Christoph von Braun

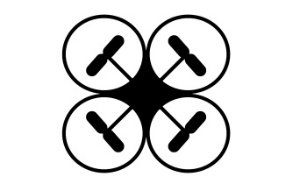
CS 4243 Writing Assignment 2

10/24/15

The project at hand will be a drone recognition and tracking suite. Ultimately, four cameras will be placed around the corners of a room, monitoring a set space in between. A quad-copter, piloted either by code or by hand, will be set to fly around that space, in some pre-determined path or pattern. The camera feeds will be send back to my main program, which identifies the drones in that three-dimensional space, plots that information visually, and determines if the flight pattern matches the pre-determined one, in order to spot a rogue drone or pilot error. In a grander vision amongst two others’ projects, if a flight error is detected, an ability is activated that will swap the drone’s controller in real time, while it continues to fly unhindered from another host. A rudimentary diagram of the above components is on the next page.

This project is slightly unique in that it does not need direct interaction from a user. Furthermore, once the program starts, there is no way in which a user could communicate with the system at all. In fact, the entire point of the combined projects is that the entire system works autonomously, without the need for any human intervention. The system should find the drones, track the drones, determine if there is an error in the flight controller, and swap that controller to another one with a known, correct state, all automatically and in more-or-less real time. However, for the purposes of Senior Design, some adjustments must be made for both demoing purposes, and as a failsafe in case one or both of the two other team members’ projects goes awry. My project has thus been compartmentalized, so that it may stand on its own for the time being.







The first such modification is the inclusion of a user interface. For the sake of having something to demonstrate when need be, all the tracked information regarding the drone’s position will be stored and displayed visually. As an intermediate step, only each individual camera’s two-dimensional stream will be recorded, specifically the drone’s x and y coordinates on that camera’s viewport. This information will be presented via a graphical representation of the camera feed, which will show a line indicating past and current behavior of the drone. Each of the four camera feeds will be presented in this manner, in four separate windows. Ultimately, the user interface will be modified to accept three-dimensional historical data, at which point showing separate camera feeds is no longer necessary. It is of note that due to the autonomous nature of the project, the interface will accept no user inputs of any kind.

The code itself is subdivided into two major chunks: the image processing and recognition, and the room model generation. Other minor components include the rudimentary user interface, the communication between the image component and the model component, and the communication to the other group members’ projects.

Operating individually on each camera stream will be a feature-detection algorithm called SURF, which analyses a series of known photos of the drone, and compares those to the incoming feeds, to detect matches. With sufficient pattern matches, the algorithm will confirm that a drone has indeed been found. If multiple cameras detect the same drone, that information can be combined to calculate the drone’s actual position in the room. The drone’s position on each individual camera is then relayed via the communication module to the mapping module, where the next step begins.

Depth will be added to the camera feeds by an algorithm that will take in the relative locations of each viewport, and combine that data into a three-dimensional map of the room, in an expanded form of a process known as stereopsis. Instead of four separate streams at this point, there will only be one representation of the room in an x-y-z coordinate space. This will ultimately be the only required form to display, although the details of how to display such a space are still being formulated. That location data can then be sent over to another team member’s component, where it may be analyzed for deviations from the expected position.

Due to the nature of multiple team members operating separately on an ideally interwoven project, my individual project will be tailored to operate more independently. Such is the case with the drone control system, as another team member is explicitly handling that aspect. I will not be requiring the drone to run autonomously, but rather manually, with either a game controller or iPhone app. Another sacrifice will be in how the program detects deviations from a set flight path. For the sake of demonstrations, I will be manually including an expected flight pattern, then deliberately control the drone away from that pattern, to test if my program detects that change. In an ideal case, I would have the expected pattern come from another member, and the autonomous drone would be given a foreign piece of code to send it on a rouge path. For my simplistic version however, the final effect will still be the same.

As mentioned earlier, there is intentionally no user input required for the entire project, and no explicit user either. The only front-end interface occurs via displaying the drone’s historical location, but this will accept no information. As a result, the interface will be extremely rudimentary. As this is a non-functional requirement for the project, it will be as basic as possible: just one window with one graph. For intermediate builds, the camera’s direct feed will be displayed, as well as a box around the drone if detected. This information is only for testing purposes, and will be eliminated for the final build.