ɛ

(°)

+

-

**θcons**

**(°)**

αmes

(°)

GYRO

+

-

**θ**c (°)

**θN**

**(°)**

**θN\_mes**

-

+

α

(°)

+

**θN\_mes(°)**

Adapt.

Ymes

(pixel)

CAMERA

+ Seuillage

TRAITEMENT D’IMAGE

Y

(pixel)

NACELLE

+

PID

-

Géométrie Ecran/Caméra

**Y**c (pixels)

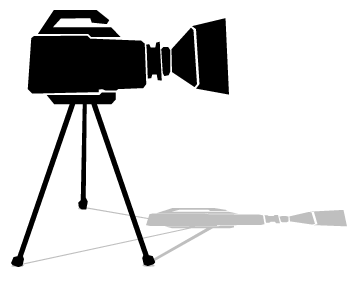
***Mode « VIDEO + GYRO évolué »***

(°)

## Consigne/sortie

La grandeur asservie est l’angle α, angle entre l’axe de la caméra et l’axe « nacelle-cible ».

L’objectif est d’annuler cet angle afin de placer la cible au centre de l’image filmée, c’est pourquoi la consigne αcons sera toujours nulle.



**θN**

C

A

*Schéma dans le cas d’une consigne non nulle pour mise en place de la relation :*

En mode ***« VIDEO + GYRO évolué »*** :

*Ligne de visée souhaitée*

**θcons**

Le choix a été fait de considérer une consigne en angle αcons plutôt qu’une consigne de position en pixel sur l’image filmée par la caméra, ou de position en mètres sur l’image restituée. Ceci permet de caractériser les performances du système nacelle en suivi vidéo. En considérant une position (en pixel ou en mètre), les caractéristiques des moyens de restitution de l’image (écran, …) influeraient sur les performances.

## Blocs Nacelle + GYRO

La nacelle peut-être pilotée en **boucle fermée ou en boucle ouverte** suivant la structure d’asservissement choisi.

θcons

(°)

NACELLE +PID

GYRO

+

-

**θN**

**(°)**

**θN\_mes**

Dans le cas général, le pilotage se fait en **boucle fermée** (concerne les structures d’asservissement *« Video + Gyro sans référence »* ; *« Video+ Gyro évolué »* ; *« Personnalisé »*), le retour est effectué par la plateforme inertielle qui mesure l’angle **θN\_mes**, angle formé par l’axe de la caméra avec l’horizontale.

Cette boucle d’asservissement comporte un correcteur PID, dont les valeurs sont réglées par défaut à KpBFnac = 800, KiBFnac = 800, KdBFnac = 1500. Concrètement cet asservissement est réalisé par la carte Arduino associée à la nacelle.

Les valeurs des coefficients PID peuvent si besoin être modifiées en cliquant sur le bloc.

En mode *« Video »* ou *« avec Roll »*, la nacelle est placée en **boucle ouverte** (pas de retour de la plateforme inertielle). Seul le coefficient Kp est alors pris en compte. Sa valeur est par défaut KpBOnac=800, mais peut-être modifiée comme en boucle fermée si besoin (cf paragraphe « Modification des paramètres externalisés » p15).

Cette partie du schéma fonctionnel correspond au système nacelle seule, avec les réglages de PID indiqué ci-dessus.

## Chaine de retour

**Caméra**

Le capteur utilisé pour la chaine de retour est la caméra **HD-3000**, qui a en fait un double rôle puisqu’elle sert aussi à renvoyer la scène filmée à l’utilisateur.

Cette caméra retourne une image en couleur au format BGR. La taille de l’image retournée est en général de taille 640\*480 pixels, même si théoriquement cette caméra pourrait fonctionner en HD (1280\*720).

Une taille d’image importante, même si elle aurait tendance à améliorer la précision, a souvent un effet néfaste sur la qualité de l’asservissement car les temps de traitement s’en trouvent allongés, et peuvent induire des retards dans la chaine. Le choix a été fait de conserver une taille d’image traitée de taille 640\*480. Ce paramètre peut néanmoins être modifié (voir paragraphe « Modification des paramètres externalisés » p15).

*Scène filmée*



*Capteur de caméra*

*Lentille*

Angle de vision

Longueur focale

|  |  |
| --- | --- |
| Angle de vision | 68° |
| Fréquence d’acquisition | 30 images par seconde |
| Résolution | 1280\*720 |

**Caractéristiques constructeur caméra HD3000**

L’image traitée par l’algorithme de traitement d’image est de taille 640\*480.

La longueur de la diagonale de cette image est donc de : pixels

En faisant l’hypothèse de linéarité de la caméra, on obtient le coefficient liant l’angle α (en °) et la position de la cible dans l’image filmée Y (en pixels\_image).

α

(°)

Y

(pixels\_image)

α

(°)

Y

(pixels\_image)

La fréquence d’acquisition annoncée par le constructeur est de 30 images par secondes, mais il n’est possible de ne récupérer que 15 images par seconde, soit une période d’acquisition de 60ms.

