QUADROCOPTER NAVIGATION AND CONTROL

Karel Doležal
Bachelor Degree Programme (3), FIT BUT
E-mail: xdolez44@stud.fit.vutbr.cz

Supervised by: David Herman
E-mail: iherman@fit.vutbr.cz

Abstract: In recent years, there has been a significant amount of attention given to quadrotor helicopters. This paper describes autonomous navigation of Parrot's AR.Drone quadrocopter which allows travel along a specified route and autonomous landing on a platform placed at the destination. Firstly, navigation architecture and extension of the UAV's sensors with GPS and compass module are mentioned. In the second part of the paper, the use of video stream from onboard cameras for the purposes of platform and obstacle detection is described.

Keywords: quadrocopter, AR.Drone, GPS, optical flow, autonomous navigation, obstacle detection

1 INTRODUCTION

AR.Drone platform is intended to be used for augmented reality gaming purposes. The vehicle is a self-stabilizing WiFi controlled UAV (Unmanned aerial vehicle) capable of sending video stream back to the user. Onboard sensors include a 3-axis gyroscope, an ultrasonic range meter (used as an altitude sensor), a horizontal and a vertical camera [1]. A SDK is available to allow development of 3rd party games. With this toolkit, the vehicle can be controlled from PC using simple high-level commands.

2 NAVIGATION ARCHITECTURE

Navigation architecture consists of blocks connected as shown in Figure 1. Yellow blocks represent the UAV's sensors and *Waypoints*, the source of mission data. Red blocks represent data processing units. The *Global navigation* block computes the distance and azimuth to the next waypoint which is used as preferred heading by the *Local navigation*. The *Platform detection* block is activated at the final waypoint. All maneuver processing and reading information from the other computing blocks is then handled by *Local navigation* block, which also detects obstacles and is responsible for avoiding them.

3 SENSOR EXTENSION

The outdoor navigation requires a GPS sensor to calculate the azimuth and distance to the next waypoint and a magnetic compass to determine the current attitude of the UAV. Conveniently, there is a module currently available which incorporates both sensors mentioned above, LS20126 (seen in Figure 2 on PCB with power source, mounted on the top of the horizontal camera). The module has a standard UART interface and sends both position data and data from additional sensors in standard NMEA text format.

3.1 Interfacing with the module

AR.Drone has an extension connector with USB port, which is reserved for future applications and currently disabled by firmware [1]. Although modifying the firmware would be in conflict with the Parrot license, there are pins on the connector marked as "Not Connected" that are in fact a two way UART port of the main processor. Furthermore, the UART port is accessible from the embedded Linux-based real-time operating system as a standard serial device located at */dev/ttyPA0*. As AR.Drone listens on TCP port 23 for telnet connections, it is possible for the user program to connect to the system remotely and execute a simple script to receive data from the LS20126 module. This sets aside the need for additional RF transmitter.

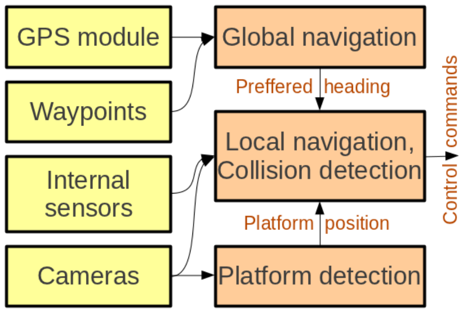

Figure 1: Navigation architecture overview.

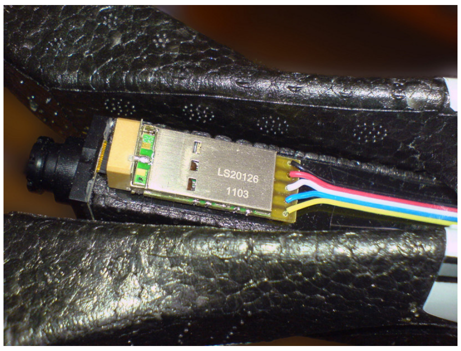

Figure 2: LS20126 module.

4 VIDEO STREAM USAGE

There are two cameras on the UAV. The incoming video has a resolution of 320×240 px at 15 frames per second. It is also possible to combine the images from both cameras, so that we get 320×240 px horizontal image with 88×72 px vertical image in the top left corner.

