

Ngoc An, Ha 967813 Nam Hung, Le 967826

7/13/2015

Summer Semester 2015

Introduction

*Automatic control Parrot AR Drone*

The goal of this project is to control the AR Drone 2.0 automatically by using computer application. This project is written in C# by using control library support for the drone.

# Introduction

## Automatic navigation for AR Drone

In the last few year, the development for quadcopter has risen to become a trend. In order to expand the research field, a better maneuverability, along with automatic hovering ability control for the quadcopter will provide a lot more usage.

This project will attempt to make the drone moving straight from a starting point, and then locate the goal by using its front camera to track a landing mark. Finally, the drone will try to land near the goal as much as possible.

In order to develop this application, we use AR.Drone control library for C#, which was created as basic helper for developers who prefer to use Visual Studio as programming application. For the image processing part, we try to use Hough transform algorithm to detect the black circle as our landing goal.

## Requirement

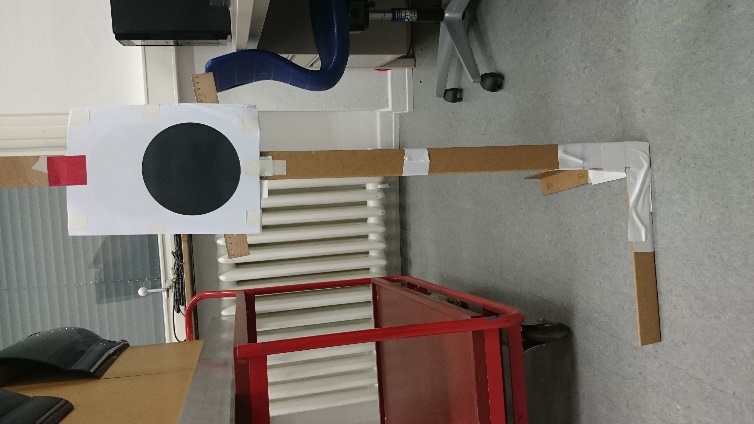
 

Figure 1 the landing mark Figure 2 the drone

# Flying process diagram

Cruising phase

Detection Phase

Hover a certain of time

Land

Still see the mark ?

Detect the black circle

Calculate flying time

Moving forward

Input distance

# Solution

## Cruising phase

In this phase, the drone will fly straight from the starting point to the detection area. Since the camera is not good when streaming, our detection radius is at best 1.5 meter from the mark. The cruising phase will make the drone flying straight from the starting point and move into the detection range.

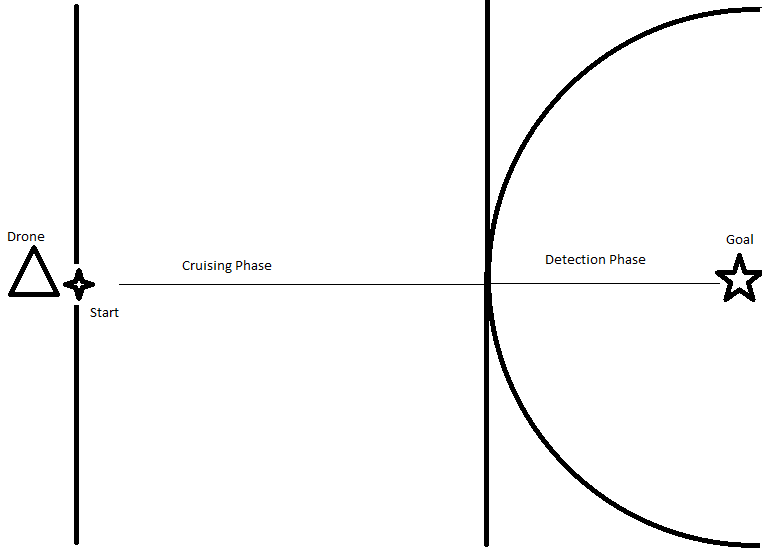


Figure 3 Map of experiment scene

The process is simple. First, we calculated the speed of the drone, by measured the distance and the time the drone took to fly, and then put it in a linear simple velocity equation. After several experiments, we had the drone average speed is about 0.7 m/s. So from then, user can input the initial distance they want and the drone will calculate the best time (T) it needs to reach the detect area.

After that, we program the drone, so it will keep moving forward (T) seconds from starting point, reach the radius 1.5 m zone, and then the drone will change to the detection mode.

## Detection phase

­In the process of experiment, we had tried two methods of automatic navigation for the drone: One with the front camera and the other with the vertical camera.

### Using the vertical camera to detect landing symbol

At first, we attempted to use the camera below of the drone to detect the pre-integrated land pad as the goal for the drone land on.