**Traitement de l’image**

Le traitement de l’image est réalisé par un programme Python, associé à l’interface (« Nacelle\_NC10\_video »). Il utilise notamment la bibliothèque OpenCV, qui fonctionne avec Python 2.

Le choix a été fait d’utiliser le principe de traitement d’image qui a l’avantage d’être plus simple et de pouvoir être programmé en partie par l’élève sous Python. Industriellement, des algorithmes type « Tracking » sont couramment utilisés.

L’algorithme proposé utilise les étapes suivantes :

* **Redimensionnement de l’image** :

L’image est redimensionnée de façon à obtenir une image de hauteur 480 pixels (ou autre valeur si modification de ce paramètre par le professeur).

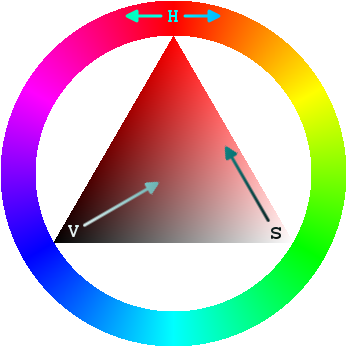
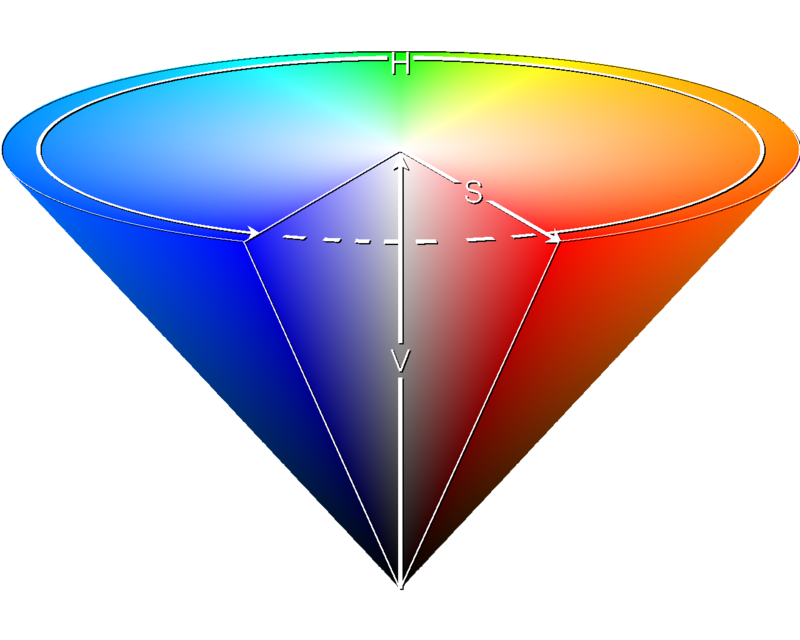
* **Conversion du format BGR au format HSV**.

Le format HSV est plus intéressant pour la reconnaissance d’une forme de couleur par traitement d’image.

H : Hue (Teinte)

S : Saturation

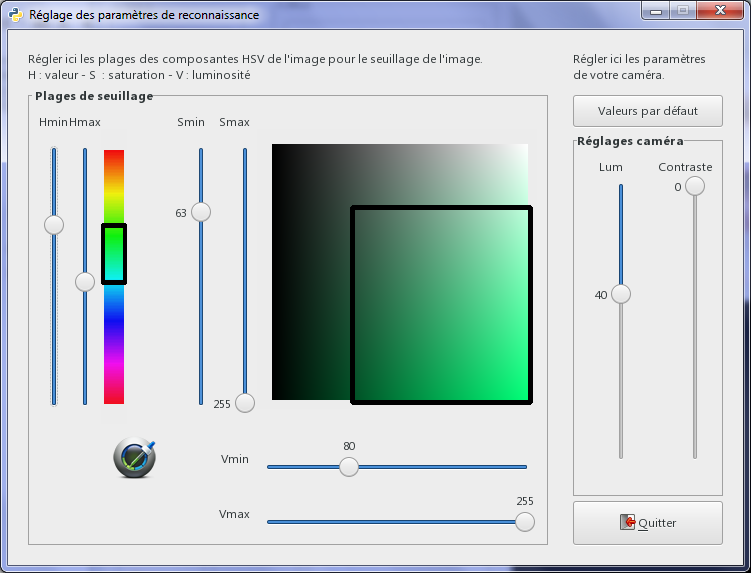
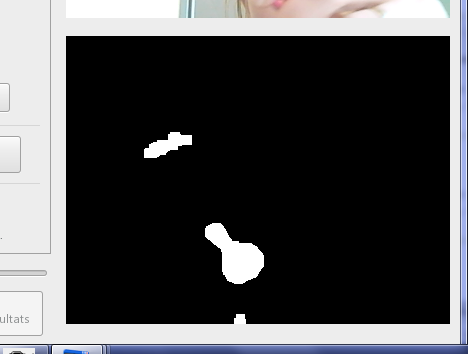
V : Value (Valeur)



**Le modèle HSV de définition des couleurs**

* **Seuillage**

Les pixels dont les valeurs de couleurs sont contenues dans une zone définie par l’utilisateur sont fixés la valeur 1 (couleur blanche), les autres sont noirs. Le résultat est une image seuillée en noir et blanc.



