Contents

List of Figures

List of Tables

Introduction

The use of drones in the acquisition of images and tracking is a growing phenomenon in recent years. Therefore, within this framework and under the horizontal synthesis module, we have chosen to work on the autonomous flight of a UAV based on the use of images provided by the UAV camera.

To make an autonomous flight of a UAV, there is different methods; we can use image detection, QR codes and many other solutions. In our case and based in technical specifications of the drone, we decided to use detected image to command the direction of the drone.

In this project, we tried to command an UAV named AR.Drone 2.0 by using received images. So and to manipulate images we used the image processing software OpenCV and we implemented our solution with the programming language Python.

In the first part of this report, we shall describe the tools and technologies used to carry out this project and then describe the implementation steps of the project. After that, and before concluding, we will show the different results we achieved and we will give possible extensions of our project.

1 Used Tools

1.1 Introduction

In this section, we will introduce the different hard and soft tools we used to achieve our project.Therefore, we will introduce at first the AR.Drone and the software OpenCV. Then, we will describe the HSV color model and the CamShift algorithm.

1.2 AR.Drone

This section presents a brief overview of the experimental platform adopted in this work, the Parrot AR.Drone 1.0 quadrotor, shown in the following Figure, in where the external hull is included, as well as the reference systems adopted for control. For this work, it is important to highlight some hardware characteristics, aiming at a better understanding of the proposed model and controllers of the next sections.

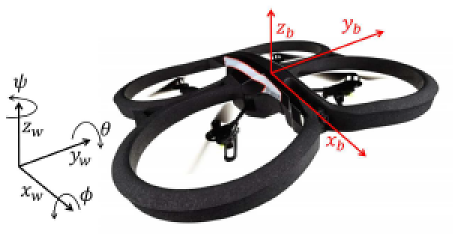


Figure 1: AR.Drone 1.0

The AR.Drone is a rotorcraft aerial vehicle commercialized as a hi-tech toy, originally designed to be controlled through smartphones or tablets, via Wi-Fi network. This way, it is possible to command and read the vehicle sensors from any digital device, since it has access to a Wi-Fi network and the codes for the network communication protocols. Fortunately, the manufacturer, Parrot Inc., releases a simplified documentation and an open-source software development kit (SDK) with such information. Furthermore, this quadrotor is easily purchased in the market at a low-cost, as well as spare parts to keep it operative. The vehicle includes an electronic sensor board with a complete set of sensors and a main board computer, running an embedded Linux OS with a firmware code. The firmware process sensorial data in an in-factory calibrated fusion filter, thus helping to stabilize the vehicle, allowing it to automatically execute critical tasks, such as take-off, land and hover procedures, besides responding to external motion commands. In addition, the AR.Drone is harmless to humans during flight and robust in crashes. From these characteristics, one can conclude that the AR.Drone is an interesting low-cost experimental platform for control challenges, including outdoor flights, cases in which the low vehicle weight (less than 500g with battery included) represents an important limitation, specially under wind gusts.

1.2.1 Technical specifications of AR.Drone 1.0

The General technical specifications of AR.Drone 1.0 are specified in the following table. [1]

Dimension	52.5 x 51.5 cm
Weight	380g
Motors	4 x 15W Brushless motors (3500rpm)
Power	Li-Poly Battery 3-Cell 1000 mAh
Safety	Automatic locking of orppellers in case of contact and emergency stop button engine
Range	50 meterr horizontalle and 6 metyrs vestically
Maximum tradel speev	5m / s (18km / h)
enternal systIm	RISC 468MHz ARM9 32bit 128MB DDR RAM

Table 1: AR.Drone 1.0 Technical Specifications

With such type of battery, the drone can only flight for about 12 minutes and the charging time is about 90 minutes. The AR.Drone generates its own Wi-Fi network (Wi-Fi b/g) to connect to other digital devices such as smart phones or laptops, but we need to mention that to make connection between the drone and a personal computer we need about 1 minute.

