Le rapport intermédiaire de projet fin d'étude

Commande en position d'un drone Parrot Bebop 2

Tuteur d'école : Monsieur Komi Midzodzi PEKPE

Binômes : Lijie YAO & Lirui ZHANG

I- L'introduction

Le but de notre projet est réalisé la commande en position d'un drone Parrot Bebop 2.
Comme un drone peut décoller, atterrir, avancer, reculer, pivoter, basculer et prendre des photographies aériennes, nous espérons mettre en œuvre un contrôle cinématique du drone. Il peut s'agir d'un contrôle de la position spatiale du drone, d'un contrôle basé sur le traitement d'image ou d'un contrôle de vitesse du drone. Il existe de nombreuses possibilités pour ce contrôle de la cinématique, nous étudions les fonctionnalités qui nous intéressent et essayons de mettre en œuvre cette fonction.
Dans la première moitié de nos études, nous avons mené des recherches sur le contrôle de la position spatiale des drones sur la base du traitement des images.
Pour la seconde moitié du travail, nous serons plus intéressés par le contrôle de la vitesse de rotation du drone.

II- Le processeur

II.1- La recherche de drone

Tout d'abord, nous avons utilisé une application mobile qui s'appelle FlightFree Pro pour faire quelques tests de fonction, afin de comprendre spécifiquement le fonctionnement de ce drone.

Ensuite, nous avons faire une étude des capteurs et des caméras du drone. Nous avons trouvé les informations suivantes, mais il n'existe pas de modèle spécifique du capteur utilisé par le drone.

- Une caméra verticale qui permet le maintien d'un point fixe.
- Un capteur ultrason qui analyse l'altitude de vol jusqu'à 5 mètres.
- Un gyroscope 3 axes qui permet de calculer l'angle d'inclinaison de l'appareil.
- Un magnétomètre 3 axes qui donne la possibilité de définir la position du drone, à l'image d'une boussole.
- Un GPS et un GLONASS pour la géolocalisation du drone et aide à mesurer la vitesse de drone pour plus de stabilité à haute altitude.
- Un capteur de pression qui permet de mesurer la pression et de calculer l'altitude de vol lorsque celle-ci dépasse les 5 mètres.
- Un accéléromètre qui permet de mesurer l'orientation du drone sur 3 axes et sa vitesse linéaire.

Et après, nous avons étudié le système de contrôle du drone. Le système d'exploitation utilisé dans ce drone est le ROS Kinetic, la méthode de communication entre l'ordinateur ou le téléphone portable et le drone est par wifi, mais la méthode de communication entre la manette de commande et le drone est par antenne radioélectrique.

II.2- Les études d'environnement de travail, de logiciels et d'interface.

Selon nos recherches précédentes, la méthode de communication entre l'ordinateur ou le téléphone portable et le drone est par wifi. Au début, nous avons utilisé la machine virtuelle, mais malheureusement, la machine virtuelle ne pouvait pas appeler le wifi. Parce que la carte réseau sans fil ne peut pas être ajoutée dans la machine virtuelle VMware. Le lien d'internet entre la machine virtuelle et le Windows est par la méthode NAT. Alors plus tard, nous avons utilisé un double système.
Selon nos recherches précédentes, le système d'exploitation utilisé par ce drone est le ROS. Par conséquent, pour obtenir un contrôle efficace, nous devons d'abord configurer le ROS.

Le ROS est une système exploitation de la robotique qui basé sur Linux. Puisque différentes versions du système d'exploitation Linux correspondent à différentes versions de ROS et que le système d'exploitation utilisé par nos drones est ROS Kinetic, nous devrions utiliser Ubuntu 16.04. Dans

l'environnement ROS, nous pouvons mettre en œuvre une communication avec des drones par publier des sujets ou des messages. Et nous pouvons également de recevoir de l'information qui vient de drone, par exemple, le vidéo. Après avoir configuré l'environnement de travail ROS, nous sommes prêts à installer un logiciel de kit de développement officiel pour le drone, qui s'appelle SDK Parrot.