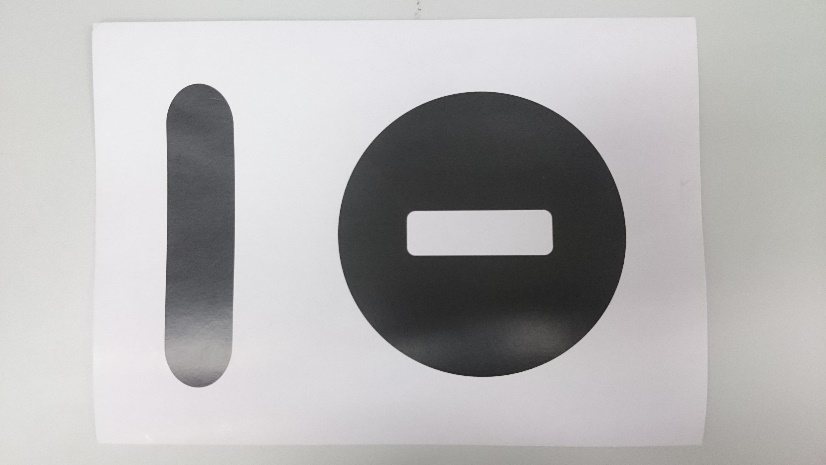
.

Figure 4 the original land pad comes along with the drone

Advantage

* The drone, once hover on this landmark, within its own firmware, will automatically land in the center of the pad.
* It is easier to detect the pad through the camera, with fewer noises.
* The drone has a sensor below, which keeps the flying height constant, meaning that the distance between the camera and the pad will not distort the image of the pad.

Disadvantage

* The delay of the data transfer between the drone and the computer is too high and unstable. It affects the landing command and makes the drone unable to hover exactly up above the pad.
* Through experiments, it is difficult to keep the drone flying straight. In addition, the cover range of the camera is small, so once the drone misses the pad due to the time delay or unstable flying process, the drone could not move back.

### Using the horizontal camera (front camera) to detect the landing mark

Because using the vertical camera was proven difficult and unstable, we moved to using front camera. For that reason, we make an impromptu landing mark and put it in the goal. To get the best result, we had to fix the flying height of the drone.



Figure 5 The landing mark

It is not recommended to use two cameras at the same time, because it will increase the delay and makes the connection between the application and the drone slow. Moreover, changing the camera channel between processes is also not advised, because there may exist the case when the drone knocks down the landmark due to the moving delay. Therefore, our final solution is program the drone so it will move to the mark as close as possible and then land itself there.

Advantage

* Wider camera range allows the mark always trackable.
* The flying delay does not affect the mark tracking much.
* The height of the tracking mark is fixed, so it is easier for navigation.

Disadvantage

* The front camera caught more noises compare to the vertical camera.
* When the drone is too near the mark, it is more difficult to track the mark.
* The drone easy pick up other points in the scene as new tracking mark.

Temporary solution

* In order to get fewer noises, we use white tape to cover some part of the experiment scene.
* When the drone misses the target while being too close, we let the drone land after a certain of time.



Figure 6 The original color is dark blue, then we use white tape to cover it.

# Algorithm Explanation

## Image Process Algorithm

## Goal tracking algorithm

We have two parameters as input for the control machine. Since the drone is unstable, we can only hope to achieve a certain range value of the parameter.

### For horizontal navigation

Center X is the coordinate in the horizontal direction in our geometry system, with the origin point is the most left point of the monitor of our application. The desire point is at the middle of the screen, which is 340 through measurement.

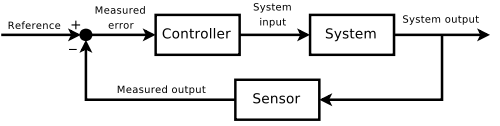
We use center X as the control input to satisfy the following condition



Moving

Direction, speed

Center X



Speed control algorithm

Drone

Camera

Measured error here is the checking condition, in this situation:

* Center X larger than Max Horizontal: the drone move left.
* Center X smaller than Min Horizontal: the drone move right.

For horizontal navigation, due to the delay connection, the difficult part is how to control the moving speed of the drone. Through experiments, we concluded that using a constant speed is not a good choice. Therefore, we had to derive a new equation, which will allow us changing the speed base on the feedback center X.

Firstly, we have to calculate the distance between center X and the desire point by the equation:

distance =/center X – desire/;

Then we calculate the changing speed by the equation:

Speed = 0.3 \*

Because the drone have a minimum speed in order to move, we set the min value of speed is 0.1 if the value is smaller than 0.1.

