Reconnaissance Drone

Name: Qiyuan Qiu

Contents

1 Abstract

Robot navigation in unknown environment is of key interest for many reasons in artificial intelligence. In this paper I propose a solution to autonomous navigation problem for drone in indoor environment using a single camera under real time constrain. The problem can be decomposed into three parts – location detection, trajectory determination and drone action control. To determine drone's relative position to surrounding environment, pinhole camera model is used; To plan a save and efficient trajectory, optical flow theory is developed; To finally update drone's location in real-time a robust and pragmatic optical flow algorithm called Pyramidal Lucas-Kanade is used. In the trajectory plan sub problem, two different obstacle avoidance strategies will be discussed, implemented and eventually tested in real world scenario. AR.Drone SDK is used to control drone actions.

2 Introduction

Drones are capable of performing a wide range of real world tasks that cannot be done by human beings from military missions, remote imaging, surveillance, environmental monitoring to carrying a transponder or repeater for wireless communication [1]. In all these applications, autonomous navigation plays a very important role. Hence the problem of navigating in an unknown environment is of key interest to people.

Real time navigation is easy for human being and animals yet it is very hard for drones to do so, especially in known environment. To navigate autonomously and search for a target pattern, the robot should be capable of sensing and interpreting surrounding environments to extract information of its own position, computing a real-time path to the target, choosing directions and speed carefully and efficiently to reach the target location without hiding into any obstacles.

One way to achieve autonomous navigation is to reconstruct the surrounding environment and use the reconstructed model as a map to make moving decision. This is usually known as Simultaneous Localization and Mapping problem or SLAM for short. Since I intended to achieve navigation under real-time constrain, reconstructing the original map is too cumbersome in this case. It typically requires other vision depth detection sensors[2] [3] [4] on top of a camera which can only provide a stream of 2 dimensional images work as a projection of 3 dimensional real world onto a camera plane.

Another group of algorithms based on optical flow work really well with single camera constrain. The basic idea is to use the video stream to compute the movement of pixel intensity in any two successive frames. With gradient information of the images, we are able to define a magnitude associate with the movement of the intensity. Then we are able to tell how fast each pixel is

moving. Because the intensity is related to objects, so we are aware of their movement in real world.

In this paper, I propose an algorithm to achieve real time autonomous navigation. The problem can be decomposed in to three parts: location detection, trajectory plan determination and drone actions control. These three sub problems can be restated as “where I am “,”where I am going”, ”how do I get there”.

The most important information extracted from the video stream sent back by the drone is the *Optical Flow* field. Every pixel has an intensity value associate with it. When an object is moving, its objection on the camera plane is moving accordingly. Optical flow is an algorithm to tell the direction and magnitude change of the intensity from successive images. There are multiple pragmatic methods to calculate the optical flow; one very good one is called the Pyramid Lycas-Kanade [5]. OpenCV library includes a very handy way to make sure of this method. For the purpose of navigation algorithm developing, I will use OpenCV’s solution but with care.

With the optical flow field I discussed two obstacle strategies. One is the balance strategy the other visual steering strategy. The balance strategy is based on the idea that optical flow is greater for close object and smaller for distant ones. The Visual steering strategy is based on vision region division and a “robustness” function that evaluates how robust each one of the direction is.

With AR.Drone API, I will be able to control the drone’s movement precisely in real time. All computation will be processed on a host computer. Video stream is fed to a computer and drone action command is issued by it too.

All programs will be tested individually and assembled together in the end. Experiment will be carried out in different indoor environments. Different obstacle avoidance strategies will be implemented and test in order to find a best one that suits my objective.

3 Navigation Algorithm

To achieve autonomous navigation, three basic problems should be solved. First, the quad rotor should know its current position with regard to its surroundings. This can be referred to as “where I am” problem. Second, the done shall be capable of computing a trajectory path real-time. This can be referred to as “where am I going” problem. Third, the drone shall be able to control its direction and speed to arrive at target position. This can be referred to as “how do I get there” problem. If these three sub problems-“where I am”, ”where I am going” and “how do I get there”- are solved, the overall goal of autonomous navigation in an unknown environment will be achieved.

3.1 Location Detection Using Camera Model

This is the *"where I am"* sub problem so it can be restated in the following way. Given a sequence of images captured by a camera, how can the drone tell how far away it is from surround objects? An image is essentially a two dimensional mapping of a three dimensional world. Assume there is a point (X, Y, Z) in space; it has a counterpart projected on image at point (u, v). The relationship between them can be determined by the following equation.

