Object Tracking Using Autonomous Quad Copter

Carlos A Munoz, Advisor: Dr. Tarek Sobh
Robotics, Intelligent Sensing & Control (RISC) Lab.,
School of Engineering, University of Bridgeport

Abstract

This work explains an approach to implement an autonomous ball catching quad copter by combining object detection and trajectory calculations. To accomplish this, a quad copter is used to autonomously catch a ball by dynamically calculating trajectory and the interception point using two Kinects. This approach is tested in a two-step process. First successful test is done on a flat surface (table) and the second test is done in the air.

Introduction

The goal of this work is to create an autonomous robotic system that can attempt to simulate a human catcher. This is implemented on the quad copter shown in Fgure 1. The idea came from watching kids always playing, throwing the ball to a wall or to the air and having to catch the ball themselves. What if we can use a robot to pick the ball up or hit the ball back automatically based on location of the ball? This can be useful in areas like golf, where “bots” can be sent to pick all the golf balls and return them back to a central location. Or a concept for a bot that acts like a dog and catches the ball in the air. The objective of this work is to implement such an autonomous system .

Hardware

A. *Quad Copter*
In this work, there are three main hardware modules that are used in conjunction to achieve the end result. These parts are one quad copter made by 3D Robotics and two Kinects made by Microsoft. The quad copter contains four motors with four blades. Two blades spin counter clockwise and two that spin clockwise. There are 4 ESC’s (electronic speed controllers) that govern the speed of the motors. The quad also comes with a power distribution board, a battery (4200mAh) and an APM (auto pilot module) that combine with the GPS one can achieve autonomy via setting a route on a map for the quad to go and come if needed be. There is also a telemetry radio module that is constantly sending information (altitude, speed, location battery level, etc.) from the quad to the pc and another telemetry radio on the PC connected via USB that sends pitch, jaw, and roll alterations to the quad. They are both transmit /receive modules.

Fig. 1.Quad copter used in this work

Figure 2: Microsoft Kinect.

B. *Kinect*
The Kinect cameras (model shown in Figure 2) are made by Microsoft. They contain three main functions. There is a depth stream that is mainly used to give an approximation on how far an object is from the Kinect. There is also a skeleton tracking module that is mainly used to detect humans in front of the Kinect. There is also a regular color stream (RGB) that can be use like a regular camera. The camera’s color stream feature is what it is used to gives us the XYZ-coordinates that are needed to accomplish this work.

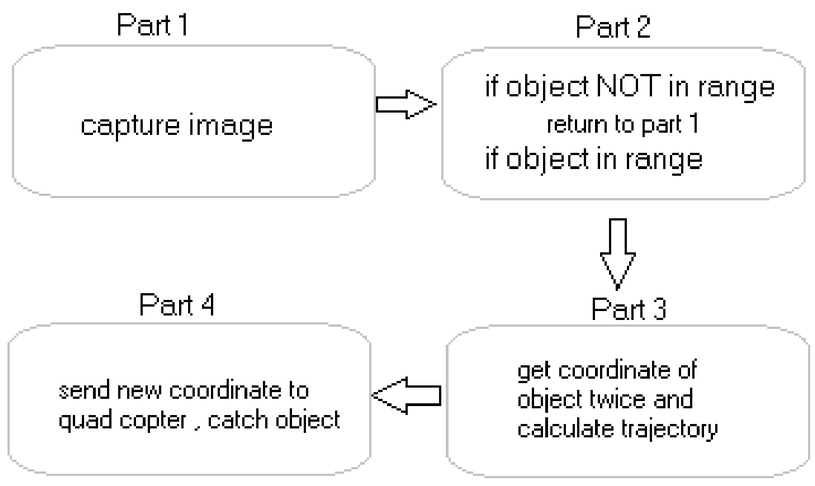

Figure 3: Tracking Algorithm

Methodology

The work is broken into four main modules. The first part is to simply grab the latest image from the two Kinects. The second module checks to see if the object that we want to track is in the images that we grabbed. If the objects are not found then we simply discard the images and get new ones. In the third module, we are first checking if the primary object (the ball) is within an initial distance from the quad. If it is, then a few milliseconds later, we take another image and use them both to acquire the expected trajectory. This trajectory is then sent to the quad that will in turn intercept the object. These are the basics of the work as show in Figure 3.