### For forward – backward navigation

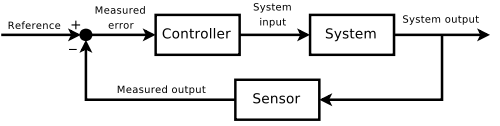
We use radius R as the control input to satisfy the following condition



Radius R

Direction

Moving



Drone

Camera

Application

Measured error here is the checking condition, in this situation:

* Radius R larger than Max forward: the drone move backward
* Radius R smaller than Min forward: the drone move forward

# User interface

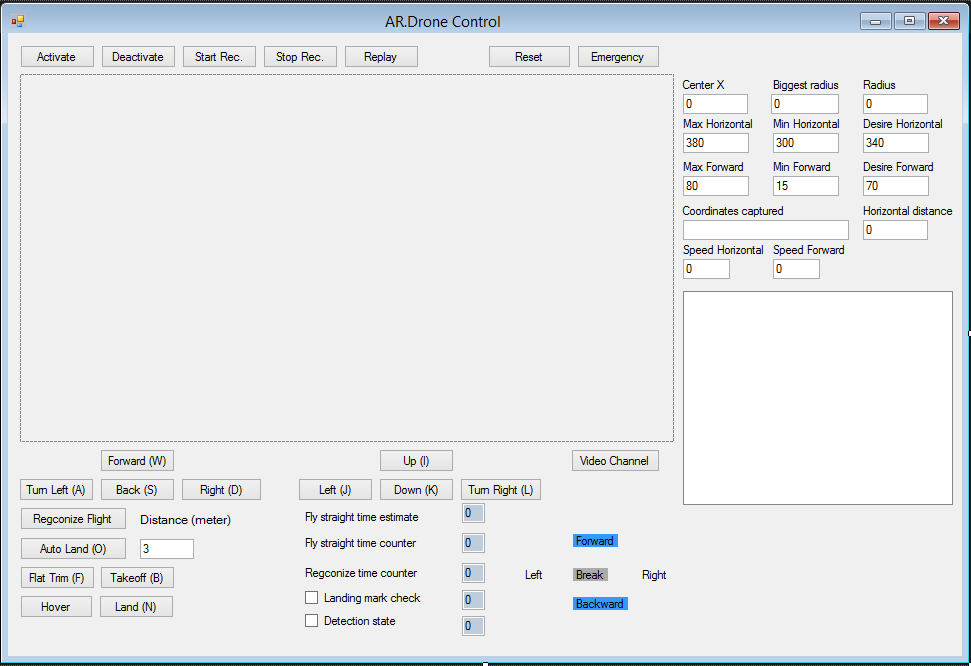


Figure 7 The control application interface

This is our application interface. Here we will explain only our implement function:

|  |  |
| --- | --- |
| **Button** | **Function** |
| Recognize Flight | Only detection phase |
| Auto Land | Our whole process: both cruising phase and detection phase |
| Detection state | Check if the drone is in detection phase |
| Landing mark check | Condition to land the drone after detection phase |
| Recognize time counter | Condition to start detection phase |
| Fly straight time estimate | Estimate cruising time base on distance |
| Fly straight time counter | Timing the cruising phase |

To track the flying direction of the drone when in detection phase, we also put in five labels indicated as “Left”, “Right”, “Forward”, “Backward” and “Break”.

We also displayed various values for the parameter and condition, in order to keep the drone on track and debug the application.

# Result

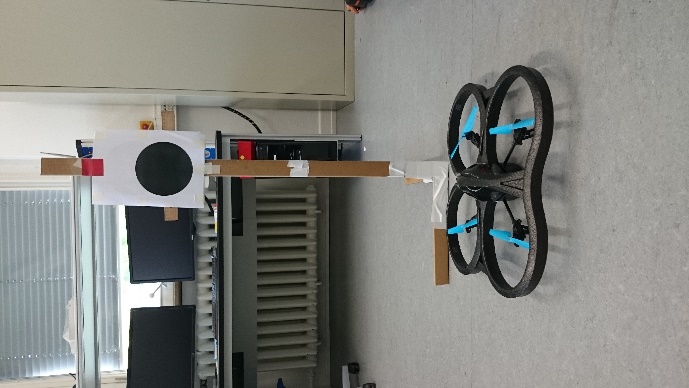
 

Figure 8 Good result Figure 9 Not so good result

## Good result

The drone landed directly in front of the landmark, meaning that the detection phase worked well.

## Not good result

The drone landed close to the landmark, meaning that the detection phase has problem when keeping the center in check.

Table of Contents

[I. Introduction 1](#_Toc424727078)

[A. Automatic navigation for AR Drone 1](#_Toc424727079)

[B. Requirement 1](#_Toc424727080)

[II. Flying process 2](#_Toc424727081)

[III. Solution 3](#_Toc424727082)

[A. Cruising phase 3](#_Toc424727083)

[B. Detection phase 4](#_Toc424727084)

[1. Using the vertical camera to detect landing symbol 4](#_Toc424727085)

[2. Using the horizontal camera (front camera) to detect the landing mark 5](#_Toc424727086)

[IV. User interface 6](#_Toc424727087)