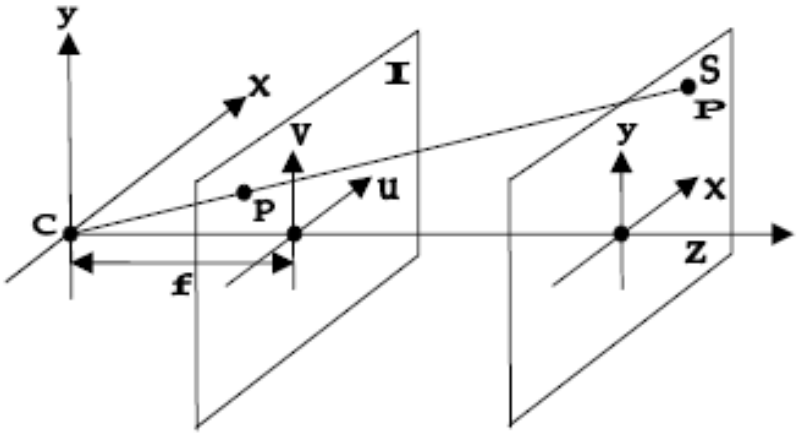

Figure 1: Pinhole Camera Model

$$u = f\frac{X}{Z}$$

$$v = f\frac{Y}{Z}$$

where f is the focal length of the camera.

3.2 Trajectory Determination Using Optical Flow

This is essentially the *"where I am going"* sub problem so this can be restated in the following way. Given a position in space, taking robot's own current position into account and make use of all other possible information such as *Optical Flow Field* [6] to come up with a path that will lead to the destination safely and effectively under real-time constrain. When it comes to trajectory path constructing, there are currently two strategies discussed here. Since strategies used here are essentially dealing with obstacle avoidance issues, more efforts should be put into developing a case specific algorithm to solve the problem addressed in the Introduction. In other

words, the drone should be capable of constructing a path to find target yet without hitting into obstacles under real-time constrain.

3.2.1 Optical Flow Theory

Every pixel has an intensity value associate with it. When an object is moving, its objection on the camera plane is moving too. Optical flow is an algorithm to tell the direction and magnitude change of the intensity from successive images [7]. All the following material relies heavily on optical flow field.

Barron and Thacker developed a tutorial [8] of how to calculate the optical flow.

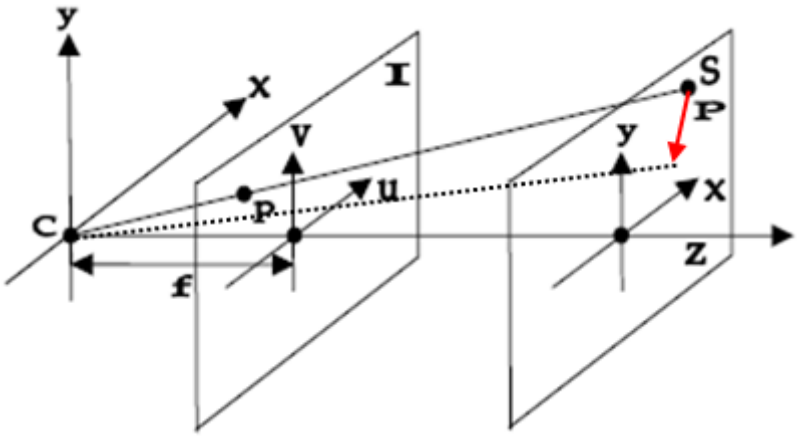

Figure 2: Optical Flow Calculation

Consider a frame taken at time t, for a point, optical flow algorithm claims that there will be a corresponding intensity $I(u, v, t)$. At the ensuing time $t + \Delta t$ there will be intensity $I(u + \Delta u, v + \Delta v, t + \Delta t)$. Since the time change Δt is really small, we can safely assume that they are equal.

$$I(u + \Delta u, v + \Delta v, t + \Delta t) = I(u, v, t)$$

Apply Taylor expansion to both sides claims that:

$$I(u + \Delta u, v + \Delta v, t + \Delta t) = I(u, v, t) + \frac{\partial I}{\partial u}\Delta u + \frac{\partial I}{\partial v}\Delta v + \frac{\partial I}{\partial t}\Delta t + \cdots$$

