**Software Design:**

At the highest level, the software for the security system runs according to the flowchart below.

The cameras that are being used for motion detection are run in a continuous loop that waits for an intruder to trigger the image subtraction software. Once an object is detected, an image processing algorithm is used to determine the location of the intruder within the room. This is assisted by having stationary cameras in the same location as the UWB sensors to accurately read the relative location of the camera within the room.

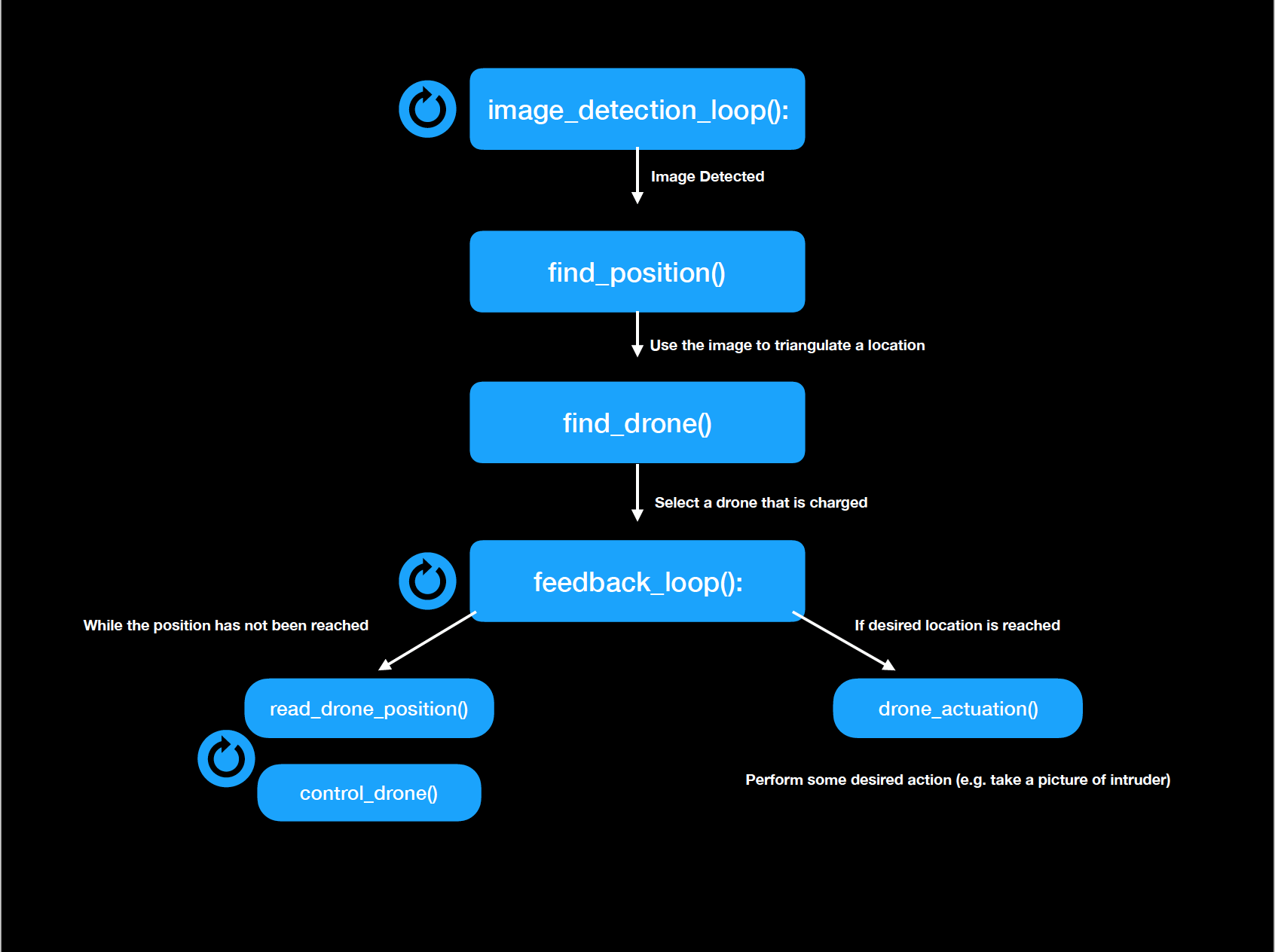


Figure X: Software Flowchart

Once the location has been computed, the round robin selector will choose a drone with a sufficient battery charge. The drone then enters a feedback loop with the central processor (most likely a desktop computer). The computer will connect to the ESP-8266 WiFi chip on the selected drone.

The drone can now be sent commands over WiFi using the Plutox Drone Api. This feedback loop will take a reading using the tag on the drone and the UWB sensors, compare that value to the desired location, send a command to the drone to correct the heading, then repeat. This process is repeated until the drone is within a specified range of the target (close enough to take a picture, or perform some other type of actuation).

**API:**

The functions are documented here:

**image\_detection\_loop():**

This will be the loop that is always running while the drones are idle. This is part of a python program that is using the process of image subtraction to detect intruders. An arbitrary number of 720p HD cameras *n* (6 in our system) will be feeding frames into the program. The program compares matrices of pixels using a common image subtraction algorithm to detect a change in the scenery (similar to what is done in this paper <https://ese.wustl.edu/ContentFiles/Research/UndergraduateResearch/CompletedProjects/WebPages/sp14/JonathanAllenSiwei/ESE498Report_Chiang_Chu_Su.pdf>*)*

**

**Figure X: Image Subtraction Program (unoccupied room)**

**Figure X: Image Subtraction Program (occupied room)**

**find\_position():**

We are attempting to use a python program that uses image processing as a means of detecting a location from a picture. This will be called after it is confirmed that an intruder is detected. The OpenCV library has been used to complete this task, as shown in this paper <http://sciencejournals.stmjournals.in/index.php/RRJoPHY/article/download/246/746>. Using light sources and keeping track of camera angles allows basic laws of triangles to be used to determine a position from pixels in an image.

**find\_drone():**

This function selects a drone that has a sufficient battery charge and if possible is close to the detected location of the intrusion. The PlutoX drone api has a system call to check the battery levels of drones and the charging stations and Drones have known locations due to the UWB sensors. This makes this process of selecting a viable drone straightforward.

**feedback\_loop():**

Once the system has detected an intruder, estimated their location, and selected a drone, we will use a closed loop system to guide the drone to the desired location. Using the UWB’s and the location tag on the drones, a C++ program will run to constantly read the current position and then perform some actuation on the drone to move closer to the source. Using the PlutoX api, commands are sent over WiFi to the ESP-8266 chip on the drone. For example, if the drone’s location is 20 meters from the intruder in one direction and 15 meters in the other, the drone might be given a command to orient itself in the direction and to move towards it (pitch\_left(), thrust\_forward(), read\_position(), etc.).

**drone\_actuation():**

Once the drone is within a specified distance of the target, some action is performed. This could possibly be taking a picture of an intruder, or blinking a light and sounding an alarm. This will be open ended. After the drone has performed this action, it will fly back to its charging station using the previous feeback\_loop.