**Plage de couleurs conservées**

**image seuillée**

* **Traitement d’érosion-dilatation**

Ce traitement a pour but de supprimer les formes de petites tailles (poussières) sur l’image seuillée. Le principe est d’enlever le bord de la forme (érosion), puis de dilater la forme en utilisant le même coefficient. Les formes de taille importante ne sont quasiment pas modifiées, en revanche les poussières disparaissent. Le coefficient est modifiable par le professeur (voir paragraphe « Modification des paramètres externalisés » p15).

* **Recherche du centre.**

A partir de l’image seuillée, un algorithme recherche le centre de la forme détectée. L’algorithme utilisé recherche simplement le barycentre de l’image seuillée en utilisant des fonctions Numpy. Cette partie du traitement est programmable par l’élève et fait l’objet d’un TP d’informatique pour tous.

Le temps mis pour effectuer l’ensemble de ces traitements est en général inférieur à la période d’acquisition de la caméra (0.06s).

**Bloc adaptation de la chaine de retour**

αmes

(°)

Kadapt

Ymes

(pixel)

La chaine caméra + traitement d’image permet d’obtenir la position de la cible dans l’image Ymes.

Dans le paragraphe précédent on a vu que la caméra pouvait être modélisée par un gain de pixels/°.

On utilise donc un bloc d’adaptation dans la chaine de retour de soit 0.0856 °/pixels.

## Elaboration de la consigne

La consigne angulaire envoyée à la nacelle θcons est une consigne en degrés envoyée à la nacelle via la carte Arduino. Elle est élaborée à partir de l’angle mesuré αmes et d’un correcteur PID.

αcons= 0  
(°)

PID

Ɛ

(°)

+

-

**θcons**

**(°)**

αmes

(°)

+

**θN\_mes(°)**

-

(°)

*Remarque : techniquement, la carte Arduino reçoit une consigne en millidegrés, le bloc 1/1000 n’existe pas dans l’algorithme de commande*.

**Bloc PID**

Un correcteur PID, bien distinct du correcteur PID propre à la nacelle (cf §2.2 nacelle + gyro p 4), est associé à l’asservissement par caméra. Sa fonction de transfert est :

Les valeurs du gain proportionnel, du coefficient intégral et du coefficient sont modifiables à partir de l’interface.

Un gain de 1000 permet d’avoir un gain global de 1.

**Structures d’asservissement proposées**

Mode « VIDEO+ GYRO sans référence » / « VIDEO+ GYRO évolué »

Il est important de noter que l’angle **θN** caractérisant la position de la nacelle est l’angle formé par la nacelle avec l’axe horizontal. Le pilotage de la nacelle se fait donc toujours par rapport à l’horizontale et non pas par rapport à sa position actuelle. Ce comportement est dû au mode de pilotage de la nacelle en boucle fermée avec retour par plateforme inertielle et à la technologie brushless du moteur (commande en position et pas en vitesse).

La grandeur mesurée Ymes, qui est la position de la cible dans l’image filmée, est une grandeur image de l’angle αmes qui, lui, est l’angle entre la position actuelle de l’axe caméra et la ligne de visée souhaitée ). Pour rappel, la consigne αcons est nulle. L’angle αmes est donc image du déplacement angulaire qu’il faudrait imposer à la nacelle pour recentrer la cible dans l’image. Il s’agit d’un déplacement par rapport à la position actuelle de la nacelle et non pas par rapport à l’horizontale.

Ce déplacement peut être calculé en effectuant l’opération : . (voir figure p 3)

C’est ce qui est fait en **mode « VIDEO + GYRO évolué »**. On utilise le retour angulaire de la plateforme gyroscopique pour construire la consigne.

αcons= 0  
(°)

PID

Ɛ

(°)

+

-

**θcons**

**(°)**

αmes(°)

+

**θN\_mes(°)**

-

(°)

En réglant, le PID comme un gain proportionnel pur , on obtient donc théoriquement un angle de consigne qui compenserait le décalage angulaire avec la cible.

Ce mode est théoriquement précis, même sans intégrateur (voir TP).

L’autre mode proposé est le **mode « VIDEO + GYRO sans référence »,** qui réalise l’asservissement sans utiliser la mesure d’angle par la plateforme gyroscopique.

