2-rotor drone addition of sensors for position and height stabilization

DTU special course January 2023

Anton Isak Larsen
s203885

Martin Herwin Schaarup
s203851

***Abstract*—This paper describes the installation and implementation of a ultrasound distance sensor for measuring the height of a two-rotor drone, when it is close to the ground (less than 6m). Furthermore, the drone gets a camera installed for detecting the x- and y-drift, and software on the drone, which takes in the raw images from the camera and outputs the x- and y-drift, will be implemented.**

***Index Terms*—2-rotor drone, ultrasound distance censor, camera.**

I. Introduction

The reason for adding a height sensor is, because the barometer on the drone is not precise enough. This is mostly a problem, when being close to the ground, since the barometers has an accuracy of ±8m, and therefore it could easily hit the ground, when trying to hover.
Adding a camera facing down for detecting x- and y-drift, should make it easier to fix problems with a constant x- and y-drift.

II. Physical installation

A. CAD modeling

For the physical implementation we drew a 3d model of a holder, which could hold both the camera and the distance sensor. See figure 1. There has been a focus on a robust and strong design so that any vibrations of the drone is minimized in the sensors.

This model is 3D printed in PLA, and the sensors are mounted on the button of the model with screws.

The ultra sound distance sensor we have chosen is the LV-MaxSonar-EZ MB1000. The transmit pin from the sensor TX, is connected to the teensy's pin 31 (RX4). With this configuration we can read from the sensor with a serial connection with format RS232.

III. Software implementation

A. Overview

The drone's brain consists of a Teensy 3.5, which computes the low-level flight controller commands and sensor readings. The Teensy then communicates with the Raspberry Pi 4, which does the more compute intensive tasks. It handles the camera and the reference tracking software.

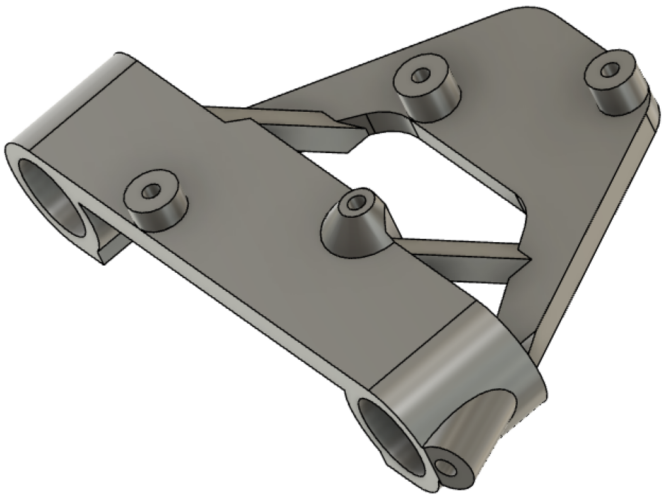


Fig. 1: 3D model of sensor holder.

We have chosen to run a GUI, graphical user interface, on the Raspberry pi. Through WiFi communication the Raspberry Pi can then be accessed with a VNC remote desktop, and everything can then be administered.

B. Ultra sound sensor

The software implementation of the ultrasound sensor, was already done in the code from the previous group. We only made minor adjustments in the code to make it work.

C. Computer vision for detecting drift

For measuring the drift in x- and y-direction, we used a Raspberry Pi 4 running Python with open-cv-3.x.x. The camera is a Raspberry Pi Camera V2.1 that faces downwards opposite to the z-direction. Since we have a mono-camera, we are able to track points in the plane perpendicular to the z-axis using algorithms included in the OpenCV library. The program function as follows:
We have captured a prev_grey image and a curr_grey image. The prev_grey image is searched for the strongest corners to track with the function using the Shi-Tomasi method, and these features are saved as prev_pts. Thereafter the optical flow from the prev_grey to curr_grey with prev_pts as the reference

points are calculated using the Lucas-Kanade with pyramids algorithm. This gives us the curr_pts, which is the continuation of feature points from the prev_pts, but in the current frame. Lastly we compute the optimal affine transformation between prev_pts and curr_pts, where translation, rotation, and uniform scaling is preserved. The output matrix gives the euclidean distance between these sets of points and the rotation angle. With the distance values, drift in the (x,y)-plane can be detected.

