# MegaUAV 2013

June 14, 2013

# Contents

# 1 Introduction

BLABLABLA BLABLA

# 2 Partie 1

## 2.1 Sous partie 1

## 2.2 Sous partie 2

# 3 Communication

## 3.1 Architecture des communications

Nous avons deux cartes : une Microkopter, la Flight-Ctrl, qui est la carte du drone et un Gumstix sur lequel notre programme fonctionne. La communication

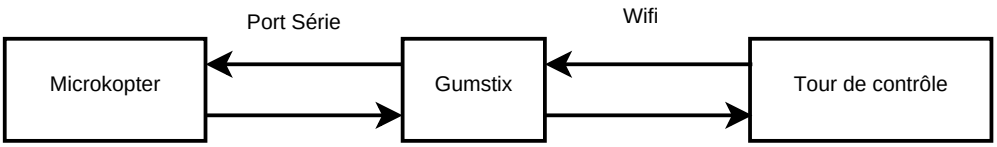


Figure 1: Architecture de la communication

entre les deux cartes se fait par port série sur

```
/dev/ttyUSB0
```

Le programme gpsd va analyser et convertir les infos du capteur gps en chaine de caractères.
les infos GPS sont récupérées directement sur la gumstix via le port série:

```
/dev/ttyS0
```

## 3.2 Communication série

### 3.2.1 Série inter carte

### 3.2.2 GPS

la trame gps sur port série est lisible via cette commande :

```
cat /etc/ttyS0
```

En cas de problème avec la trame GPS,tapez cette commande :

```
killall gpsd; gpsd /dev/ttyS0
```

Notre programme va aller lire dans ce descripteur pour récupérer la trame GPGGA.
Détail:

**Trame GGA**

Nous utiliserons la trame identifiée GGA dans nos algorithmes dont la spécification est donnée ci-dessous:
$GPGGA,hhmmss.ss,llll.ll,a,yyyyy.yy,a,x,xx,x.x,x.x,M,x.x,M,x.x,xxxx

| Name | Example | Data Description |
|---|---|---|
| Sentence Identifier | GPGGA | Global Positioning System |
| Time | 170834 | 17:08:34 Z |
| Latitude | 4124.8963, N | 41d 24.8963' N or 41d 24' 54" N |
| Longitude | 08151.6838, W | 81d 51.6838' W or 81d 51' 41" W |
| Fix Quality:(0,1,2) | 1 | Data is from a GPS fix |
| Number of Satellites | 05 | 5 Satellites are in view |
| Précision horizontale | 1.5 | Accuracy of horizontal position |
| Altitude | 280.2, M | 280.2 meters above mean sea level |
| Height of geoid above WGS84 ellipsoid | -34.0, M | -34.0 meters |
| Time since last DGPS update | blank | No last update |
| DGPS reference station id | blank | No station id |
| Checksum | *75 | Checksum |

Figure 2: Trame GPGGA

Nous gardons la longitude, la latitude, le nombre de satellites , la précision horizontale et l'altitude. Exemple d'une trame GPGGA sur le port série:
`GPGGA,11373.00,4902.59765,N,00205.00247,E,1,09,1.07,73.1,M,46.0,M,,*65`

## 3.3 Réseau

Afin d'assurer la communication entre le drone et la tour de controle il a été nécéssaire de développer un protocole de communication appelé MUAVCOM
On distingue deux canaux:

- un canal où la source d'émission est le drone, il envoie ses informations en continu à la tour de contrôle (batterie, accéléromètre, état, position gps, image de la caméra etc ...).
  La tour de controle envoie un acquittement pour certaines infos critiques.
- un canal de pilotage dans lequel la tour de contrôle va envoyer des ordres : pour changer le comportement du drone, passer du mode manuel au mode auto, choisir de l'ia à utiliser, envoyer des commande de vol, etc ...

Le protocole est construit comme ceci:

- une entête dans laquelle sera renseigné l'id du drone, l'id de la flotte du drone
- le type de requête et un code d'erreur si besoin, ensuite vient les informations de la requete en question

Ci-dessous la liste des requetes :