PID

Ɛ

(°)

+

-

**θcons**

**(°)**

αmes

(°)

GYRO

+

-

**θN\_mes**

NACELLE

+

PID

(°)

αcons= 0  
(°)

Le comportement sera fondamentalement imprécis, mais pourra être rendu précis en utilisant un intégrateur. (voir TP)

***Illustration des différences de comportement mode « VIDEO + GYRO sans référence » /mode « VIDEO + GYRO évolué »***

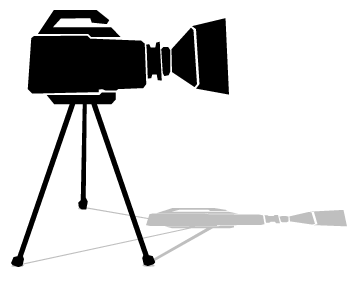
M

**Roulis**

Tension Phase 1

Tension Phase 1

Tension Phase 1



y

**Image vue par la caméra**

**θN**

α

**yN**

C

**y**

A

θcons

*On se place dans la situation où la cible est légèrement placée au-dessus du centre dans l’image filmée.*

*Le déplacement angulaire de la nacelle permettant la correction devrait être positif (« vers le haut ») pour corriger l’écart.*

***En mode « VIDEO + GYRO sans référence »****, la consigne angulaire* θcons *est bien un angle image de α positif, mais défini par rapport à l’horizontale, ce qui conduit dans ce cas à un déplacement « vers le bas » de la nacelle. Le déplacement de la nacelle est au départ à l’inverse de ce qui est attendu. La présence d’un intégrateur corrigera la position avec le temps.*

***En mode « VIDEO + GYRO évolué »****, le déplacement de la nacelle a bien lieu « vers le haut » dès le départ, puisque la consigne est calculée à partir de la position actuelle.*

Mode « VIDEO »

Comme indiqué dans le paragraphe 2.2, la nacelle est placée en boucle ouverte dans ce mode. Le seul capteur utilisé est ici la caméra. La plateforme gyroscopique n’est pas utilisée dans ce mode.

Mode « Personnalisé »

Ce mode permet de programmer la partie élaboration de la consigne directement en Python et est détaillé dans le paragraphe 3. (p11).

Mode « Avec Roll »

Ce mode permet de réaliser un suivi dans les 2 directions, mais nécessite de pivoter la structure supérieure d’un quart de tour. La plateforme gyroscopique se trouve alors inutilisable en l’état car les positions des moteurs ont été modifiées par rapport à l’horizontale. Il a donc été choisi de placer les 2 moteurs en boucle ouverte, et de n’utiliser que le retour caméra.

**Limitation du déplacement à 20°**

Dans tous les cas, de déplacement en tangage et en roulis est limité à 20° maximum. Cette contrainte est inhérente au fonctionnement du moteur brushless (évite un saut sur le pôle suivant).

## Déplacement de la cible

Il est possible de faire fonctionner le suivi en utilisant une cible quelconque (objet coloré), à condition que cet objet se distingue bien de son environnement par sa couleur.

**Y**c(pixels)

**θN**

**(°)**

-

+

Géométrie écran/caméra

**θ**c(°)

Pour pouvoir effectuer des TP, il est nécessaire d’avoir une description précise du mouvement de la cible. On utilise alors un écran secondaire qui sert à projeter une cible dont le mouvement pourra être paramétré par l’utilisateur avec des signaux types (Echelon, rampe, sinus…). La couleur de la cible et sa forme peuvent aussi être choisis.

Le déplacement de la cible **Y**c est paramétré en pixels. Ces pixels correspondent à ceux de l’écran de projection de la cible.

Ce déplacement de cible est lancé en parallèle du processus de traitement et d’asservissement, lors du lancement de la mesure.

L’algorithme est prévu pour qu’en cas de ralentissement de la tâche d’asservissement (suite à un temps de traitement très long par exemple), le déplacement de la cible ne soit pas perturbé (les 2 tâches sont lancées parallèlement de manière indépendantes).

En revanche, les mesures sont prises à chaque itération du processus de traitement/asservissement.

# Utilisation de l’interface vidéo

Les documents « Prise en main nacelle video » et le fichier d’aide associé aux Viewer NC10 présentent l’utilisation de l’interface.

Quelques points importants sont repris ci-dessous :

## Technique de réglage des seuils

La qualité du suivi dépend très fortement de la qualité de la reconnaissance de la cible et donc du seuillage. L’étape de réglage est fondamentale avant de commencer un suivi ou même une visée.

Le but est d’obtenir une image seuillée, sur laquelle la forme à détecter apparait en blanc alors que le reste de l’image est uniformément noir.

1. S’assurer au préalable que l’objet à détecter (cible) se distingue bien du reste de la scène filmée par ses propriétés de couleurs. Vérifier notamment que d’autres objets situés en arrière-plan ou dans l’environnement immédiat, n’aient pas une couleur voisine de l’objet à détecter. Il a été remarqué qu’une cible verte sur fond noir permet le plus souvent de respecter cette contrainte.
2. Sur l’écran « image brute », vérifier que la couleur de la cible est bien visible. Si ce n’est pas le cas, procéder à un réglage des propriétés de luminosité/contraste. On peut aussi modifier l’arrière-plan de l’écran cible.

