Autonomous Flight With the Ar Drone©

Maarten Inja & Maarten de Waard
5872464 & 5894883

January 31, 2011

1 Introduction

1.1 Our Goal

Our absolute goal, is winning in some subtasks of the Summer-IMAV 2011 Indoor competition [1]. This is a competition for flying radio controlled vehicles to autonomously solve some tasks. For this competition, we tried to solve the following sub-tasks:

- Pick-up Object
- Exit Building
- Release Object

We used the Parrot Ar Drone© to solve these problems.

1.2 The Ar Drone©

The Ar Drone© is an over WiFi remote controlled quadrocopter that has several onboard sensors:

- One vertical camera, pointing downward
  This is also used for stabilisation and calculating the speed. It has a 63° angle.
- One horizontal camera, pointing forward
  This is a 91° angle camera.
- Ultrasound altimeter, to measure the altitude
- 3 axis accelerometer (measures propellor acceleration)
- 2 axis gyrometer
- 1 yaw precision gyrometer

Furthermore, it has an onboard computer system running Linux.

[1] This link refers to a pdf file explaining the competition.

Figure 1: The Ar Drone©

2 Controlling the Ar Drone©

The Ar Drone© has like any other flying vehicle the usual 3 critical flight dynamic parameters for the rotation angles; pitch, yaw and roll.

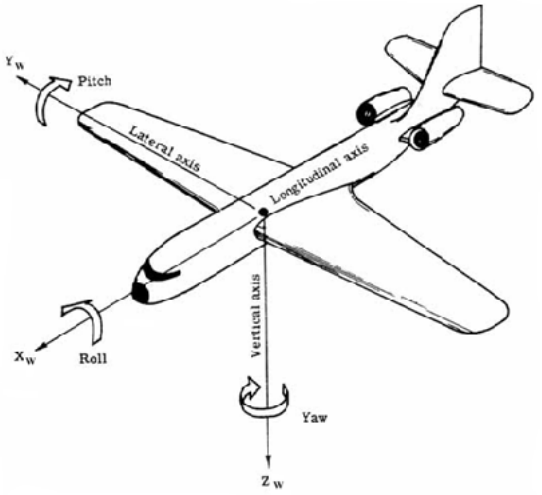

Figure 2: Movement Directions For Flying Vehicles

2.1 ROS

2.2 SDK

Software Development Kid

2.3 Extending C with Python

It's awesome possum

3 Methods Used

We divided the first subtask in three subgoals: Finding the object; tracking the object and picking up the object. This shows step by step how to do that.

3.1 Finding The Object

To find the object we used a simple routine, in combination with a more complicated recognition. The routine is: Start flying at a height, and then turn 360 degrees. While turning we constantly check the front camera (since we can only check one camera properly at the same time) for our object.

We chose to make an object that was bright pink. This simplified things for us: We only had to recognize a big enough surface of our color. We tried a couple of things to recognize our color:

The first step was to define the values that our color ranged from. We chose to divide our image in HSV values. This divides the image in 3 different layers: Hue, Saturation and Value. These layers represent the values of the image's "Hue, Saturation and Value" as shown in figure 3. OpenCV changes these values to a range of 0 to 255 (to create images it can show) by dividing the Hue by 2, and the saturation and value by $\frac{100}{255}$. The advantage of HSV above RGB (Red Green Blue) values is that it is easier to recognize of your color needs to be a bit lighter (thus, increasing the Value) than that your color needs to be a bit greener.

OpenCV placed the pixels that were in the right HSV range in a new picture. This picture had a value of 1 on the pixel where our color-values were good, and a value of 0 where they were bad. This resulted in a white blob where almost every white pixel was on our object. Something like figure 4

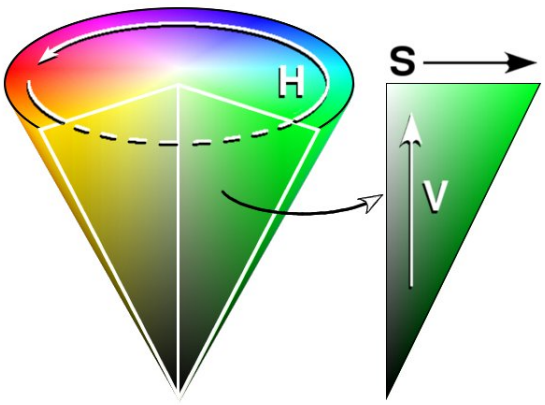

Figure 3: Hue (H), ranging from 0 to 360 degrees; Saturation (S), ranging from 0 to 100% and Value (V) ranging from 0 to 100%

Figure 4: An example of our image after the first processing

3.2 Tracking The Object

3.3 Picking Up The Object

4 Results

5 Future Work