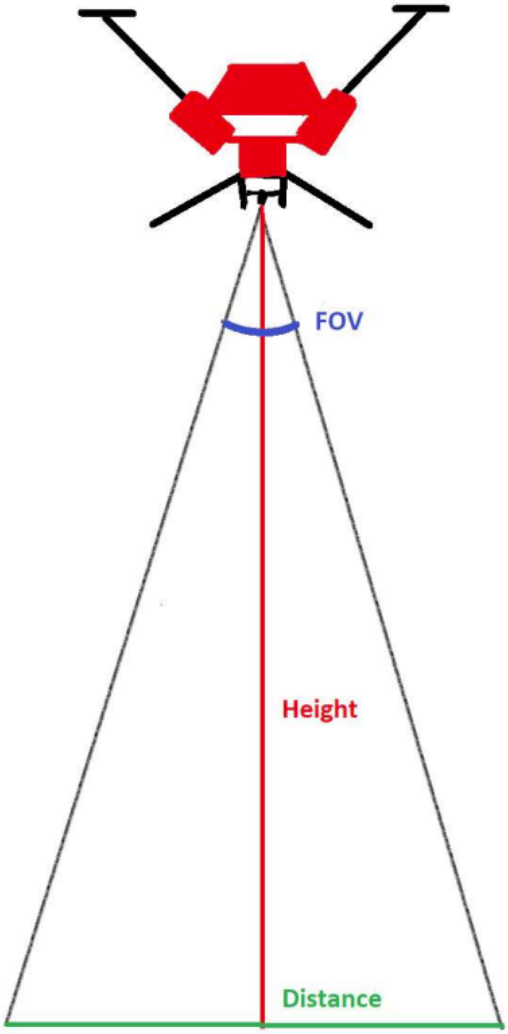
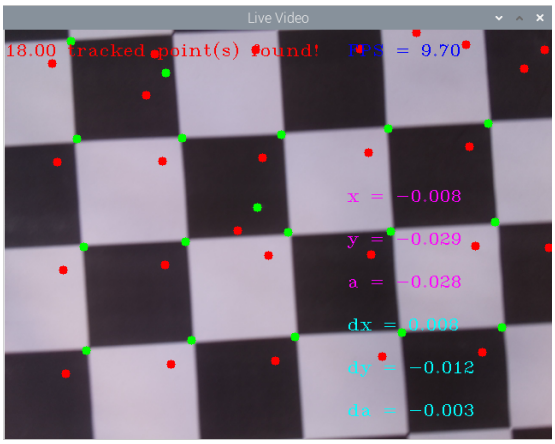


Fig. 2: Visualisation of the camera perspective

However, this drift is defined in pixels making the camera pick up a larger drift the closer the viewpoint of the camera is to the ground. This occurs because of the lack of information when projecting a 3D-environment onto a 2D-environment. By obtaining the height/distance to ground with the ultrasound-sensor, we can accurately calculate the dimensions x and y in distance meters. This is done by using trigonometric function for right triangles. Since the camera being used is a Raspberry Pi Camera V2.1 (module v2) the horizontal FOV



(a) Drift tracking from the Raspberry Pi camera

(b) Feature tracking Raspberry Pi camera

Fig. 3

is 62.2 degrees and the vertical FOV is 48.8 degrees[1]. With this the distance spanning the ground in the cameras view can be calculated (see also figure 2):

$$\frac{opposite}{adjacent} = \tan(\theta)$$

$$\theta = \frac{FOV}{2},\ adjacent = height,\ opposite = \frac{distance}{2}$$

$$distance = 2 \cdot \tan(\frac{FOV}{2}) \cdot height$$

Knowing the distance in the camera's view and the resolution in pixels, the drift can be calculated in a unit of length, e.g. meters, through conversion.

$$distance\ per\ pixel = \frac{distance}{resolution}$$

This will give a different result for the x- and y-axis, respectively, however the drift can now be accurately measured.
As a side note, the camera needs to calibrated to prevent warping.