*Il arrive souvent (notamment si la cible est affichée sur un écran d’ordinateur) que celle-ci apparaisse dans des tons blafards, s’approchant du blanc ; un réglage de la luminosité résout en général ce problème.*

1. Commencer par effectuer un préréglage en utilisant l’outil pipette.
2. Affiner ce réglage, si nécessaire, en utilisant les curseurs (Hmin, Hmax, Smin, Smax, Vmin et Vmax)
3. (optionnel) Si le réglage n’est toujours pas satisfaisant, placer tous les curseurs Hmin, Smin et Vmin au minimum, puis Hmax, Smax et Vmax au maximum : l’image seuillée est alors entièrement blanche. Limiter progressivement la zone, jusqu’à n’avoir plus que la cible en blanc.

## Fonction initialisation

L’appui sur ce bouton déclenche l’initialisation de la position de la cible.

Cette opération permet de placer la cible au centre de l’image caméra (conditions initiales nulles), avant de commencer la mesure. Le principe est de conserver la position de la nacelle (horizontale en général), et de déplacer la cible de manière à placer celle-ci au centre de l’image.

Ce processus utilise un asservissement dont le retour est obtenu par caméra et traitement d’image. La qualité du centrage dépend donc de la qualité du traitement d’image, et nécessite un bon réglage des seuils de colorimétrie pour pouvoir fonctionner.

Dans le cas où la cible n’est pas détectée, le processus s’arrête.

Cette fonction n’a d’intérêt que dans le cas où on utilise un écran secondaire pour la cible (et pas un objet coloré).

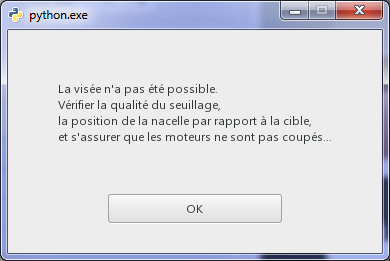
## Bouton viser

L’appui sur le bouton déclenche une « visée » de la cible, le but étant de placer la cible au centre de l’image filmée, par déplacement de la nacelle et donc de la caméra.

Le processus s’arrête automatiquement dès que la cible est considérée comme étant placée au centre de l’image filmée. On considère que c’est le cas si la distance entre le centre de l’objet reconnu et le centre de l’écran est inférieure à 4 pixels pendant 20 tours de boucles consécutifs (soit environ 1 seconde en mode Standard). Cette valeur en pixels peut être modifiée par le professeur (voir paragraphe « Modification des paramètres externalisés » p15).

Durant la visée, l’élaboration des consignes se fait en mode « VIDEO + GYRO évolué » avec un PID réglé sur : P=1000 ; I=200 ; D=0. Ces valeurs peuvent être modifiées par le professeur (voir paragraphe « modification des paramètres externalisés » p15).

Si, au bout de 10 secondes, l’objectif n’est toujours pas atteint, on considère que la cible est « inatteignable » dans les conditions et le processus s’arrête. La fenêtre suivante s’affiche alors :



Ce bouton a un intérêt si l’on souhaite démarrer un suivi avec une position initiale de la nacelle qui n’est pas l’horizontale. Sinon, il est plus judicieux d’utiliser le bouton initialisation (voir paragraphe précédent).

## Bouton suivi

L’appui sur ce bouton déclenche la mesure.

2 processus sont alors envoyés en parallèle :

* Le déplacement de la cible sur la fenêtre « cible » (placée sur un écran secondaire), suivant les paramètres entrés dans la fenêtre « Mesures » (ces paramètres sont rappelés pour mémoire dans la partie « Consigne »)
* Le suivi de la cible par la nacelle

Lorsque la mesure est terminée, les 2 processus s’arrêtent, le tracé des courbes de résultat est alors possible en cliquant sur « Visualiser les résultats ». La durée de la mesure a été réglée dans la fenêtre « Mesures ».

Si l’appui sur le bouton suivi ne provoque aucun déplacement de la cible (alors que les déplacements de la cible ne sont pas réglés sur Aucun) ou que le déplacement de la cible est saccadé, il peut être utile d’agir sur le paramètre periode\_rafraichissement\_cible (voir paragraphe « modification des paramètres externalisés » p15).

# Programmation en python par l’élève : mode personnalisé

2 parties peuvent être programmées par l’élève directement en Python :

* La partie « Elaboration de la consigne »
* La partie « Recherche du centre de la cible »

PID

Ɛ

(°)

+

-

**θcons**

**(°)**

αmes

(°)

NACELLE +PID

GYRO

+

-

**θ**c

(°)

**θN**

**(°)**

**θN\_mes**

+

-

α

(°)

**θN\_mes(°)**

Ymes

(pixel)

CAMERA

+ Seuillage

Y

(pixel)



Ces 2 fonctions ont été choisies car elles sont accessibles en termes de programmation par un élève et ne nécessitent pas de bibliothèques très spécifiques.