- PILOTE_REQ_MANUAL: on demande au drone de passer en mode manuel
- R_PILOTE_REQ_MANUAL: acquitement du passage au mode manuel
- SEND_INFO: envoi des informations du drone
- R_SEND_INFO: acquitement de reception des infos
- EMERGENCY: (non utilisé) indication que le drone se trouve en situation critique (ex: plus de batterie)
- MISSION: non utilisé, à définir
- PILOTE_MANUAL: envoi des informations de pilotage manuel
- PILOTE_REQ_AUTO: on demande au drone de passer en mode automatique
- R_PILOTE_REQ_AUTO: acquitemment du passage au mode auto
- PILOTE_REQ_OFF: on demande au drone de passer dans l'état inactif
- R_PILOTE_REQ_OFF: acquitement de cette état
- SEND_IMG_SIZE: on envoi la taille de l'image que la caméra capture
- R_SEND_IMG_SIZE: acquitement de la reception de cette taille
- SEND_IMG: envoi d'une partie d'image
- SEND_GPS_INFO: envoi des information GPS
- HELLO: message envoyer à la tour de controle pour indiquer la mise en marche
- R_HELLO: réponse de la tour de controle au hello
- R_SEND_IMG: acquitement de la reception de l'image
- GPS_INFO_START: on demande au drone d'envoyer les infos
- GPS_INFO_STOP: on demande au drone d'arreter d'envoyer les info gps

- IMAGE_SEND_START: on demande l'envoi des images capturés par la caméra
- IMAGE_SEND_STOP: on demande d'areté d'envoyer l'image.

Datagramme MUAVCOM :
`|IP|UDP|FLOTTE_ID[INT]|DRONE_ID[INT]|TYPE_R[INT]|TIMESTAMP[INT][INT]|ERROR[INT]|DATA|`

l'entête muavcom commence à FLOTTE_ID, et se termine à ERROR, elle fait 24 octets.
Détail:
`-image TYPE\_R = SEND\_IMG`

`|ID\_PART[BYTE]|IMAGE\_PART[BYTE]|`

A cause du MTU par défaut (maximum taille des packets accepté) limité à 1500 octets, de la plupart des cartes réseaux,nous sommes dans l'obligation de découper l'image. Les images capturées font environ 25ko que nous envoyons en 25 parties.
Dans les données nous avons le numéro de partie de l'image suivi de la partie elle même.
`TYPE\_R = SEND\_GPS\_INFO`

`|latitude:longitude:nombre de satellite:précision:altitude|`

Les données GPS sont envoyées sous forme de chaines de caractères. On envoie seulement la latitude, la longitude, le nombre de satellites captés, la précision de la position horizontale et l'altitude.
`-info microkoptère TYPE\_R = SEND\_INFO`

Les informations de la carte microkopter sont envoyées dans un tableau de 32 int. Ce tableau, contient de niveau de batterie, l'état du quadri, les erreurs éventuelles, etc ...
`-pilotage TYPE\_R = PILOTE\_MANUAL`

`|NICK[INT]|ROLL[INT]|YAW[INT]|GAS[INT]|`

Lorsque que l'on pilote le quadri à l'aide du joystick,on envoie les infos qui vont indiquer la puissance des moteurs.
`-ia TYPE\_R = PILOTE\_REQ\_AUTO`

`|TYPE|`

Lorsque l'on passe en mode automatique, on indique quel type d'IA on a choisi.

# 4 Vol Stationnaire

Pendant le vol, le quadricoptère est soumis à diverses forces extérieures perturbant la stabilité de ses mouvements, tels que la puissance des moteurs et le poids de l'appareil non homogèneiquement répartis, le retour de force lié au brassage d'air des hélices, etc...
Nous cherchons donc dans un premier temps à compenser les déplacements non intentionnels de notre appareil afin d'effectuer un vol stationnaire.

## 4.1 Algorithme

Notre algorithme est basé sur la vision, obtenue grace à une caméra reliée au Gumstix. Un traitement sur l'image servira à extraire des points d'interêt qui serviront de référentiel au sol dont nous suivront les positions les coordonnées par rapport au temps. Ces coordonnées serviront à déterminer les mouvements de l'appareil par rapport à ces points au sol, sous forme de vecteurs, qui serviront ensuite à déterminer la compensation à effectuer afin de limiter les déplacements de l'appareil dans l'espace.