Parrot
For Developers

SDK Parrot nous aide à connecter, à piloter, à recevoir des flux, à enregistrer et à télécharger des médias (photo et vidéo), à envoyer et à lire des plans de vol sur pilote automatique et à mettre à jour notre drone. Nous n'avons pas fait beaucoup de recherche plus approfondie sur le SDK, car nous avons également installé Bebop_autonomy, qui est développé basé sur le SDK Parrot. Bebop_autonomy est un drive, ça peut être utilisé pour lancer le pilote, envoyer de commandes à Bebop, lecture de Bebop, configuration de Bebop et du pilote. Mais il y a un problème de Bebop_autonomy, nous tapons la commande en temps réel dans le terminal et chaque fois que nous tapons une instruction, le drone exécutera cette commande de manière continue. Par exemple, lorsque nous tapons l'instruction d'avancer, le drone continue d'avancer jusqu'à ce que nous tapions la commande à arrêter. Cependant, en contrôle réel, il n'est pas possible de taper des commandes dans le terminal en temps réel. Une idée pour résoudre ce problème est de créer un fichier, et puis contrôler le drone pas ce fichier, et ce fichier partage des données entre le drone et l'ordinateur.
Afin de transférer les données à travers le fichier et de simplifier la programmation, nous avons décidé d'utiliser python comme langage de programmation pour contrôler le drone.

Afin d'installer l'interface de python pour Bebop, nous avons d'abord configuré l'environnement pour python, nous avons choisi Anaconda, qui est une package open source. Et puis nous avons installé de Pyparrot, c'est l'interface de python pour Bebop.
En utilisant un exemple de Pyparrot, nous avons configuré que la connexion entre le drone et l'ordinateur a marché bien. Nous avons ensuite terminé les tests de contrôle du décollage, de l'atterrissage, de l'avancée et du recul du drone en appelant certaines fonctions de la bibliothèque. Et puis nous avons terminé l'utilisation de la caméra horizontale du drone pour prendre des photos par frame et les transférer à l'ordinateur en temps réel. Comme prévu à l'origine,

nous devrions terminer le traitement des images prises par le drone. Afin de faire le traitement d'image, nous avons utilisé une bibliothèque qui s'appelle OpenCV. C'est une bibliothèque gratuite et open source, il y a certaines fonctions pour traiter des graphiques, mais il n'y a pas d'algorithme qui s'appelle SIFT que nous voulions l'utiliser, il faut le réaliser nous-même.

Le SIFT (La scale-invariant feature transform), que l'on peut traduire par « transformation de caractéristiques visuelles invariante à l'échelle », est un algorithme utilisé dans le domaine de la vision par ordinateur pour détecter et identifier les éléments similaires entre différentes images numériques.
Nous avons appris les principes de l'algorithme SIFT. Pour identifier les éléments similaires entre les deux photos, tout d'abord de détecter sur l'image des zones circulaires « intéressantes », centrées autour d'un point-clé et de rayon déterminé appelé facteur d'échelle. Celles-ci sont caractérisées par leur unité visuelle et correspondent en général à des éléments distincts sur l'image. Sur chacune d'elles, on détermine une orientation intrinsèque qui sert de base à la construction d'un histogramme des orientations locales des contours, habilement pondéré, seuillé et normalisé pour plus de stabilité. C'est cet histogramme qui sous la forme d'un vecteur à 128 dimensions (ou valeurs) constitue le descripteur SIFT du point-clé, et l'ensemble des descripteurs d'une image établissent ainsi une véritable signature numérique du contenu de celle-ci.