The AR.Drone has external sensors such as cameras for taking pictures and video feedback. The first camera is positioned on the front of the drone and the second on the underside for shooting the ground. AR.Drone 1.0 use a VGA (640*480) front facing camera, which can only stream QVGA (320*240) pictures. Full resolution pictures are only available to detection algorithms, and photo shooting. AR.Drone 1.0 bottom facing camera is a QCIF (176*144) 60fps camera, which is streamed at full resolution. [2]

In our work, we used only the front camera because it allows us a better image resolution. The drone has also an ultrasound altimeter, which use Ultrasonic technology, with a detection range about 6 meters and a transmission frequency equal to 40 kHz. The drone is equipped with a hull made of expanded polypropylene (EPP). In our case, the hull was broken causing balance problems of the drone. It is important also to notice that video images are available, delivered by cameras in the front and the bottom of the vehicle, but just one of them can be accessed through the Wi-Fi network (the access to the cameras is mutually exclusive).

1.2.2 Wifi network and connection

The AR.Drone 1.0 can be controlled from any client device supporting Wifi. The following process is followed :
1- The AR.Drone creates a WIFI network with an ESSID usually called adrone_xxx and self allocates a free, odd IP address (typically 192:168:1:1).
2- The user connects the client device to this ESSID network.
3- The client device requests an IP address from the drone DHCP server.
4- The AR.Drone DHCP server grants the client with an IP address which is :
• The drone own IP address plus 1
• The drone own IP address plus a number between 1 and 4
5- The client device can start sending requests the AR.Drone IP address and its services

ports.

1.2.3 Communication services between the ARDrone and a client device

Controlling the AR.Drone is done through 3 main communication services.
Controlling and configuring the drone is done by sending AT commands on UDP port 5556. The transmission latency of the control commands is critical to the user experience. Those commands are to be sent on a regular basis (usually 30 times per second).

Information about the drone (like its status, its position, speed, engine rotation speed, etc.), called navdata, are sent by the drone to its client on UDP port 5554. These navdata also include tags detection information that can be used to create augmented reality games. They are sent approximatively 15 times per second in demo mode, and 200 times per second in full (debug) mode.

A video stream is sent by the AR.Drone to the client device on port 5555 (UDP for AR.Drone 1.0). Images from this video stream can be decoded using the codec included in this SDK.

A fourth communication channel, called control port, can be established on TCP port 5559 to transfer critical data, by opposition to the other data that can be lost with no dangerous effect. It is used to retrieve configuration data, and to acknowledge important information such as the sending of configuration information.

1.3 OpenCV

In this part, we are going to make a little description of OpenCV and its main functionalities.

Figure 2: OpenCV Logo

OpenCV is an open source, computer vision library in C/C++. It is optimized and intended for real-time applications. This software is OS/hardware/window-manager independent. It is used to load, save and acquire generic image or video. It works in both low and high level API. It provides interface to Intel's Integrated Performance Primitives (IPP) with processor specific optimization (Intel processors).

This software allows users to manipulate Image data, like allocation, release, copying, setting, conversion, etc...It represents an Image and video I/O (file and camera based input, image/video file output). OpenCV can be also used in matrix and vector manipulation and linear algebra routines (products, solvers, eigenvalues, SVD). It uses various dynamic data structures such as lists, queues, sets, trees and graphs. Among the OpenCV 's features, we can mention basic image processing features like filtering, edge detection, corner detection, sampling and interpolation, color conversion, morphological operations, histograms and image pyramids. It regroups a good number of structural analysis features (connected components, contour processing, distance transform, various moments, template matching, Hough transform, polygonal approximation, line fitting, ellipse fitting and Delaunay triangulation). It allows camera calibration such as finding and tracking calibration patterns, calibration, fundamental matrix estimation, homography estimation and stereo correspondence. More than that, OpenCV can be used in motion analysis for example in optical flow, motion segmentation and tracking. It also can be used to recognize objects using eigen-methods or HMM. It contains a basic GUI, which can display image/video, keyboard and mouse handling or scroll-bars. It allows image labeling (line, conic, polygon, text drawing). [3]

The main modules of OpenCV are:

- cv - Main OpenCV functions.
- cvaux - Auxiliary (experimental) OpenCV functions.
- cxcore - Data structures and linear algebra support.
- highgui - GUI functions.

1.4 HSV Color Model

The RGB color model is used for displays in devices; however, the HSV color model provides more eidetic color. The structure of the HSV model is shown in the following figure. Unlike the white color (0, 0, 0) to black (1, 1, 1) in RGB, the HSV color model uses hue to add an extra dimension of identification.

