PEX 02: Mission & Target Acquisition

Due @1030 on 05 April

Help Policy

AUTHORIZED RESOURCES: Any, except another cadet's assignment or published solutions to the assigned problem.

NOTE:

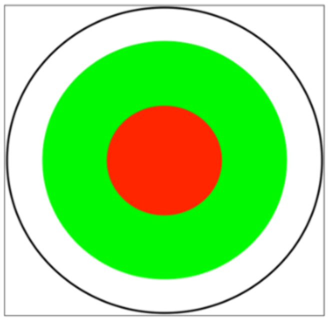
- Never copy another person's work and submit it as your own. Here are a few blatant examples of copying:
    - Making an electronic copy of another cadet's solution and then modifying it slightly to make it appear as your own work.
    - Reading a printout or other source of another cadet's work as you implement your solution.
    - Completing your entire solution by following explicit instructions from another cadet, while he/she refers to his/her own solution
- Do not jointly implement a solution
- Helping your classmates learn and understand the homework concepts is encouraged, but extensive assistance should generally be provided by DFCS instructors. Only provide assistance up to your depth of understanding, beyond which assistance by more qualified individuals is more appropriate and will result in greater learning. If you have to look at your solution while giving help, you are most likely beyond your depth of understanding.
- Help your classmates maintain their integrity by never placing them in a compromising position. Do not give your solution to another cadet in any form (hard copy, soft copy, or verbal).
- **DFCS will recommend a grade of F for any cadet who egregiously violates this Help Policy or contributes to a violation by others. Allowing another cadet to see your assignment to help them will result in a zero on this assignment.**

Documentation Policy

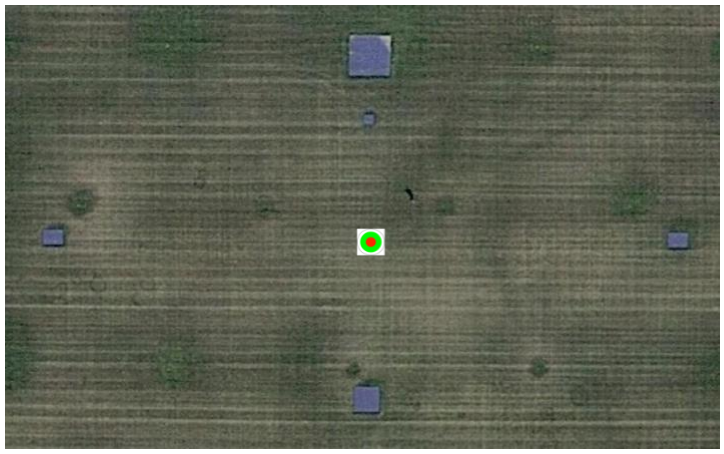
- You must document all help received from sources other than your instructor or instructor-provided course materials (including your textbook).
- The documentation statement must explicitly describe WHAT assistance was provided, WHERE on the assignment the assistance was provided, and WHO provided the assistance.
- If no help was received on this assignment, the documentation statement must state "NONE."
- If you checked answers with anyone, you must document with whom on which problems. You must document whether or not you made any changes, and if you did make changes you must document the problems you changed and the reasons why.
- Vague documentation statements must be corrected before the assignment will be graded and will result in a grade deduction equal to 5% (ceiling) of the total possible points.

This assignment will set you up for integrating with the **Tarot X6 V2.2** Drone system.

The task at hand will prepare us for the more complex task of object identification & tracking in the final project. In this project we will send our drone out on a mission to locate a ground target. Once that target is located, the drone will center itself over the target and safely land on top of it. As you can imagine, this might be a similar task performed by a delivery drone not unlike what Amazon is working to release as part of an automated delivery system.



Once your project is complete, your drone will be sent out on a mission, looking for a target on the ground similar to what is shone on the left. The target will be randomly placed somewhere in Stillman field as shown below from an overhead perspective. Once the drone acquires the target, the drone will break from the mission's search pattern and center over the target. As the drone begins to land, it will constantly adjust its coordinates so that it lands as close to the center of the target as possible. Once it has safely landed, the drone will disarm and your mission will be complete.