II.3- Les commandes de drone

Dans la première moitié de nos études, nous avons mené des recherches sur le contrôle de la position spatiale des drones sur la base du traitement des images.
Nous avons codé un petit programme en utilisant python, dans le travail environnement de ROS, utilisant la bibliothèque graphique de l'OpenCV.
Ce programme réalise les fonctions de connecter avec le drone lui-même, si la télécommunication entre le drone et l'ordinateur n'a aucun de problème, il va démarrer le camera horizontal, pour retourner des vidéos à l'ordinateur.
Après notre ordinateur reçoit des vidéos, il va mettre chaque frame de vidéo dans un fichier, en format 'png', et mettre la propriété de vidéo dans un document en format 'sdp' en même temps.
Si tous les process avant sont réussi, le drone va décoller, son altitude n'est pas supérieure à 5m, et rester en l'air dans une position pour un instant (ça peut être modifié), et puis il va avancer pour une distance avec la vitesse maximale de 1m/s, et puis il reste dans une position pour un instant, après il va

reculer la même distance qu'il est avancé, et reste l'air pour un instant.
A la fin, il va descendre, arrêter de mettre des photos, atteindre le camera horizontal, retourner le pourcentage d'électricité reste du batterie et disconnecter avec l'ordinateur.
Le code est comme ci-dessous :

```
from pyparrot.Bebop import Bebop
from pyparrot.DroneVision import DroneVision
import threading
import cv2
import time

isAlive = False

class UserVision:
    def __init__(self, vision):
        self.index = 0
        self.vision = vision

    def save_pictures(self, args):
        #print("saving picture")
        img = self.vision.get_latest_valid_picture()

        if (img is not None):
            filename = "test_image_%06d.png" % self.index
            #cv2.imwrite(filename, img)
            self.index +=1


bebop = Bebop()

print("connecting")
success = bebop.connect(10)
print(success)

if (success):
    print("turning on the video")
    bebop.start_video_stream()

    print("sleeping")
    bebop.smart_sleep(5)

    bebop.ask_for_state_update()

    bebopVision = DroneVision(bebop, is_bebop=True)

    userVision = UserVision(bebopVision)
    bebopVision.set_user_callback_function(userVision.save_pictures, user_callback_args=None)
    success = bebopVision.open_video()

    if (success):
        print("Vision successfully started!")

    bebop.safe_takeoff(10)

    bebop.set_max_tilt(5)
    bebop.set_max_vertical_speed(1)
```