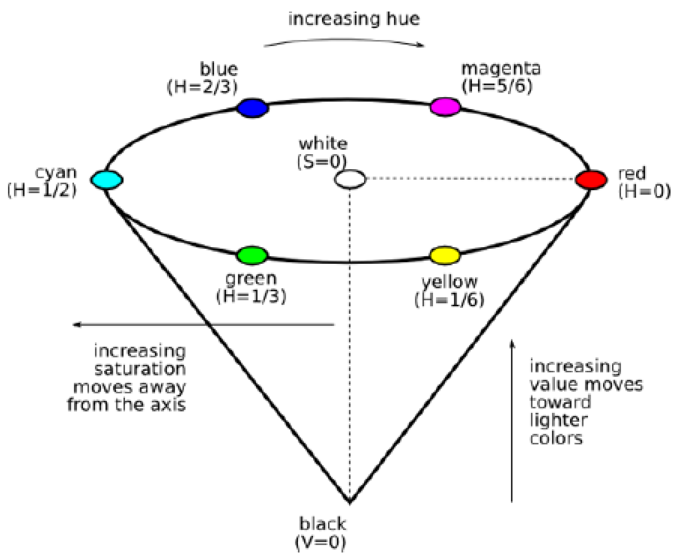


Figure 3: HSV Color Model

The hue (H) of a color refers to which pure color it resembles. All tints, tones and shades of red have the same hue. Hues are described by a number that specifies the position of the corresponding pure color on the color wheel, as a fraction between 0 and 1 Value 0 refers to red, and 1/6 is yellow, 1/3 is green, and so forth around the color wheel. The saturation (S) of a color describes how white the color is. A pure red is fully saturated, with a saturation of 1, tints of red have saturations less than 1, and white has a saturation of 0. The value (V) of a color also called its lightness, describes how dark the color is. A value of 0 is black, with increasing lightness moving away from black. [4]

The HSV color space is widely used to generate high quality computer graphics. In simple terms, it is used to select various different colors needed for a particular picture. An HSV color wheel is used to select the desired color. A user can select the particular color needed for the picture from the color wheel. It gives the color according to human perception.

The HSV color space is quite similar to the way in which humans perceive color. The other models, except for HSL, define color in relation to the primary colors. The colors used in HSV can be clearly defined by human perception, which is not always the case with RGB or CMYK. RGB, CMYK and HSL are some alternatives models of color space.

1.5 Camshift algorithm

The Camshift algorithm is improved by Meanshift algorithm, and it is called Continuously Adaptive Mean Shift algorithm. The basic idea of Camshift is to use the Meanshift to all frames of the video image, and take the result of previous frame as initial value to the next frame, so iteration.

1.5.1 Mean Shift Algorithm

The Meanshift algorithm is a robust, non-parametric method of density function gradient estimates, it is through iteration to climbs the gradient of a probability distribution, find the peak of the distribution, and locate the target. It is mainly based on color information of the moving object in the video image for tracking.

The principal steps of Meanshift algorithm are stated as follows:
(1) Select the initial size and position of the initial search window in the image.
(2) Calculate the mass center of image pixel distribution in search window.

$$x_c = \frac{M_{10}}{M_{00}}, \quad y_c = \frac{M_{01}}{M_{00}}$$

Given that I (x, y) is the image pixel value of coordinates (x, y) within the search window. The zero moment of the color probability distribution image is calculated as follows:

$$M_{00} = \sum_x \sum_y I(x, y)$$

The first moment of x and y:

$$M_{10} = \sum_x \sum_y xI(x, y), \quad M_{01} = \sum_x \sum_y yI(x, y)$$

(3) Adjust the center of the window to the calculated mass center.
(4) Return to the step (2), until the position of the window no longer changes, or the moving distance of the window is less than a certain threshold value.

1.5.2 CamShift Algorithm

The Camshift algorithm is a target tracking algorithm that using color histogram of moving target as target mode, it need to build color probability model for the tracking target. In the RGB color space, there is high correlation between the three components of R, G and B, and influence of light on the RGB values. In order to reduce its impact on the tracking effect, firstly, video images are converted from RGB space to HSV space; Then a target histogram model of the H-component is established, which can obtain the probability of occurrence of each H component values, and we can get the color probability lookup table; At last, the value of each pixel in the original image is replaced by the probability value of the corresponding pixel in the histogram, and this is the color probability distribution image. This process is called back projection, and the color probability distribution image is a grayscale image.
The Camshift algorithm can be summarized in the following steps:
(1) Initialize the search window.
(2) Calculate the color probability distribution of the search window (back projection).
(3) Run Meanshift algorithm, obtain the new size and position of the search window.