To do this we will be using Mission Planner, Python, DroneKit, OpenCV and SITL. We will rely on RBG images coming from the camera installed on our drone to locate a static target and land safely over it. We will be using SITL to test our code before attempting to try this with an actual drone. Only after careful testing is complete, will we try on a real-world target. **Attention to detail is critical for this assignment!** You will **follow all of the TODO comments** in **`drone_mission.py`** to complete your code.

The Process Explained

In this project we are taking a brief break from neural networks. Here, we will use OpenCV to detect a specific target of interest and then execute on that target by landing over it. OpenCV (*Open Source Computer Vision Library*) is a library of functions (mostly) aimed at real-time computer vision. It was originally developed by Intel, but is supported by a different company now called, Itseez, Inc. (now acquired by Intel). The library is cross-platform and free for use under the open-source Apache 2 License. OpenCV now features GPU acceleration for critical real-time operations.

All of the starter code contains a LOT of comments and useful links. Read those comments carefully and follow the various links to learn how you can perform the necessary steps to achieve your objective. The program flow is as follows:

(1) Connect to the drone’s autopilot
(2) Check for an existing flight plan/mission that was previously uploaded to the autopilot
    a. If there is no mission, disconnect from the autopilot and terminate program
(3) Arm the drone and place it into “GUIDED” mode
(4) Takeoff to 50 ft
(5) Change mode to “AUTO”
(6) Execute drone’s internal mission/flight plan.

(7) While the drone's mission executes, your program will be in a constant loop looking for images with attributes that match your target of interest.
   a. `search_for_target()` contains the main control loop for this task.
   b. If the program "thinks" it picked up a target, it will temporarily suspend the drone's internal mission and reposition the drone over the place where it originally picked of the target.
   c. If the program does not consistently pick up a positive target in subsequent video frames, disregard the target and resume drone's mission.

(8) Once the program has confirmed the target with reasonable certainty, it will switch to "GUIDED" mode, breaking the drone out of its current flight plan.

(9) The program will instruct the drone to descend at a half meter per second while it continually adjusts it's position relative to the confirmed target's center.

(10) At this point your program will continue to monitor the target's position relative to the drone.

(11) The drone will begin a very slow decent from 50 ft to around 10 ft.

(12) While descending, the program will make minor adjustments to the drone's position, attempting to keep the drone centered over the target, or at least within an acceptable radius from the target.
   a. If the confirmed target is lost for any reason and cannot be recovered, the program will switch the drone back over to its internal flight plan and continue searching for another target.

(13) Once the drone reaches 10 ft in altitude, the program will switch to "LAND" mode, allowing the drone to take over the landing procedure.

(14) Once the drone has landed, the program will disarm and disconnect from the autopilot and terminate the program.