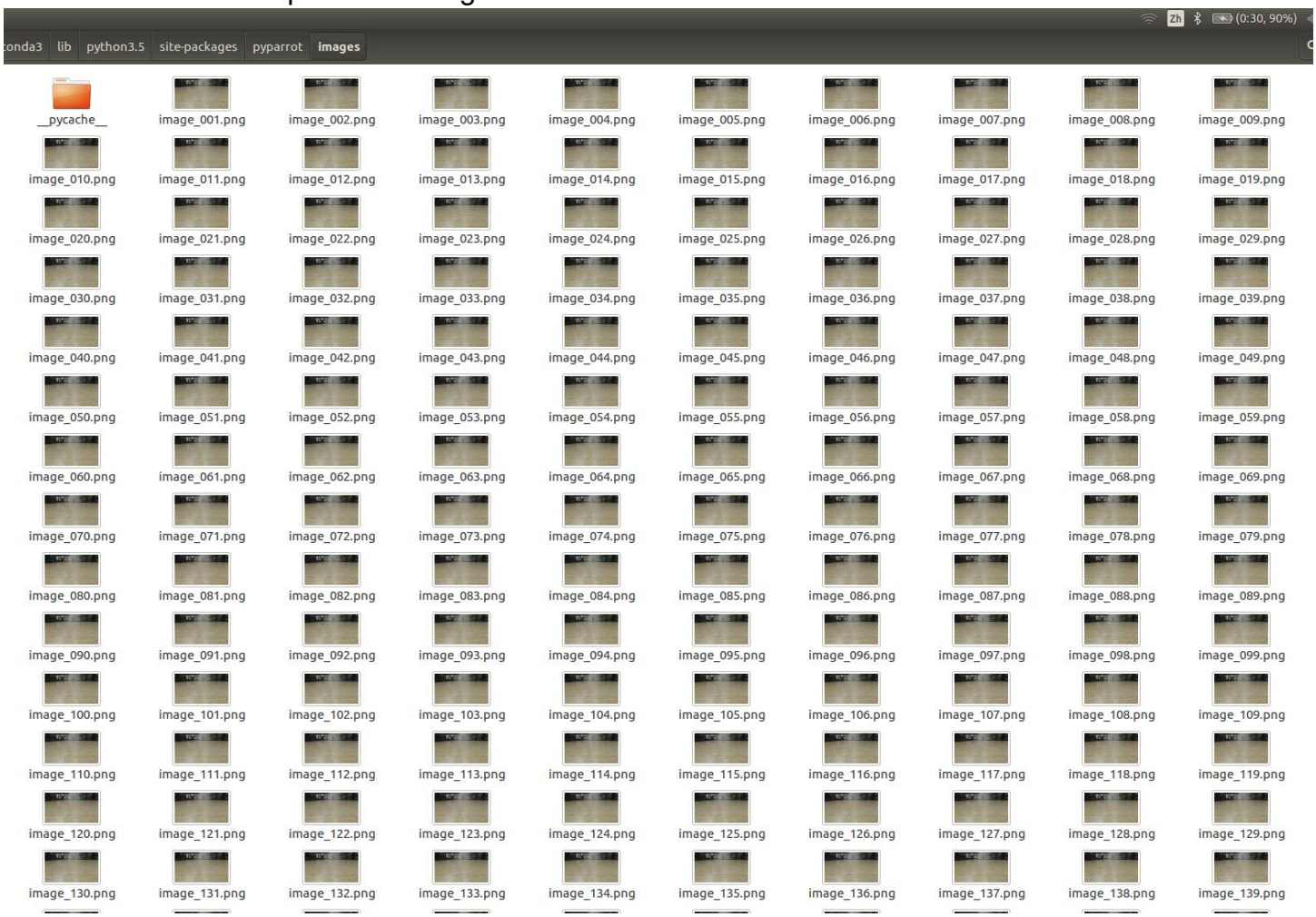
```
    print("Flying direct: going forward (positive pitch)")
    bebop.fly_direct(roll=0, pitch=50, yaw=0, vertical_movement=0, duration=1)

    bebop.smart_sleep(2)

    print("Flying direct: going backwards (negative pitch)")
    bebop.fly_direct(roll=0, pitch=-50, yaw=0, vertical_movement=0, duration=1)


    bebop.smart_sleep(2)
    bebop.safe_land(10)

    print("Finishing demo and stopping vision")
    bebopVision.close_video()

    print("DONE - disconnecting")
    bebop.stop_video_stream()
    bebop.smart_sleep(5)
    print(bebop.sensors.battery)
    bebop.disconnect()
```