Les bibliothèques utilisées sont principalement ***time*** et ***numpy***.

D’autres bibliothèques sont importées dans le programme principal et pourraient être utilisées si besoin : ***threading, serial, os, cairo, cv2, csv***.

Attention, l’interface est programmée en Python 2.7. Les fonctions de l’élève devront être compatibles avec python 2.7. Bien faire noter une différence importante : en python 2 le signe « / » est la division entière :

>>> 5/2

2

Pour forcer la division :

>>> 5/2.

2.5

Il est conseillé de tester les fonctions en dehors de l’application avec une installation classique de Python, puis d’intégrer ensuite la fonction programmée dans l’interface avec nacelle.

## Structure de la boucle générale

|  |  |
| --- | --- |
| ret, im\_b = lecture\_image\_camera(LumCont) #capture d’une première image  while (t<(duree)) and ret:  t=time.clock()-t0 #temps ecoule depuis le début de la mesure  #mesure des angles  anglePitch,angleRoll=LectureAngles()  #traitement image et affichage  [x,y,imc,ims]=**traitement\_im**(im\_b,hsv,interf.algo\_trait) #traitement de l'image x,y : position du point reconnu  pitch,roll,t1,Olddepl\_y,Olddepl\_x,PitchSum,RollSum=**fe\_c.elaboration\_consigne**(y,x,Olddepl\_y,Olddepl\_x,hautIm,largIm,t1,PID,PitchSum,RollSum,anglePitch,angleRoll)  EnvoiChaine('sp ', 'spRoll ', str(roll), False) # Envoi à Arduino de la commande de roulis  EnvoiChaine('sp ', 'spPitch ', str(pitch), False) # Envoi à Arduino de la commande de tangage  #stockage des resultats dans L  L=[0.]\*11 #liste temporaire pour stockage resultats  L[0]=t #temps  L[1]=(y\_cible) #positions cible  L[6]=(x\_cible)  L[2]=hautIm/2.-y #positions detectees  L[7]=x-largIm/2.  L[3]=pitch/1000. #consignes de pitch et roll après PID video  L[8]=roll/1000.  L[5]=anglePitch/1000. #angles mesures  L[10]=angleRoll/1000.  Lres.append(L)  #capture de l'image suivante  ret, im\_b = lecture\_image\_camera(LumCont)  ecriture\_resultats(Lres) #ecriture des résultats  EnvoiChaine('sp ', 'spRoll ', '0', False) # consigne de roulis à zero  EnvoiChaine('sp ', 'spPitch ', '0', False) #consigne de tangage à zéro | Mise à jour du temps t  Capture d’une image  Mesure des angles par plateforme gyroscopique  Détermination du centre par traitement de l’image  Elaboration de la consigne  Envoi des consignes à Arduino  Capture d’une image  Stockage des résultats  Ecriture des résultats dans fichier  Remise en position de réf. de la nacelle  t<duree  et Retour caméra |

## Programmation de la partie reconnaissance du centre

On accède à cette fonction en cliquant sur le bloc **Traitement** du schéma fonctionnel.

L’élève définit la fonction **cherche\_centre** qui est appelée, après le seuillage, par la fonction **traitement\_im**. Les arguments sont imposés ainsi que le type de sortie.

Cette fonction est appelée à chaque tour de boucle du programme général. L’élève n’a pas à programmer la boucle.

Cette fonction détermine le centre de la forme détectée à partir de l’image seuillée.

**L’argument** de la fonction est une **image seuillée** en noir et blanc.

**Les sorties** sont les **coordonnées (xc,yc)** du centre trouvé, de type *flottant*. L’origine est le coin supérieur gauche de l’écran.

Un corrigé possible est disponible dans le fichier ***« algorithmes\_perso\_trait\_cor »*** du répertoire :

**« C:\Program Files (x86)\Nacelle\_NC10\_Video\Exe\_VR »** (*répertoire d’installation*).

*Remarque : l’image brute (au format BGR (0-255), sous la forme d’un tableau Numpy) est aussi passée en argument dans cette fonction. Elle est inutile dans l’activité de base qui consiste à faire reconnaitre le centre à partir de l’image seuillée. Elle a été ajoutée pour permettre d’autres activités de reconnaissance directement à partir de l’image brute, basées sur d’autres techniques (tracking, reconnaissance de formes…).*

## Programmation de la partie Elaboration de la consigne

On accède à cette fonction en choisissant la structure d’asservissement « Personnalisé » et en cliquant sur le bloc Python.

L’élève définit la fonction **Elaboration\_consigne**. Les arguments sont imposés ainsi que le type de sortie.