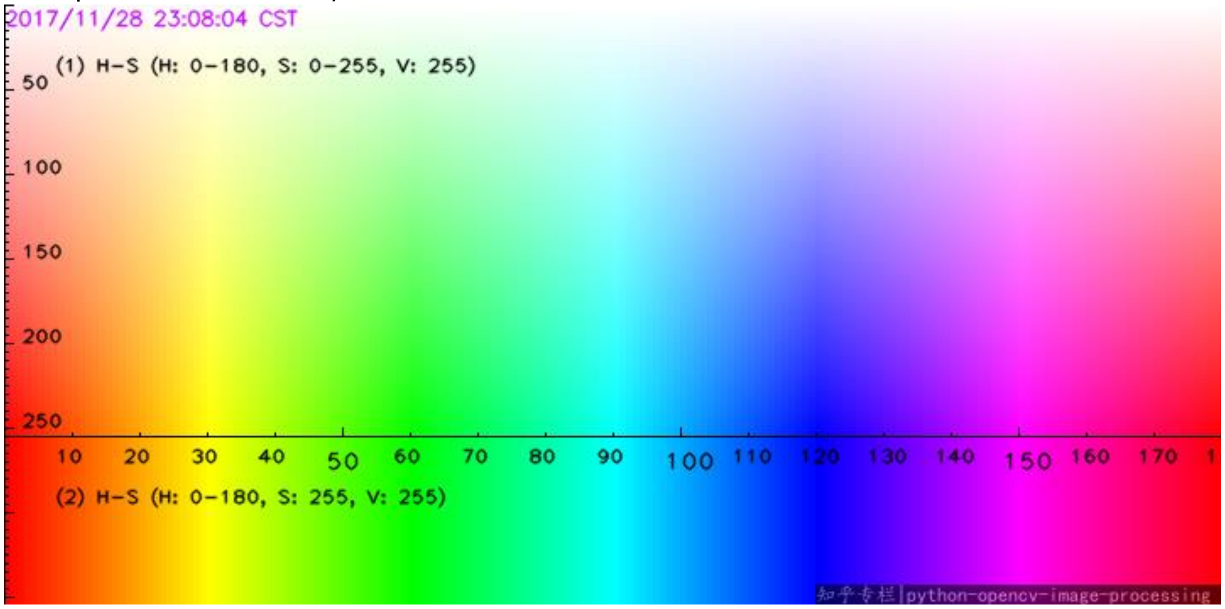
Finding the Target

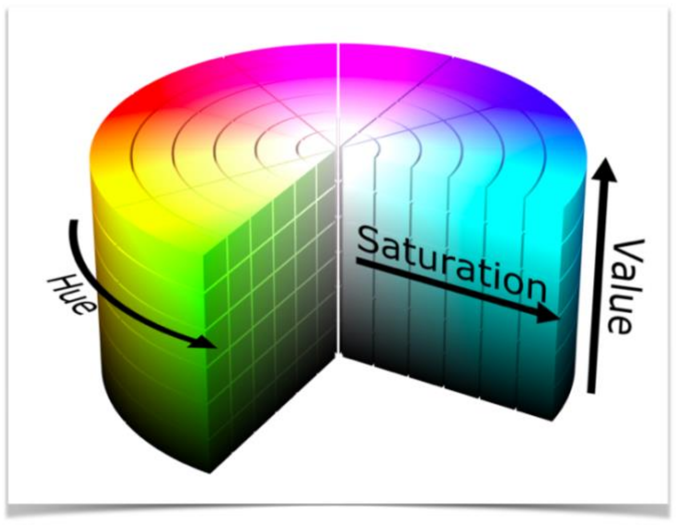
You will use OpenCV's advanced image processing algorithms to build our target sensor. Keep in mind that this target sensor is NOT object recognition in the strictest sense. This sensor will rely simply on a specific color range and size of the target surface area to differentiate between our target and everything else around it. You will learn how hue, saturation, and value (HSV) works and how you can use these values to isolate a specific range of colors in RGB images. In addition, you must process these incoming images further in order to inspect for a target of interest.

`search_for_target()` is complete and provides the main loop for this program; however, you must complete both the `determine_drone_actions()` and `check_for_initial_target()` functions so that the program can examine the properties in each incoming image for the proper color and size of a potential target and execute drone actions based on certain conditions (details provided in comments throughout the startup code you are provided with). The best way to go about this is to first consider nothing that has to do with controlling the drone and focus solely on the code that will detect and track your target. The simplest way to test this code while you're developing it is to have an actual camera on hand. Any RGB camera will do – even the one built into your laptop. You can use a few lines from your code in *Lab 05* to replace code in the `get_cur_frame()` function to capture incoming video frames. Another way will be to test with a RealSense camera. I will have some RealSense cameras available, but in limited supply. If you have a GoPro, that would work, too. Ideally, you will be streaming at 640x480 resolution (we want to stay on the lower side for this project). However, if your device cannot easily be configured for that resolution, you can simply down sample the images or change the FRAME_HORIZONTAL_CENTER and FRAME_VERTICAL_CENTER variables accordingly.