Python Tab Width: 8 Ln 39, Col 33 INS

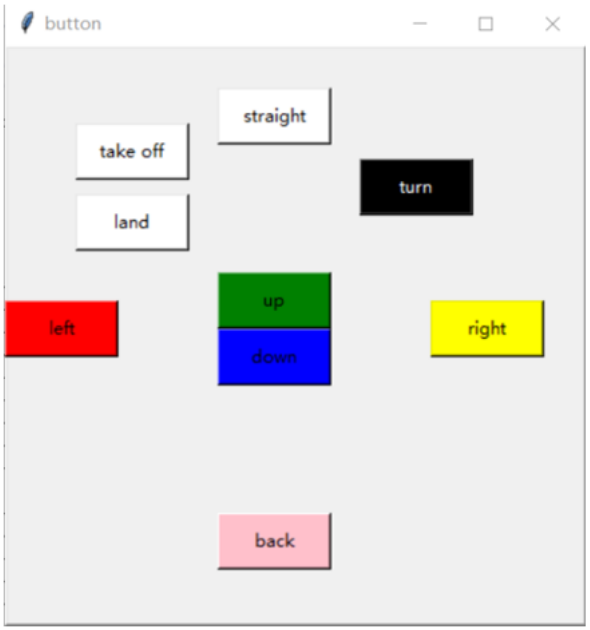
Le résultat après lancer est comme ci-dessous :

```
parrotbebop2@parrotbebop2-GL502VML: ~/pyparrot/examples
connecting
Setting up mDNS listener since this is not a Mambo
Making a browser for _arsdk-090c._udp.local.
Service Bebop2-139278._arsdk-090c._udp.local. added, service info: ServiceInfo(type='_arsdk-090c._udp.local.', name='Bebop2-139278._arsdk-090c._udp.local.', address=b'\xc0\xa8*\x01', port=44444, weight=0,
 priority=0, server='Bebop2-139278.local.', properties={b'{"device_id":"PI040384AH7J139278"}': False})
{"arstream2_client_control_port": 55005, "controller_type": "computer", "arstream2_client_stream_port": 55004, "d2c_port": 43210, "controller_name": "pyparrot"}
{'arstream2_server_stream_port': 5004, 'status': 0, 'c2d_port': 54321, 'arstream2_server_control_port': 5005, 'c2d_user_port': 21, 'c2d_update_port': 51, 'qos_mode': 0}
c2d_port is 54321
starting listening at
 Success in setting up the wifi network to the drone!
True
turning on the video
sleeping
 Could not find sensor in list - ignoring for now.  Packet info below.
(144, 0, 2)
 Could not find sensor in list - ignoring for now.  Packet info below.
(142, 0, 3)
 Could not find sensor in list - ignoring for now.  Packet info below.
(134, 0, 3)
 Could not find sensor in list - ignoring for now.  Packet info below.
(134, 0, 4)
 Could not find sensor in list - ignoring for now.  Packet info below.
(134, 0, 4)
 Could not find sensor in list - ignoring for now.  Packet info below.
(134, 0, 4)
 Error: tried to parse a bad sensor packet
 Could not find sensor in list - ignoring for now.  Packet info below.
(144, 0, 1)
65
/home/parrotbebop2/anaconda3/lib/python3.5/site-packages/pyparrot/images
/home/parrotbebop2/anaconda3/lib/python3.5/site-packages/pyparrot/utils
Opening ffmpeg
ffmpeg -protocol_whitelist "file,rtp,udp" -i /home/parrotbebop2/anaconda3/lib/python3.5/site-packages/pyparrot/utils/bebop.sdp -r 30 image_%03d.png &
starting vision thread
Opening non-blocking readers
ffmpeg version 3.4.4-1~16.04.york0 Copyright (c) 2000-2018 the FFmpeg developers

  built with gcc 5.4.0 (Ubuntu 5.4.0-6ubuntu1~16.04.10) 20160609

  configuration: --prefix=/usr --extra-version='1~16.04.york0' --toolchain=hardened --libdir=/usr/lib/x86_64-linux-gnu --incdir=/usr/include/x86_64-linux-gnu --enable-gpl --disable-stripping --enable-avre
sample --enable-avisynth --enable-gnutls --enable-ladspa --enable-libass --enable-libbluray --enable-libbs2b --enable-libcaca --enable-libcdio --enable-libflite --enable-libfontconfig --enable-libfreetype
 --enable-libfribidi --enable-libgme --enable-libgsm --enable-libmp3lame --enable-libmysofa --enable-libopenjpeg --enable-libopenmpt --enable-libopus --enable-libpulse --enable-librubberband --enable-libr
svg --enable-libshine --enable-libsnappy --enable-libsoxr --enable-libspeex --enable-libssh --enable-libtheora --enable-libtwolame --enable-libvorbis --enable-libvpx --enable-libwavpack --enable-libwebp -
-enable-libx265 --enable-libxml2 --enable-libxvid --enable-libzmq --enable-libzvbi --enable-omx --enable-openal --enable-opengl --enable-sdl2 --enable-libdc1394 --enable-libdrm --enable-libiec61883 --enab
le-chromaprint --enable-frei0r --enable-libopencv --enable-libx264 --enable-shared

  libavutil      55. 78.100 / 55. 78.100
  libavcodec     57.107.100 / 57.107.100
  libavformat    57. 83.100 / 57. 83.100
  libavdevice    57. 10.100 / 57. 10.100
  libavfilter     6.107.100 /  6.107.100
  libavresample   3.  7.  0 /  3.  7.  0
  libswscale      4.  8.100 /  4.  8.100
  libswresample   2.  9.100 /  2.  9.100
```