[1]https://www.raspberrypi.com/documentation/accessories/camera.html

Additionally, the camera software can detect angular drift in the rigid transformation, and this can be utilized if the drones gyro lacks in accuracy.
The program visualized can be seen in figure 3a, when the drone is moving. The distance moved (dx and dy [m]) in one frame is calculated from the distance between prev_pts (red dots) and curr_pts (green dots). In figure 3b the camera picks up reference points from a more random pattern, while the drone is still.

D. GUI

Our advisor Jens Christian Andersen has provided us with a GUI, graphical user interface, for the drone. This makes it easier for others to use the drone and see the measurement's from the drone graphically. The GUI consists of a lot of tabs, which you can be enable or disable, for example: gyro, height, motor. Here we have added a tab called camera which can be seen in figure 4. The x- and y-drift are being displayed here in decimal number and graphically as a function of time. The velocities, dx and dy, are also display as decimal numbers.

IV. Future work

For future work, we need to get the height measurement from the Teensy and combine it with our camera software on the Raspberry Pi, in order to get a more precise x- and y-drift in units of length.

V. Discussion

Our project have the benefits of detecting drift, so it can be used in future project's control design for making the drone more stable. Possible pitfalls with our project, is the lack of testing for our setup in different environments. If the ground is very uniform, it could be difficult for the drone software to find any features with high contrast to track, leading to inaccurate drift estimations.
With our current knowledge it was easiest to implement the camera software in python, for all the top level actions. The packages we used (OpenCV) where written in C++, so it is only the top level actions which are done in python. With a greater knowledge of C++, we could also write the top level action in C++ for greater efficiency.
If problems with accuracy of the reference tracking software are encountered, it should be considered to use stereo cameras, to get more depth information. Instead of only having depth information for one point, as we have now with the ultrasound sensor, we could have depth information about the whole plane the camera is seeing. This could increase the accuracy of the drift estimations.
However, currently it cannot be realised on the current Raspberry Pi model, because of the number of camera inputs, you would have to upgrade to a StereoPi.

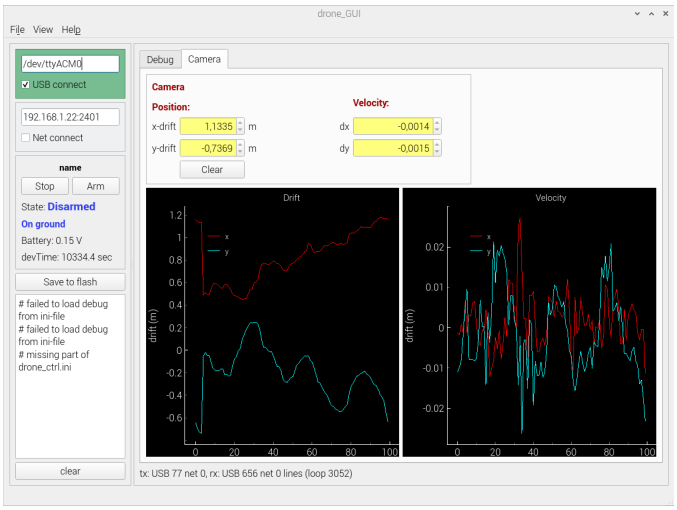


Fig. 4: GUI for the camera.

VI. Conclusion

We have accomplished our goals of the physical implementation of the ultrasound sensor and the camera. For the software implementation we have accomplished implementing reference tracking software on the Raspberry Pi. For the software implementation to be complete, we need to fine adjust the camera software and feed it into the GUI.