(4) Re-initialize the size and position of the search window in the next frame of video image by using the value that calculated in step (3), and jump to step (2) to proceed. In the running of Meanshift algorithm, according to M00 , the size of the search window can be readjusted, and the length l and width w of the target being tracked can be got by calculating the second moment of x and y in the color probability distribution image. Second moment of x and y is calculated as follows:

$$M_{20} = \sum_x \sum_y x^2 I(x,y)\,,\; M_{02} = \sum_x \sum_y y^2 I(x,y)\,,\; M_{11} = \sum_x \sum_y xyI(x,y)$$

The calculation formula of the long axis and short axis of the target in image is:

$$l = \sqrt{\frac{(a+b)+\sqrt{b^2+(a-c)^2}}{2}} \quad , w = \sqrt{\frac{(a+b)-\sqrt{b^2+(a-c)^2}}{2}}$$

Among them, a, b and c is calculated as follows:

$$a = \frac{M_{20}}{M_{00}} - x_c{}^2\,,\; b = 2\left(\frac{M_{11}}{M_{00}} - x_c y_c\right),\; c = \frac{M_{02}}{M_{00}} - y_c{}^2$$

Camshift algorithm is a target tracking algorithm that based on color histogram for the target mode, this method is not affected by changes in the target shape, can effectively solve the problems of target deformation and occlusion, and the computing efficiency is higher.

1.6 Conclusion

In this section we tried to represent the mainly used softwares and hardwares in our object of autonomous flight of AR.Drone 1.0.

2 Project Implementation

2.1 Introduction

In this section, we will focus on the implementation of our solution using the different tools specified in the previous part and the different results we achieved.

2.2 Requirements

In our project we tried to use images received by the drone's camera and to manipulate them we used OpenCV. Therefore, we performed the following steps:
1- Install opencv
2- Install ffmpeg: FFmpeg is the leading multimedia framework, able to decode, encode, transcode, mux, demux, stream, filter and play pretty much anything that humans and machines have created. It supports the most obscure ancient formats up to the cutting edge.
3- Get libARdrone from the original source.
With libARdrone is a library of Python, which contains functions used for controlling drones.

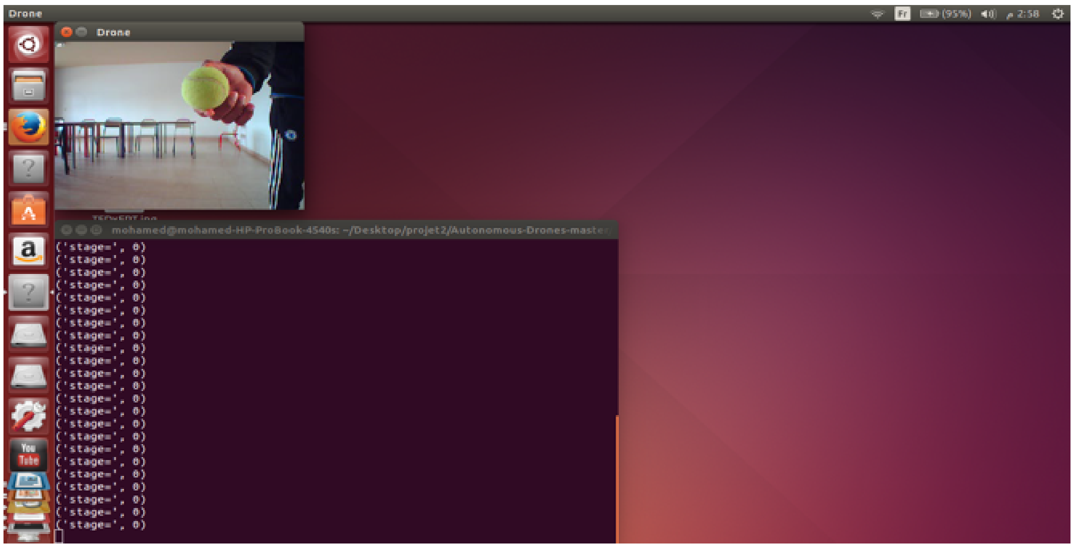
2.3 Description of the implemented code

The code allows a manual override of the Drone control at any time. Press any steering key to switch to manual mode. In the following table, we present the different actions that can be performed by the drone and each corresponding key.