For very small transition, high order terms can be ignored.

$$\frac{\partial I}{\partial u}\Delta u + \frac{\partial I}{\partial v}\Delta v + \frac{\partial I}{\partial t}\Delta t = 0 \text{ or}$$

$$\frac{\partial I}{\partial u}\frac{\Delta u}{\Delta t} + \frac{\partial I}{\partial v}\frac{\Delta v}{\Delta t} + \frac{\partial I}{\partial t}\frac{\Delta t}{\Delta t} = 0 \text{ finally:}$$

$$\frac{\partial I}{\partial u}v_u + \frac{\partial I}{\partial v}v_v + \frac{\partial I}{\partial t} = 0$$

$$\nabla I \cdot \vec{v} = -\frac{\partial I}{\partial t}$$

$\vec{v}$ is the optical flow at pixel(u,v) at time t.

There are many method can be used to calculate Optical flow field for successive images. OpenCV provides an algorithm called Pyramid Lycas-Kanade . The idea of this method is to down sample images to multiple smaller layers and find patterns at the smaller layers as a guess. Then use the guessed result to combine with lower layers to come up with an accurate optical flow field.

Given an optical flow field, we are able to tell where obstacles with regard to the drone are. Two optical filed based obstacle avoidance strategies are discussed.

3.2.2 Balance Strategy

This strategy developed by Temizer [7] is simple and straightforward. Given the knowledge of optical flow field information, simply add up all the flow on the left hand side and right hand side to get two flow sums V_{left} and V_{right}. Then the control strategy is to balance the optical flow on both sides.

On the contrary, if we want to track an object, the drone should turn to the side that shows more motion.

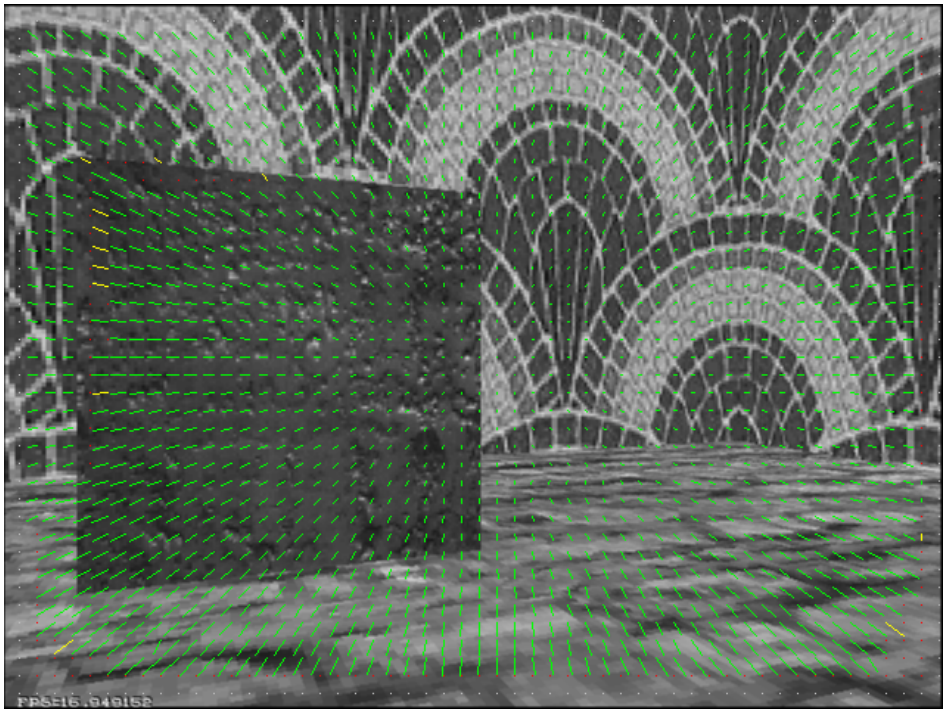

Figure 3: Optical Flow Field

The sum of the left half optical flow is greater than that of right; the drone should turn right to avoid collision.

3.2.3 Visual Steering Strategy

Yuan, etc. [9] developed a visual steering strategy without global reconstructing of drone's surrounding environment. Figure 3 shows that vision of the drone is divided into five directions and devised a function to evaluate the "robustness" of each single direction. Then the drone should always pick the direction with the highest "robustness".