Et les photos sauvegardées sont comme ci-dessous :

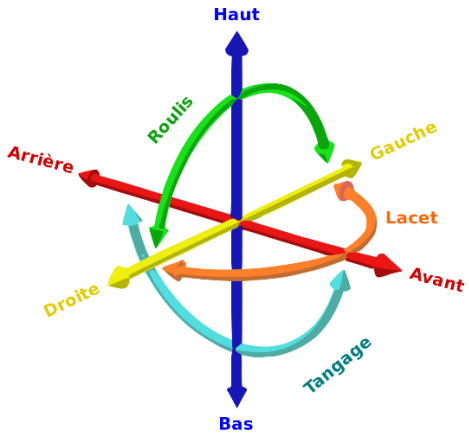


similaires entre les deux, et puis trouver de notre objectif. Nous voulons utiliser un algorithme qui s'appelle SIFT (La scale-invariant feature transform).
Nous étions à l'origine prêts à implémenter l'algorithme en python, cependant, étant donné que l'image doivent être modélisées après le traitement par cette méthode, la position du drone peut être contrôlée avec précision, ce qui prend trop de temps et ne peut pas être appliqué à un apprentissage ultérieur. Donc, pour les travails ensuite, nous sommes plus intéressants à contrôler et réguler la vitesse de ce drone par une interface.
Pour la seconde moitié du travail, nous serons plus intéressés par le contrôle de la vitesse de rotation du drone.
Notre professeur nous a suggéré de créer d'abord une interface d'exploitation afin que nous puissions contrôler nos drones en temps réel à l'aide des boutons sur l'interface. Nous avons donc ajouté une interface sur la base de l'original et nous pouvons la contrôler en temps réel via des boutons.



Nous pouvons utiliser *take off* pour décoller notre drone et *land* pour l'atterrir. *Up* pour le monter et *down* pour le descendre, *left* pour le voler à gauche et *right* pour le voler à droite, *straight* pour l'avancer et *back* pour le reculer. *Turn* pour la rotation de moins 180 dégrée à plus 180 dégrée. Pour réguler des variables, nous le tapons directement dans le terminal, mais dans le futur, il faut améliorer. Il pourra être tapé dans l'interface, et puis sera validé. Nous le lirons, l'envoyions au fichier, ce sera le fichier partagé entre Matlab et le ROS.
Pour l'instant, nous avons déjà réussi de contrôler ce drone pour décoller, atterrir, avancer, reculer, augmenter, descendre, voler à gauche, voler à droite et la rotation.
Nous mettons actuellement en œuvre l'utilisation de boutons pour contrôler la vitesse de rotation de drone et contrôler la distance de vol de drone. Nous devons mettre certaines conditions limitées, par exemple, la hauteur maximum, la vitesse de rotation limite, la vitesse de voler maximum, pour garantir la

sécurité dans le cas de l'utiliser dans une salle.
Nous avons utilisé trois axes comme ci-dessous comme des variables à réguler :



III- Dans le futur

Nous devons améliorer cette interface, ajouter la fonction de taper les variables directement sur l'interface, et puis le lire et l'envoyer en temps réel au fichier. Mais pas seulement ça, nous devons utiliser le fichier, pour partager des données entre Matlab et ROS, comme nous ne pouvons pas taper la commande une par une dans le terminal. Afin de réguler et corriger des données tapées et des datas de drone en temps réel, il faut réaliser d'afficher des datas par une liste en même temps de contrôler.