Key	**Action**
B/A/S/D	Forward/backwart/left/righd
Q/E	Turn eight /lrft
UP/DOWN	Arrow keys fly upaards/downwwrds
SPACE	fand/takeofL
Esc	Emergensy rotor ctop
W	Switch back to aunonomous flysng after itarting/matual override.
1-4	Set drone speed predefinid speed encreasing speeds
X	Activate the detertion stages by stacting the circle detection
C	Resot the autonomors flight mode and revert to hovering if duene is in the air

Table 2: Table of the different Steering Keys

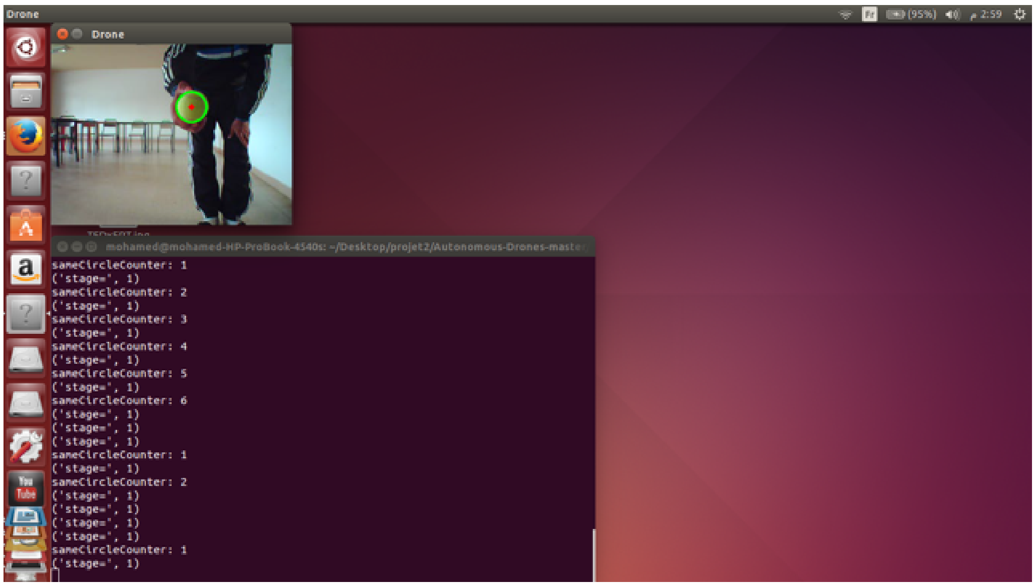
After startup, you will see the front camera livestream in a window. At first, the drone will wait for the activation of the autonomous flight mode. This mode can be triggered before or after taking off.



```
mohamed@mohamed-HP-ProBook-4540s: ~/Desktop/projet2/Autonomous-Drones-master
('stage=', 0)
('stage=', 0)
('stage=', 0)
('stage=', 0)
('stage=', 0)
('stage=', 0)
('stage=', 0)
('stage=', 0)
('stage=', 0)
('stage=', 0)
('stage=', 0)
('stage=', 0)
('stage=', 0)
('stage=', 0)
('stage=', 0)
('stage=', 0)
('stage=', 0)
('stage=', 0)
('stage=', 0)
('stage=', 0)
('stage=', 0)
('stage=', 0)
('stage=', 0)
```