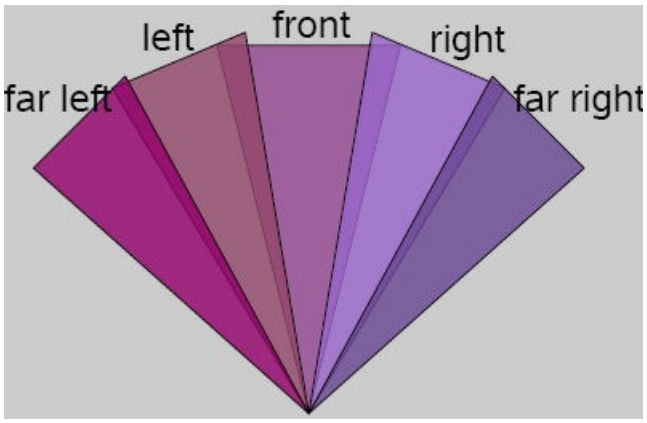

Figure 4: Visual Direction Division

Visible region are divided into five groups

3.3 Drone Action Control Using AR.Drone SDK

From section 3, a direction is

4 Implementation Discussion

4.1 AR.Drone SDK

AR.Drone is a quad-rotor designed by a French company Parrot. The SDK provides API that simplifies the programming of drone actions. Through the API, video stream, drone action can be handled in a more friendly way.

AR.Drone's latest sdk [10] provides a very good interface to control the drone action. It also provides a framework to simplify the procedure of custom code integration with its onboard firmware. Since the purpose of this paper is to develop a portable algorithm based only on vision information, its simplicity will help reduce unnecessary mistakes thus speed up the development. For this reason, AR.Drone is picked as the platform for experiment.

4.2 OpenCV vs MATLAB

OpenCV stands for Open Source Computer Vision Library [11].

This Library is mainly used in real time computer vision. It is written in c/c++ . Similar work can be done by MATLAB computer vision toolbox. There are multiple reasons to choose OpenCV. Matlab has a better memory management and more smooth learning curve but very slow compared with OpenCV. In addition, even though OpenCV is difficult to debug and requires programmer to manage memory wisely and explicitly, its fast speed, vast support in open source community outweigh the MATLAB.

4.3 Overall Procedures

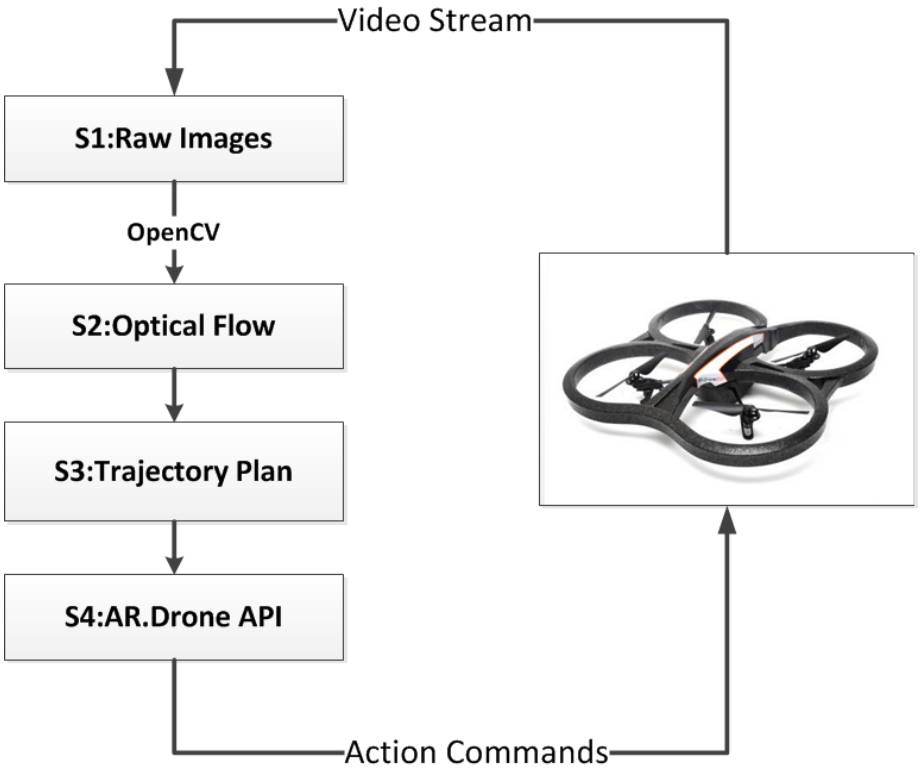

Figure 5: Data Flow Diagram

To implement the algorithm discussed in section 3. I will use the above structure. The AR.Drone is a Wi-Fi access point. The video information will be transmitted through Wi-Fi to the left side. The left side is an pc running a Ubuntu 12.04 LTS 32bit operating system.