4.1 Landing platform detection

After reaching the final waypoint of the route, the video is switched to combined view, showing both horizontal and vertical images, and the control program starts looking for the platform, which is a 60×60 cm red box. A conversion of the image to HSV color space and a simple threshold filter is applied. Resulting binary image is then eroded to suppress noise, and a sufficient count of pixels in the horizontal image is considered to be a possible platform.

4.2 Obstacle detection

The obstacle detection method is based on the idea that obstacles such as trees and pedestrians near the moving quadrocopter cause higher optical flow in the video than the background. The length of optical flow vectors can be thresholded and high values counted as obstacles. The problem here is that the movements of the vehicle itself cause great amount of movement in the image. Therefore the major movements must be subtracted or suppressed to eliminate this problem. The vector length thresholds must also be calibrated to ignore obstacles that are far enough.

The processing consists of number of stages. First of all, the video stream is processed by OpenCV library calculating the optical flow between each two images [2]. There often is an image-wide drift caused by wind or by moving willingly. To deal with this, the mean value of all vectors is computed and then subtracted from them.

Another source of unwanted movement is rotation. Typical unwanted flow caused by maneuvering AR.Drone is shown in Figure 3. As can be observed, the center of rotation is not always in the center of the image, because the movements are often composed. When examining the vectors from left to right, notice that y-axis part of vectors reaches zero at the center of the rotation. Knowing this, we can approximate y-axis parts of the vectors in dependence on their x position in image using least squares method. At the point where $y = 0$, there is an x coordinate of the center of rotation. We do the same for x-axis part in dependence on y to determine the y coordinate. When the center is known, the flow vectors are changed so that only component towards the center is retained.

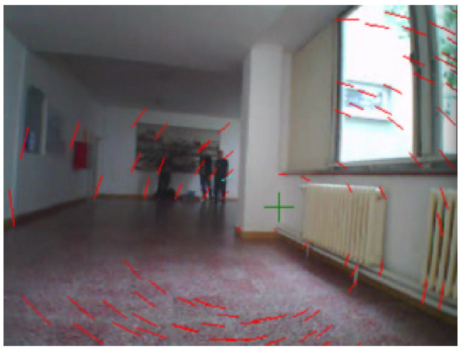
Figure 3: Optical flow caused by rotation of the vehicle.

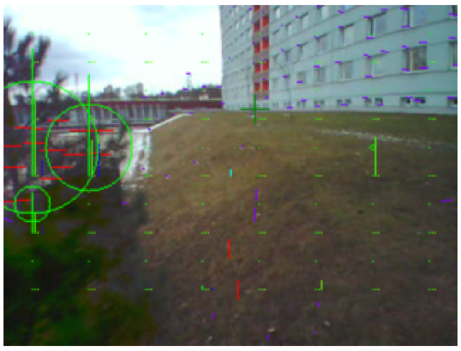
Figure 4: A tree detected when flying near by. The group of red lines indicates vectors above threshold.

Finally, the lengths of the vectors are thresholded and summarized over few last frames. A score is computed according to length and number of vectors that lay within squares of a 8×6 grid. In Figure 4, the radius of green circles corresponds to detection scores. The figure shows a case where the detection worked quite well; however, more calibration is needed for usable operation in various environments.

5 CONCLUSION

The GPS and compass module are connected to the main processor and accessed over built-in WiFi. Landing platform is recognized by color from camera image in HSV color space. The obstacle detection method measures length of optical flow vectors after that major vehicle's movements are subtracted. At the present time, the above-described detection methods are working suitably although a proper calibration is needed.

ACKNOWLEDGEMENT

I would like to thank my supervisor David Herman for his expertise and support during my work. This work was supported by research plan MSM0021630528.

REFERENCES

[1] Parrot SA. (2011) *AR.Drone Developer Guide*. [e-book] Available through: projects.ardrone.org https://projects.ardrone.org/attachments/download/365/ARDrone_SDK_1_7_Developer_Guide.pdf [Accessed: 27th February 2012].

[2] Bradski, G. and Kaehler, A. (2008) *Learning OpenCV*. Sebastopol: O'Reilly, p.555.