Image Processing

The images gathered by both Kinects are taken as raw 30 frames per second color streams. These are then converted to images of type CV. These images are then converted to the HSV color space. The reason that is done is because intensity is easier to use to separate the object from background noise/color. Once the HSV upper limit and lower limit have been obtained, a gray image gets created based on those two values.

Now that intensity only exists in the image, we can start looking for shapes (contours). Rectangles are extracted using boundaries for height and width. If rectangles are detected then the object that we are trying to locate has been found and it is shown with the filled rectangle on the screen. This is the approach used over and over per frame to attempt accuracy on the location of the object while moving in the space provided.

A. *Object detection*
One thing to note is that the recognition of the object works best if done by intensity rather than color (RGB). Using the HSV values (hue, saturation, and value) the objects can be added to the array of objects that are going to be searched. Each one is added using the visual box in the program. The image that is initially black and white, shows the pictures in real time and by sliding each value, the object becomes more apparent (stays white) while the background fades (goes to black).

Fig. 4.Green ball added to the program for continuous detection

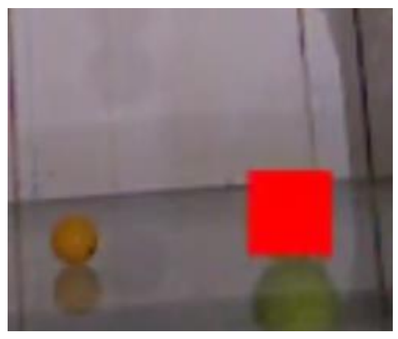

Figure 5. Green ball found, red box implies detection

These values are essential and they need to be as accurate as possible since they are used in combination with the threshold value to determine if the object that we are tracking is in the frame that is being grabbed from the Kinect. Most of the code to achieve this is done using *emguCV* .Net wrapper for the C# language.

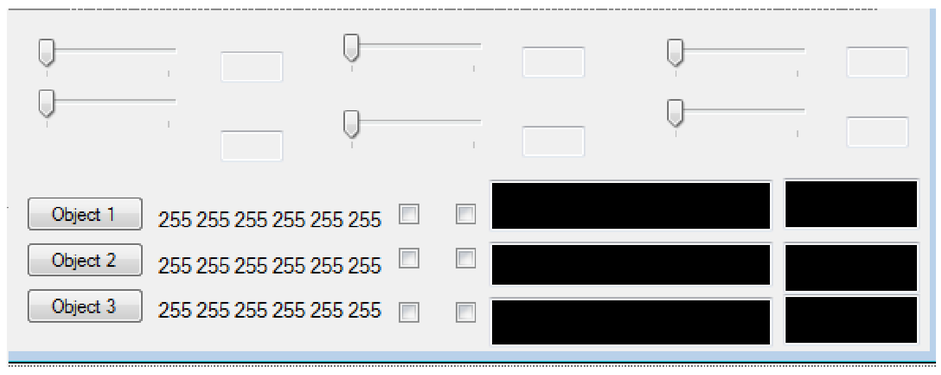

Fig.6. Six sliders, two for each HSV upper and lower limit values, can track 3 objects at the same time if needed

B. *Threshold values*
These values can be tweaked and they are based on the HSV values of the object that is going to be tracked. At the beginning of the program, the user can see a rectangle showing the object being tracked after the values have been stored. This rectangle only shows if the threshold value satisfied the range (set up distance value for the object).

C. *Coordinates*
The coordinates of the objects are represented in XYZ values. Two of the values (X and Y) are given by the first Kinect. The third value (Z) is given by the second Kinect (its own X value). For the first test only the x and y value are used since the ball is rolled on the table. For the second test all three values are used. These values are in relation to the size of the ImageBox used in C#. The real values have to be calculated based on the distance of the camera in relation to the ball and the quad copter. These values will vary depending on the length of the surface (for the first test) and the distance of the quad (for the second test).

Conclusion

In this work, an autonomous quad copter was implemented to capture a ball. Calculations to create a quad copter behave like a human and intercept an object is just one very small example of what autonomy can do and how it will impact us in the not so distant future. Great things are possible and hopefully this example can benefit and inspire someone to take this work and expand it to next level.
More information including in depth explanation, demo videos and pictures please visit: https://sites.google.com/site/carlosmrobotics/home