Cette fonction est appelée à chaque tour de boucle du programme général (qui gère l’affichage, la capture d’image par la caméra, le traitement, la communication avec arduino…). L’élève n’a pas à programmer la boucle générale.

Arguments de la fonction

|  |  |  |
| --- | --- | --- |
| **Nom** | **Description** | **Type** |
| y | Position verticale en pixel du centre de la forme reconnue, l’origine est le coin supérieur haut | *flottant* |
| x | Position horizontale en pixel du centre de la forme reconnue, l’origine est le coin supéreur haut | *flottant* |
| Olddepl\_y | Ecart en y du précédent tour de boucle | *flottant* |
| Olddepl\_x | Ecart en x du précédent tour de boucle | *flottant* |
| hautIm | Hauteur de l’image en pixel | *entier* |
| largIm | Largeur de l’image en pixel | *entier* |
| t1 | Instant mesuré au dernier tour de boucle | *flottant* |
| PID | Liste des 3 valeurs de coefficient PID : PID=[P,I,D] | *Liste de 3 flottants* |
| PitchSum | Somme des consignes en tangage depuis le début de la mesure | *flottant* |
| RollSum | Somme des consignes en roulis depuis le début de la mesure | *flottant* |
| anglePitch | Angle de tangage en millidegrés retourné par la plateforme inertielle | *flottant* |
| angleRoll | Angle de roulis en millidegrés retourné par la plateforme inertielle | *flottant* |

Sorties

|  |  |  |
| --- | --- | --- |
| **Nom** | **Description** | **Type** |
| Pitch | Consigne de tangage en millidegrés envoyée à la carte Arduino | *flottant* |
| Roll | Consigne de roulis en millidegrés envoyée à la carte Arduino | *flottant* |
| t2 | Instant courant (argument d’entrée du tour de boucle suivant) | *flottant* |
| depl\_y | Ecart en y | *flottant* |
| depl\_x | Ecart en x | *flottant* |
| PitchSum | Somme des consignes en tangage depuis le début de la mesure | *flottant* |
| RollSum | Somme des consignes en roulis depuis le début de la mesure | *flottant* |

Un corrigé possible est disponible dans le fichier ***« algorithmes\_perso\_cons\_cor »*** du répertoire :

**« C:\Program Files (x86)\Nacelle\_NC10\_Video\Exe\_VR »** (*répertoire d’installation*).

# Modification des paramètres externalisés.

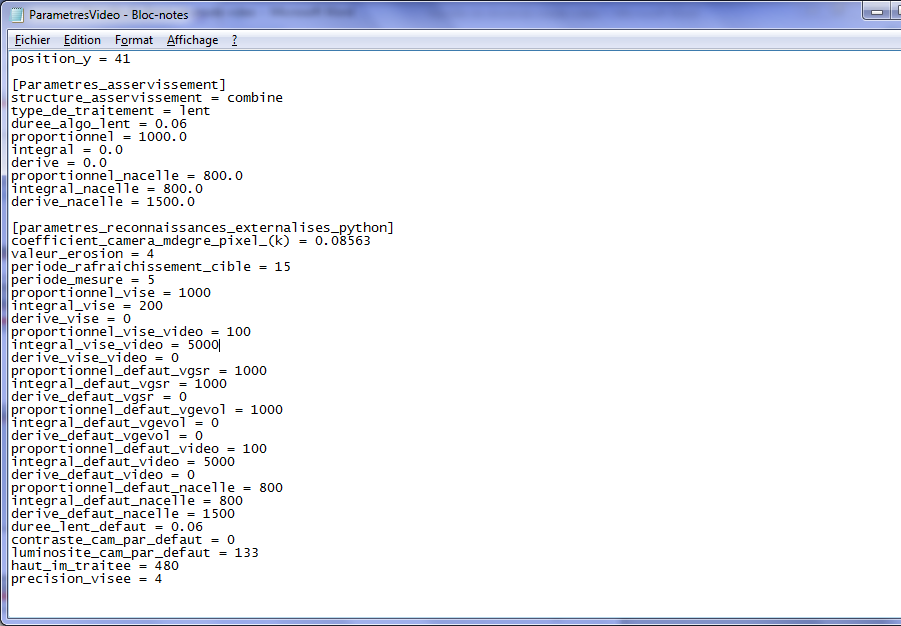
Pour apporter plus de souplesse et pouvoir s’adapter au mieux à différents environnement de travail et besoins, certains paramètres du programme ont été externalisés.

Le professeur peut les modifier, pour tenter d’optimiser le fonctionnement.