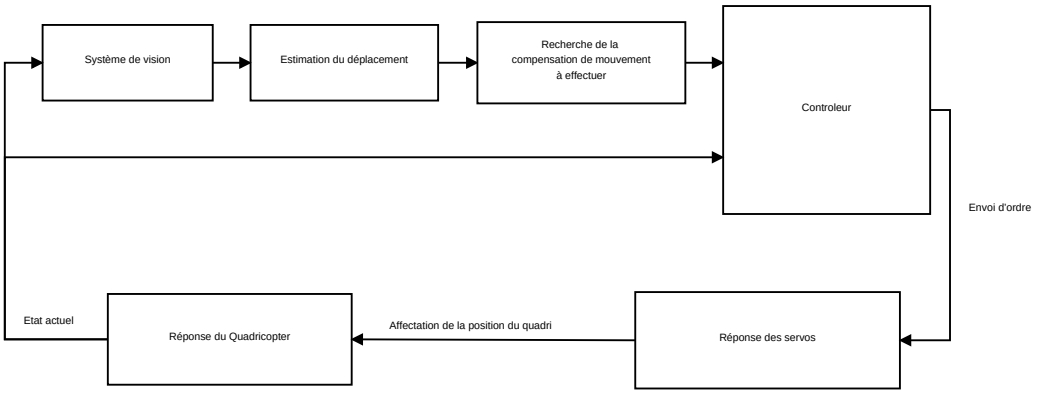


Figure 3: Algorithme

## 4.2 Estimation du déplacement

Pour estimer la direction de déplacement, l'algorithme se base sur la transformation affine de points d'interêt entre deux images consécutives.

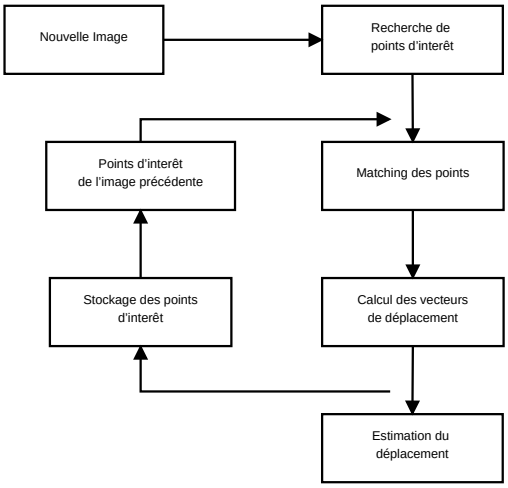


Figure 4: Traitement Image

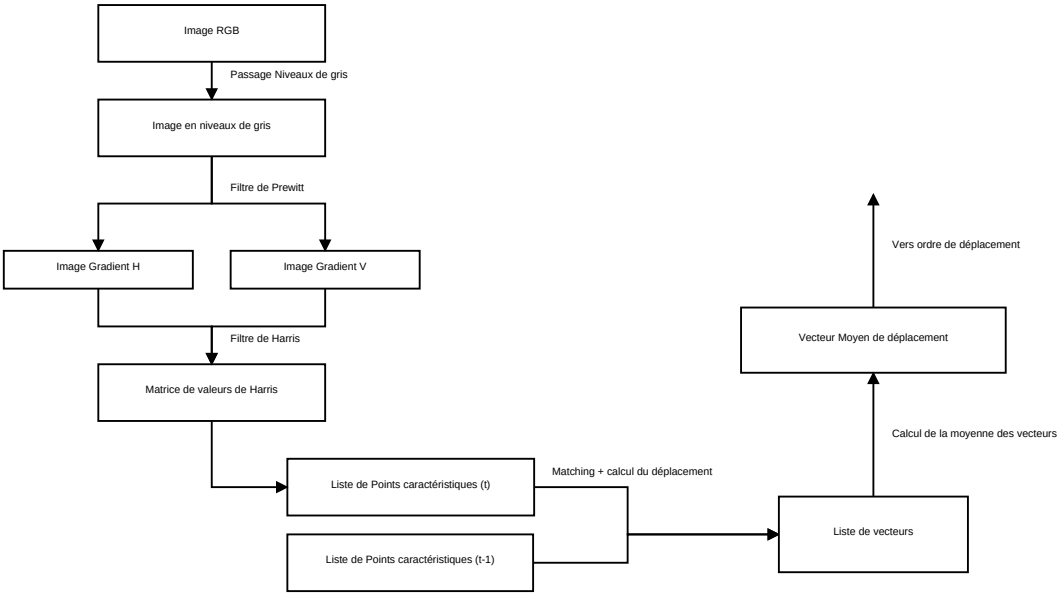


Figure 5: Traitement Image

## 4.3 Extraction des points d'interêt

### 4.3.1 Détection des contours

Nous ne cherchons pas à proprement dit à détecter les contours mais nous avons besoin de calculer les gradients horizontal et vertical de l'image pour la détection

des points d'interêt.

#### 4.3.2 Détection des coins

Nous avons choisi une détection de points d'interêts par détections de coins (corner detection). Nous avons choisi d'implémenter 2 méthodes:

- Le détecteur de Harris
- L'algorithme FAST

#### 4.3.3 Harris

$$PrewittH \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

$$PrewittV \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

#### 4.3.4 FAST

### 4.4 Matching-Tracking

### 4.5 Compensation du mouvement

## 5 Conclusion