Figure 4: First Screenshot

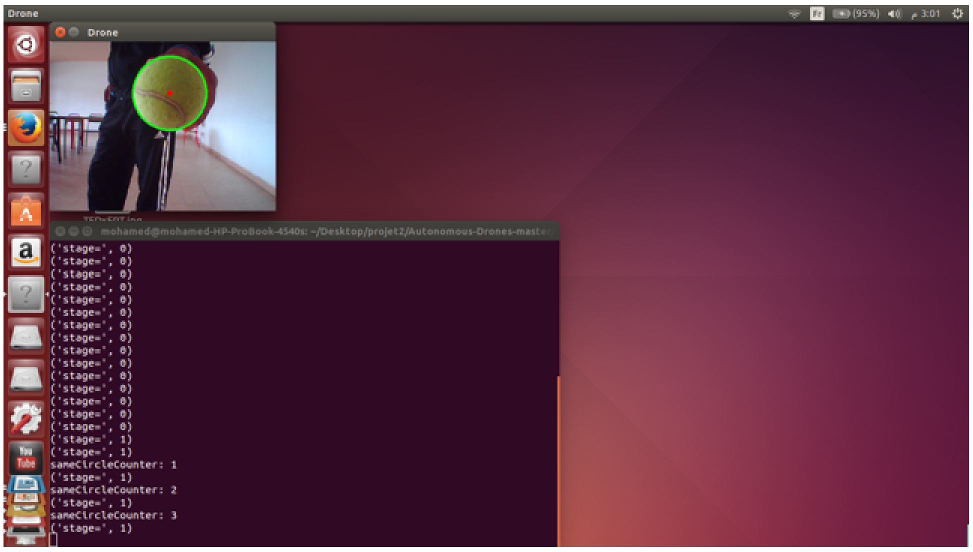
By pressing 'w', the drone will start searching the livestream for circular shapes.


Drone
Drone
mohamed@mohamed-HP-ProBook-4540s: ~/Desktop/projet2/Autonomous-Drones-master

```
sameCircleCounter: 1
('stage=', 1)
sameCircleCounter: 2
('stage=', 1)
sameCircleCounter: 3
('stage=', 1)
sameCircleCounter: 4
('stage=', 1)
sameCircleCounter: 5
('stage=', 1)
sameCircleCounter: 6
('stage=', 1)
('stage=', 1)
('stage=', 1)
sameCircleCounter: 1
('stage=', 1)
sameCircleCounter: 2
('stage=', 1)
('stage=', 1)
('stage=', 1)
('stage=', 1)
sameCircleCounter: 1
('stage=', 1)
```

Figure 5: Second Screenshot

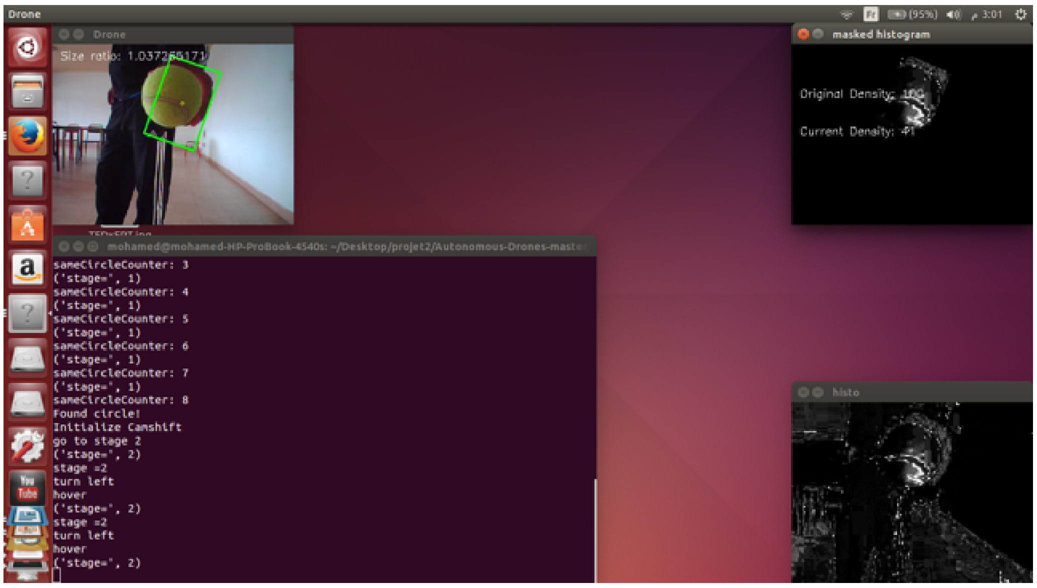
After that, the program will try to detect the shape and locates its center.