*Les modifications de ces paramètres ont évidemment des conséquences importantes sur le comportement et il est conseillé d’agir avec parcimonie. En cas de fausse manipulation les valeurs par défaut sont données dans les tableaux ci-dessous.*

Ces paramètres se situent dans le fichier ***« ParametresVideo.ini »*** du répertoire :

**« C:\Program Files (x86)\Nacelle\_NC10\_Video\Exe\_VR »** (*répertoire d’installation*).



**Zone [parametres\_reconnaissances\_externalises\_python]**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nom du paramètre** | **Effet** | **Valeur** | **Type** |
| coefficient\_camera\_mdegre\_pixel\_(k) | Valeur du coefficient Kadapt (cf par 2.3) | 0.0856 | float |
| valeur\_erosion | Modifie la valeur du coefficient de dilatation/érosion. Revient à la taille minimale des poussières qui sont éliminées dans l’image seuillée. On peut diminuer ce coefficient pour travailler avec cibles très petites et ainsi éviter la suppression de la cible dans l’image seuillée. | 4 | int |
| periode\_rafraichissement\_cible | L’image de projection de la cible est rafraichie de manière régulière. Il s’agit du temps (en millisecondes) entre 2 rafraichissements.  La modification de ce paramètre peut avoir une influence sur la fluidité du mouvement de la cible. Il peut arriver que si ce temps est trop court, la cible reste complétement immobile car l’image n’a pas le temps de s’afficher entre 2 rafraichissements. | 15 | float |
| periode\_mesure | Temps d’attente entre 2 prises de mesures. Ce temps est limité par le bas par la lecture des angles via la carte Arduino. | 5 | float |
| proportionnel\_vise | Valeur du coef. proportionnel du PID utilisé lors de la visée (appui sur bouton Vise) | 1000 | float |
| integral\_vise | Valeur du coef.i intégral du PID utilisé lors de la visée (appui sur bouton Vise) | 200 | float |
| derive\_vise | Valeur du coef. dérivé du PID utilisé lors de la visée (appui sur bouton Vise) | 0 | float |
| proportionnel\_vise\_video | Valeur du coef. proportionnel du PID utilisé lors de la visée en mode « **avec Roll »** | 100 | float |
| integral\_vise\_video | Valeur du coef.i intégral du PID utilisé lors de la visée en mode « **avec Roll »** | 5000 | float |
| derive\_vise\_video | Valeur du coef. dérivé du PID utilisé lors de la visée en mode « **avec Roll »** | 0 | float |
| proportionnel\_defaut\_vgsr | Valeur du coef. proportionnel du PID utilisé par défaut en mode VIDEO + GYRO sans référence | 1000 | float |
| integral\_defaut\_vgsr | Valeur du coef. intégral du PID utilisé par défaut en mode VIDEO + GYRO sans référence | 1000 | float |
| derive\_defaut\_vgsr | Valeur du coef. dérivé du PID utilisé par défaut en mode «VIDEO + GYRO sans référence» | 0 | float |
| proportionnel\_defaut\_vgevol | Valeur du coef. proportionnel du PID utilisé par défaut en mode VIDEO + GYRO évolué | 1000 | float |
| integral\_defaut\_vgevol | Valeur du coef. intégral du PID utilisé par défaut en mode VIDEO + GYRO évolué | 0 | float |
| dérive\_defaut\_vgevol | Valeur du coef. dérivé du PID utilisé par défaut en mode VIDEO + GYRO évolué | 0 | float |
| proportionnel\_defaut\_video | Valeur du coef. proportionnel du PID utilisé par défaut en mode « VIDEO » | 100 | float |
| integral\_defaut\_video | Valeur du coef. intégral du PID utilisé par défaut en mode « VIDEO » | 5000 | float |
| dérive\_defaut\_video | Valeur du coef. dérivé du PID utilisé par défaut en mode « VIDEO » | 0 | float |
| proportionnel\_defaut\_nacelle | Valeur du coef. proportionnel du PID de l’asservissement de la nacelle seule (carte Arduino) | 800 | float |
| integral\_defaut\_nacelle | Valeur du coef. intégral du PID de l’asservissement de la nacelle seule (carte Arduino) | 800 | float |
| derive\_defaut\_nacelle | Valeur du coef. dérivé du PID de l’asservissement de la nacelle seule (carte Arduino) | 1500 | float |
| duree\_lent\_defaut | Durée par défaut de l’algorithme lent | 0.06 | float |
| contraste\_cam\_par\_defaut | Valeur prise par le paramètre de luminosité lors du clic sur « valeurs par défaut » | 0 | float |
| luminosite\_cam\_par\_defaut | Valeur prise par le paramètre de luminosité lors du clic sur « valeurs par défaut » | 133 | float |
| haut\_im\_traitée | Hauteur en pixel de l’image traitée. Si la taille de l’image fournie par la caméra est plus grande, celle-ci est redimensionnée. | 480 | float |
| precision\_visee | Valeur utilisée pour tester la convergence de la visée (condition d’arrêt de la visée suite à appui sur le bouton vise).  Si cette valeur est trop grande, la visée peut s’arrêter alors que la cible n’est pas complétement au centre. En diminuant trop cette valeur on risque de ne pas atteindre le critère de convergence et d’avoir un temps de visée très long. | 4 | float |