VII. Guide for future users

Useful Linux terminal commands:

cd .. *go back one directory*
ls -a *list all files*
mkdir folder_name *make folder*
geany *opens teansy code editor called Geany*
clear *clear terminal*
pwd *print working directory*
ls a* *shows all which start with a*
sudo cp <fileName> <path> *copy file to*
grep -r search_term . *search in the current directory and sub directories.*
wget -O foldercomp.zip _web_Link_ *downloads folder from web*
chmod +rwx file_name *changes read, write, and execute permissions.*
nano filename.txt *edit file*
cat filename.txt *view file*

A. Firmware

There is a guide for uploading the firmware to the Teensy the Drone Control wiki[2], we will just add a few comments to make it easier for future users. It should be possible to upload the firmware to the Teensy from Windows, but this is a real struggle, so we recommend using Linux for uploading the firmware. The easiest way to do it, however, is to use the Raspberry Pi, which already runs Linux. In the drone_ctrl directory, you need to make shortcuts to the Arduino libraries, as described in the wiki.

B. Rasberry Pi camera and camera software

In our reference tracking software we are using OpenCV, and our code only works with a specific version of OpenCV, namely version 3.x.x. So follow the guide below for setting up the camera and software: In the terminal type:

```
$ pip install opencv-python==3.4.18.65
$ sudo raspi-config
  3 → 1 → Yes
$ sudo reboot
$ sudo apt-get update && sudo apt-get upgrade
$ sudo apt autoremove
$ pip install -U numpy
$ pip install --upgrade pip setuptools wheel
$ sudo apt-get install -y libhdf5-dev
libhdf5-serial-dev python3-pyqt5
libatlas-base-dev libjasper-dev
```

C. Rasberry Wifi

For connecting the GUI to the drone over Wifi, our advisor has made a file called mission, which should establish a wifi connection between your computer and the Raspberry Pi. We had a lot of problems with this code not working on the drone. Therefore we are using another method where we are connecting to the Raspberry Pi as a remote desktop. This is set up and working on our drone so you don't need to do any of the following, but if you want te implement it from scratch you need to do the following:
Your computer and the raspberry pi needs to be on the same network, for examle DTUsecure. For connecting the raspberry pi to DTU's Wifi, look at the wiki page.[3] under "Connecting Raspberry Pi to Eduroam". Enter the following in the terminal on the raspberry pi to enable VNC:

```
$ sudo raspi-config
  5 → 3 → Yes
```

Then download VNC viewer on your computer.
Now you can connect to your Raspberry Pi, and see the desktop.
To boot the raspberry pi without a screen attached you need to do the following in the Rasberry Pi terminal:

```
$ cd /boot
$ sudo nano config.txt
```

Here you need to uncomment/add:

```
hdmi_force_hotplug=1
```

and uncomment/add the following lines:

```
hdmi_group=1
hdmi_mode=16
```

Then press 'ctrl+x', and then 'y' and close the terminal. OBS. be very careful when editing this file. If misspells happens, the Raspberry Pi will not boot at all.
Last press the Raspberry Pi logo → Preferences → Screen Configuration → Layout → Screens → HDMI-1 → Resolution → 1920x1080

D. GUI

When editing the GUI you do not have to change anything manually in mainWindow.py. This file is produced by Qt Designer. So download Qt Designer, and open the mainWindow.io. When you have made your changes in Qt Designer open the terminal in the drone_gui folder and write:

```
$ pyuic5 mainWindow.ui -o mainWindow.py
```

This will create the python file used by drone_gui.py.
To add a tab, you need to design a new tab in Qt Designer, add a new file myTab.py, and then change drone_gui.py and main.py.
Run the GUI by typing in the command prompt:

```
$ python drone_gui.py
```

E. Issues and difficulties

The code found in the SVN repository for the GUI and firmware is not plug and play, so we have made a lot of small changes so it works. For anyone working on this in the future, use our code.

[2]http://rsewiki.elektro.dtu.dk/index.php/Drone_firmware

[3]http://rsewiki.elektro.dtu.dk/index.php/Instructions_for_getting_started