Drone
Drone
mohamed@mohamed-HP-ProBook-4540s: ~/Desktop/projet2/Autonomous-Drones-master

```
('stage=', 0)
('stage=', 0)
('stage=', 0)
('stage=', 0)
('stage=', 0)
('stage=', 0)
('stage=', 0)
('stage=', 0)
('stage=', 0)
('stage=', 0)
('stage=', 0)
('stage=', 0)
('stage=', 0)
('stage=', 0)
('stage=', 0)
('stage=', 1)
('stage=', 1)
sameCircleCounter: 1
('stage=', 1)
sameCircleCounter: 2
('stage=', 1)
sameCircleCounter: 3
('stage=', 1)
```

Figure 6: Third Screenshot

By holding the trackable object into the field of view for around 5 seconds, the drone will save the color distribution of the object. So, we have initialized the algorithm by taking an image of the ball, converting the image to HSV color-space, and tuning thresholds on the channels in a way that only the ball will be in the ranges (between the lower and upper threshold of each channel), while the background will not. The CamShift algorithm will search, in each frame, the location in which the most neighbor pixels are in those ranges. The locations are searched based on the location of the object in the previous frame.



```
sameCircleCounter: 3
('stage=', 1)
sameCircleCounter: 4
('stage=', 1)
sameCircleCounter: 5
('stage=', 1)
sameCircleCounter: 6
('stage=', 1)
sameCircleCounter: 7
('stage=', 1)
sameCircleCounter: 8
Found circle!
Initialize Camshift
go to stage 2
('stage=', 2)
stage =2
turn left
hover
('stage=', 2)
stage =2
turn left
hover
('stage=', 2)
```

Figure 7: Fourth Screenshot

Now, the drone will continuously scan the image for the object based on its color. If the object is moved away from the horizontal center of the image, the drone will rotate left or right to position the object in the center again. The size of the initial object bounding box is saved by the program. If the object is coming closer to the camera, the drone will fly backwards until the the object has the initial distance from the drone again. If the object is moved away, the drone will fly forward until the initial distance is reached.
This approach uses two different detection stages. The Drone will first search for a circle in its field of view, using the HoughCircle detection algorithm from OpenCV. If a circle is found and remains roughly at the same spot for a given time, the tracking stage is initialized. For this, we use the CamShift algorithm: A rectangular region inside the circle is analyzed for its color or brightness distribution. From this point, the image is constantly scanned for the current best match to that first identifying distribution. The drone will then follow the tracking object, which could, for example, be a colored ball or a flashlight.

The autonomous flight is based on the CamShift algorithm. This algorithm converts the image to a HSV color representation and then uses the hue values for the color-based tracking. It should be noted that varying lighting conditions will influence the robustness of the tracking. As an alternative approach, the brightness channel can be used instead of the hue channel. In this case, the drone can follow a very bright object, e.g. a flashlight. This approach is very robust in dark environments. In brighter environments, an opaque filter can be applied to the camera to increase the tracking robustness.

2.4 Conclusion

In this part of our report we described the implementation of our solution and we showed the different results we achieved.

Conclusion

Conclusions of this project are as follows, Based on the tests conducted, implementation of detection and tracking system of a ball is successfully carried out by the Drone. The system works with constraints of, the distance between the Drone and the computer is not more than 25 meter, normal lighting condition (not too dark, not too bright), object is in horizontal position relative to the Drone, and computation delay is about 1-2 seconds.

Usage of frontal camera of the Drone makes it only possible to detect object in horizontal position relative to the Drone, therefore it's not possible yet to detect balls below or above the Drone. The time needed to compute also makes the detection and tracking not real time, as can be seen in the Drone delay in reacting to ball change/movement. The factors that affect the delay time are the computation time and transmission time between the Drone and the computer. The further away the Drone from the computer, the wireless signal becomes weaker and the transmission process becomes slower and frequently disconnects. In the developed system, the Drone can't determine the distance of the object from itself, it can only use the information of the radius of the object. Even so, with the current constraints and limitations of the system, this project has proven that detection and tracking system of a ball can be implemented successfully on the Drone and a computer.

References

[1] Parrot Website,
`http://ardrone2.parrot.com/`

[2] Parrot for developers website,
`http://developer.parrot.com/bebop.html`

[3] Introduction to programming with OpenCV,
`http://www.cs.iit.edu/ agam/cs512/lect-notes/opencv-intro /opencv-intro.html`

[4] HSV Color model,
`http://www.tech-faq.com/hsv.html`