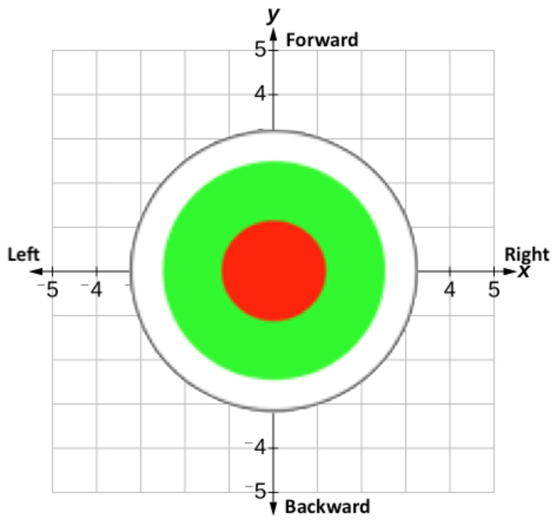
Details for completing this part of the project are contained in the actual comments in **`drone_mission.py`**.

One thing to note here: for testing purposes, **your target can by ANYTHING** – any shape, size, or color. You can set those parameters yourself in the code. See: tuples `COLOR_RANGE_MIN` and `COLOR_RANGE_MAX` as well as the value stored in `MIN_OBJ_RADIUS`. Modifying these variables will allow you to use objects on hand for testing – especially when you're testing your drone mission in SITL and FlightGear (more on this later). Once you understand how to use and threshold HSV, you may use the chart below to isolate a color for testing purposes (and ultimately to calibrate against the in-field/real target before launching the actual mission in Stillman field). In this chart Saturation is along the Y-axis and hue is along the X-axis (range between 0 and 180). Saturation is in a range of 0 to 255. "Value" is set to 255 for the entire chart; it is normally a range between 0 and 255 that determines how dark or how light a color appears along the z-axis; the lower the number, the darker the color (see 3D the cylindrical example below this chart).





Coming in For a Landing



Of course, ultimately, we're interested in safely landing over our target. To do this we're going to approximate a few things while also making a few assumptions. Keep in mind that this problem is largely a 2D problem. Our camera will be pointed directly towards the ground, and our (real) target will have no height. In addition, the Z axis (altitude) will decrease at (essentially) a constant rate of .5 m/s; therefore, we can (mostly) ignore the Z axis in our calculations. We're really only concerned with lining up on an X, Y Euclidean plane upon decent. The image on the right shows the overhead perspective of a target on the ground and the drone's movements relative to the target. This is based on the camera's orientation on the 2D plane, which will also be the drone's orientation. You will need to ***flip*** the Y coordinates such that to move forward will be a change in the negative direction. Note that all measurements we take from an image with respect to size or position **are in pixels**. Typically, we require some sort of way to convert pixels into reality, i.e. measurements in the real world that the drone can reliably execute on. Fortunately, we don't much care about measuring actual geospatial positioning in this exercise. Why? Because we're going to take a somewhat nondeterministic approach to centering and landing by making very minute adjustments to the drone's X, Y position every second during the landing procedure.

The first of our assumptions is that the drone's position is always in the center of every incoming image from the camera. This is a reasonable assumption, since the camera will be (more or less) centered under the drone. The next assumption we'll make: the drone will never fly at an altitude greater than 50 feet for this mission. *If the drone happens to be at a waypoint with a greater altitude, our code will forgo any attempt to detect a target*. Target detection will only occur between 50ft and 10 ft. This allows us to make a reasonable "guess" about the minimum size of our target to determine if we've detected a legitimate target of interest.

We will calculate the offset of the drone's position relative to the target's position in the frame ("frame" and "image" are one in the same) in order to determine the adjustments needed to keep the drone over the target. This process will continue until the drone reaches 10 ft above the ground. After that, we will discontinue any more adjustments and instruct the drone to land. Once the drone lands, we can assess how well the program performed by examining how closely to the target the drone rests. In any case, our primary concern is landing the drone in a safe location, unharmed, while causing no injury or damage to people and property in the process. Do not worry about actual scale and distance for now; you may use the sizes as already defined in the program for testing in the SITL environment.