Before doing any image processing, the OpenCV library will need to be set up properly, the AR.Drone API should be install and set up properly as well.

After the pc environment is set up, a control program will be written in C programming language for its efficiency. The control program will fist fetch raw image and render it with OpenCV library to acquire the optical flow field information. This corresponds to S1 and S2 in the above figure. This information will be used to compute obstacle avoidance algorithm. Finally the program will find a path that is both safe and efficient. This corresponds to S3 in the above figure.

To actually go to the target destination pointed out by the path generated in S3, one will need to translate the path information into AR.Drone action command. This can be done leveraging AR.Drone API. This corresponds to S4 in the above figure. Once the action commands are received, the AR.Drone will move accordingly.

To examine the effectiveness of the algorithm, the AR.Drone will be test in an unknown indoor environment which, in this case, can be a school building. Multiple tests will be carried out in same building. Repeat experiment will also be designed and conducted in different school buildings. If the AR.Donne is able to find the desired pattern, then one should believe the strategy will work. Otherwise, new approach should be proposed.

It is very likely that the obstacle avoidance strategy in S3 will take more time to tune up, hence if the currently discussed algorithms do not work, more robust method should be developed and test as soon as possible. I am optimistic about the existence of such a functional algorithm because the indoor environment allows me to make some assumption like straight ling movement which can help reduce the complexity of the overall problem.

5 Budget

Item	Price	Note
AR.Drone	$299.95	Will be used as the robot
Battery	$41.00	Two battery will help keeping a continuous test
Total	$340.95	

6 Timeline

Month Number	Short-term Objective	Note
1	• Read papers • Familiarize with OpenCV Libriary	
2	• Code optical flow • Test optical flow with different objects in different indoor environment	
3	• Code obstacle avoidance strategies • Test different strategies	
4	• Code drone action control • Integrate action control with trajectory plan • Test the drone in a slow mode	
5	• Tune the overall program to smooth the drone's performance • Test the overall algorithm in different real time environment	
6	• Experiment record review • Final presentation	

7 List of figures

8 Reference

[1] G.L.BARROWS, "Future Visual Microsensors for Mini/Micro-UAV Applications," in *7th IEEE International Workshop on Cellular Neural Networks and their Applications*, 2002

[2] G.Dissanayake, P.Newman, S. Clark, H.F. Durrant-Whyte, and M.Csorba. *A solution to the simultaneous localization and map building (SLAM) problem.* IEEE Transactions of Robotics and Automation, 2001.

[3] J.Miller, H.Plinval, K.Hsiao *Mapping Controured Terrain: A Comparison of SLAM algorithms for Radio-Controlled Helicopters*, Cognitive Robotics, Spring 2005.

[4] Andrew J.Davison, *Real-Time Simultaneous Localisation and Mapping with a Single Camera*, Robotics Research Group, Dept. of Engineering Science, Univerisy of Oxford, UK.

[5] Lucas,B., and Kanade, T., *An Iterative Image Registration Technique with an Application to Stereo Vision*, Proc. of 7th international Joint Conference on Artificial Intelligence(IJCAI), pp. 674-679.

[6] Ted A. Camus, "Real-Time Optical Flow", Ph.D. Thesis, May 1995.

[7] Optical Flow Based Robot Navigation, http://people.csail.mit.edu/lpk/mars/temizer_2001/Optical_Flow/

[8] J.L.Barron and N.A.Thacker. Tutorial: Computing 2D and 3D Optical Flow, http://www.tina-vision.net/docs/memos/2004-012.pdf

[9] C. Yuan, F. Recktenwald, and H. A. Mallot, *"Visual steering of UAV in unknown environments,"* in IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS 2009), 2009, pp. 3906–3911.

[10] ARDRONE open API platform, https://projects.ardrone.org/

[11] OpenCV, http://opencv.willowgarage.com/wiki/