University of Canterbury

Summer Scholarship

Future Work

Author:
Michael McAdam
69780829
mjm369@*uclive.ac.nz*

March 8, 2016

1 Fix Sinking Issue

During testing, it was observed that the UAV would consistently lose height as it moved. It is suspected that this is caused by the off-board control consistently telling it to move 0 m/s in the Z direction meaning that the UAV cannot correct the height that it naturally loses as it tilts from side to side. This could be fixed by recording the Z position at the start of the offboard control and then using this as a set point for the remainder of the test.

Long term, a method for adjusting the height dependant on the position of the insect will need to be developed.

2 Test on Small Target

The algorithm has been proven to work using a large retro-reflective target, and indoor tests have been done which prove that the smaller targets can be tracked with correct lighting conditions. It simply remains to test the algorithm outside in a test flight to ensure that the software can track a smaller target.

3 Automatic Calibration

The success of the program relies on the calibration of the camera such that the target has a good contrast to the background. The camera is not set on auto settings because this naturally tries to balance the whole frame which is not the desired effect.

It could be of use to either have an automatic calibration algorithm, that could (for example) start completely dark and then step up until the target is identified (assuming the target is in the frame).

Failing that, integrating a manual calibration process into the insect tracking code would mean that the Flycapture software would not have to be used independently of the program.

NOTE: The API provides functions which can set these values, one of the first steps that should be completed is to set the frame rate to the highest possible value as this typically defaults to 10 FPS.

4 Develop and tune PID

Currently, there is no integral or derivative functionality implemented, and the proportional gain is set arbitrarily to 0.5. The first step in increasing the reactiveness of the tracking algorithm is to implement the integral and derivative control and then tune the response of the UAV.

5 Height

The height of the UAV above the beetle was set to be five metres - this was based off a trade off between algorithm reliability and interference with the beetle. In testing it was found that the downdraft and noise caused by the proximity of the UAV was higher than expected - meaning that the system may need to be re-designed to operate at a higher height relative to the beetle.

5.1 Investigate different lenses

The focal length of 12 mm was chosen based on this height, using the minimum number of pixels needed to describe the insect and it's projection onto the sensor. If the height was to be raised the focal length of the lens would also have to be raised. Another aspect that needs to be balanced with this is the re-activity of the gimbal - higher focal lengths are going to need more stability from the gimbal if the footage is to be stablised.

5.2 Torch strength

It might also be necessary to increase the strength of the infrared LED. While most of the testing performed has not been done in complete darkness, it has been found that the UAV has to be relatively close to the target before the retro-reflective tape becomes bright enough to be tracked. It will be important to investigate that the retro-reflectance/LED combination will provide enough contrast at a higher height to be effective for the computer vision software.

6 Computer Vision Algorithm

The computer vision algorithm implemented prioritises efficiency as the software has to work with large images at a high speed.

6.1 Feature Detector

It could be worthwhile investigating more classical feature detection techniques including: SIFT, SURF and ORB. The method currently used doesn't actually consider how pixels are clustered within a frame and instead relies on the natural gradients that a smoothly illuminated object will develop. An alternative to this is to look at local gradients within a frame - this could help to raise detection sensitivity at the possible cost of speed.

It would be recommended to use the in-built functions provided by OpenCV for each of these feature detection techniques - these functions have input parameters which can help to tune the detectors and output objects with useful information about the feature detected. These detectors would likely detect a much large set of features so some amount of effort will have to be focused on finding a reliable set of classifiers which could reliably pick out the beetle from a set of features.

Another negative with using these built-in detectors is less transparency - i.e, you are not aware of everything that the detector is doing - this means that tuning to the application may be more difficult.

6.2 Predictive Filter

To help increase reliability in the algorithm it could be useful to implement a predictive filter such as a Kalman or particle filter. This could reduce the effect of outliers on the accuracy of the program and could be used as a classifier in selecting the correct output from the feature detector. These methods would similarly add more strain to the CPU so it would be important to balance accuracy and performance.