Keep in mind that we are examining one image at a time as frames come in from the camera. Frame rates can be anything between 5 and 30 frames per second (fps). Each frame we examine is a *frozen moment in time* – a moment that has already passed. This means that the frame we are looking at will be in the past, and we're lagging a bit behind the current moment in time. This means that there will be some *inherent latency* in our process – a delay between when we confirm a target and when we execute an action. If we picked up a target in an earlier frame, chances are the drone is no longer in that same spot when it first picked it up; therefore, our code keeps a positional snapshot for each frame that we examine. That way we can reposition the drone to the original location where the target was first spotted to get a better look.

Follow the instructions (in the comments) in `drone_mission.py` to complete the code that will execute according to requirements explained in this document.

How to Proceed

Now that you have an idea of what needs to be done, how do you go about making it happen? First, create a new PyCharm project and configure the environment on Python 3.6 or Python 3.7. Install the following libraries:

- **Pyrealsense 2.2**: for Linux or Mac this is somewhat of an involved project; however, for Windows, it should be simpler. `pip install pyrealsense2`. Go here for more details: https://pypi.org/project/pyrealsense2/ Give yourself time for this one, in case things don't go smoothly.
- **OpenCV 4.5**: `pip install opencv-python`. https://pypi.org/project/opencv-python/
- **Imutil**: `pip install imutils`
- **NumPy 1.20**: `pip install numpy`. https://pypi.org/project/numpy/
- **Pymavlink 2.4**: `pip install pymavlink`. https://pypi.org/project/pymavlink/
- **Dronekit 2.9.2**: `pip install dronekit`. https://pypi.org/project/dronekit/
- **PyAutoGUI** 0.9.52: `pip install PyAutoGUI`. https://pypi.org/project/PyAutoGUI/
- **PyGetWindow** `0.0.9: pip install PyGetWindow`. https://pypi.org/project/PyGetWindow/

There are several python modules provided to get you started. The modules are:

- **drone_lib.py**: this module contains all the functions necessary to interact with autopilot installed on the drone. Everything from arming the drone, executing a flight plan, controlling the drone's position, landing and reading certain telemetry information is contained therein. While you don't necessarily need to modify this code, you are free to do so where it makes sense for your project.
- **fg_camera_sim.py**: this module provides a simple simulation of a camera feed that comes directly from the FlightGear simulator. This module is needed when you're testing your code in the simulated environment.
- **drone_mission.py**: this module will contain the bulk of all your code. This is the most important module, and it references the other two modules to assist in carrying out your mission for this PEX. There is a lot of starter code therein, but there is quite a bit of thinking and coding that you must provide to make this effort a success.

Follow instructions provided in the starter code to complete the coding tasks. Start with the code related to identifying a target. Test it separately and thoroughly before moving on to the code that controls the drone's actions. Next, complete the code related to the drone's actions. To test this code thoroughly you will need to setup a mission in SITL so that you have a reasonably realistic environment under which to test. **This environment will be reviewed and initially configured in-class**. We will all configure the SITL test together in class, step-by-step; more details to come. For the SITL environment we will choose something already in the scenery to use as a target. By the time you get to this stage, your environment should be ready to use.

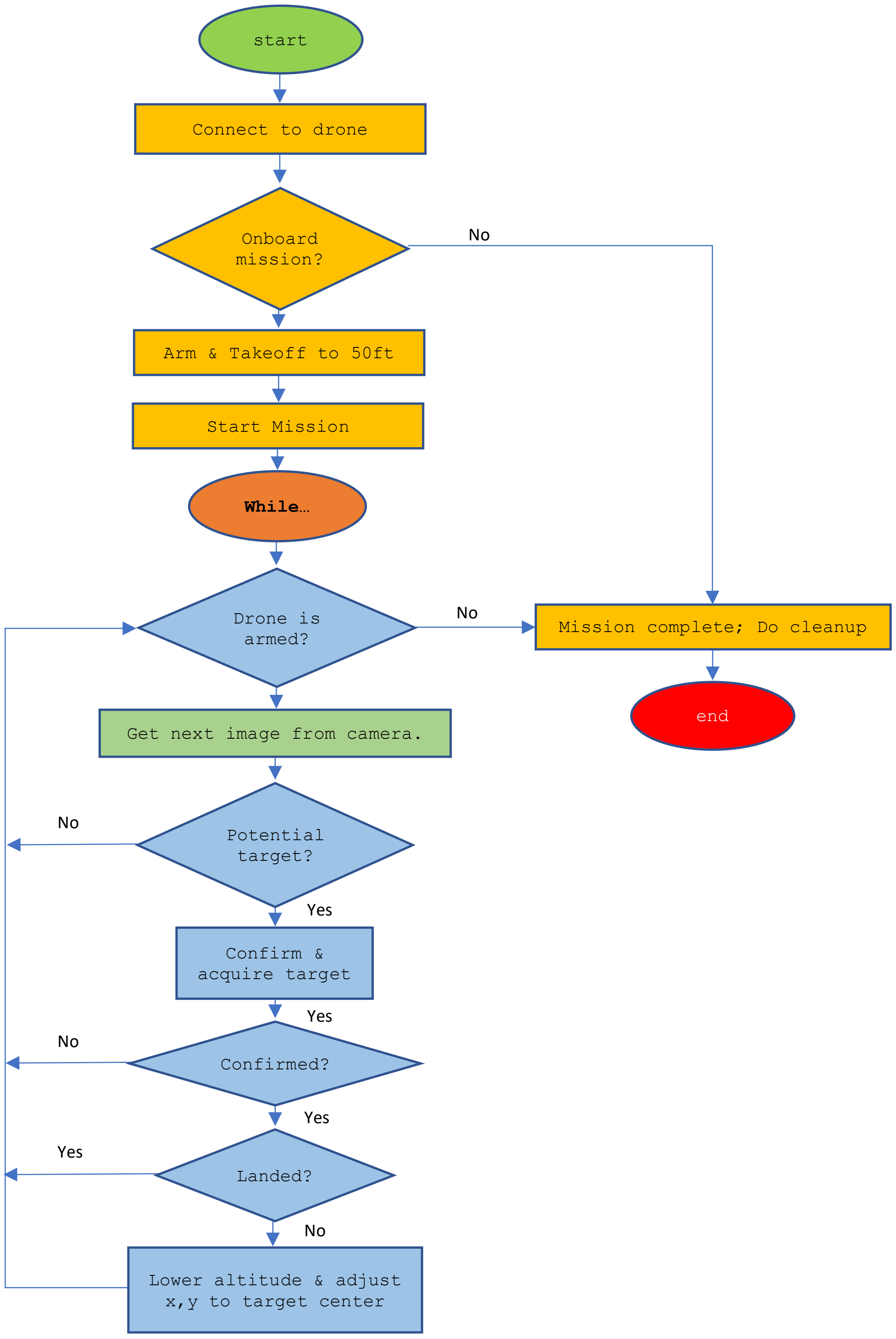
Project Deliverable

Your final code will be executed in-class on a PC configured with a SITL environment. All those in class will help to evaluate your code's performance. If it runs and results are reasonable, your code will be allowed to execute on a physical drone on another in-class day, after we've made further preparations. It is noted that your code may require minor tweaks before it is allowed to run on a drone.

If your code does not run or it does not perform within reasonable expectations, you will be docked points and your program will not be allowed to run on the drone. If this happens, the **best** score you can receive is 65%. If your code is accepted to run on a real drone, your minimum score is 80%. If your mission executes well during the real mission (i.e. I don't have to override it and it lands within the "safe zone" next to the target), you will receive 100% for a grade.

NOTE: You may all discuss among yourselves the difficulties you've each encountered and approaches you may have taken to get around them; however, **no one** may supply ***any code*** or direct answers to anyone else; your final solution ***must*** be something from ***your own work***.

A Simple Visualization of Program Flow


start
Connect to drone
Onboard mission?
No
Arm & Takeoff to 50ft
Start Mission
While…
Drone is armed?
No
Mission complete; Do cleanup
end
Get next image from camera.
No
Potential target?
Yes
Confirm & acquire target
Yes
No
Confirmed?
Yes
Yes
Landed?
No
Lower altitude & adjust x,y to target center