Autonomous Target Following on Quadcopter Platforms
Eric Wang1,2, Arman Karimian2, Roberto Tron2
Monta Vista High School, 21840 McClellan Road, Cupertino, CA 950141; Department of Mechanical Engineering, Boston University, Boston, MA 022152

Introduction
Quadcopters are becoming more prevalent, with the market for drones growing at a rapid rate. Their success is due to its lightweight, easy to deploy nature and the low operating costs combined with increasing reliability.
The aim of this project is to capitalize on the advantages of a quadcopter by programming one to autonomously detect and follow a moving target. Currently, most unmanned aerial vehicles are still manually controlled by a remote operator. This not only requires the operator’s full attention but also training in flying the drone. While there are some commercial implementations of the target following system (marketed for photography purposes), these implementations often pair vision with GPS. Our research will be based purely on visual target tracking.
Applications for autonomous target following are numerous. For example, quadcopters can be rapidly deployed to follow a police chase down the streets. These drones can be released by the police to keep tabs on suspects before the appropriate units can be sent or they can be released by the media without the need to assemble a full camera crew. This research will enable operators to manage multiple cheap, easily deployable autonomous drones.

Platform
The project is designed for the Parrot Bebop 2 Quadcopter, an off-the-shelf drone equipped with Wi-Fi and a camera. The firmware on the Bebop provides basic stabilization capabilities allowing our software to provide simple velocity commands.

Figure 2: The AprilTag used.

The AprilTag is mounted on a custom built two wheel drive ground-based robot with a Raspberry Pi. It is remotely controlled by a keyboard.

Figure 3: The ground-based robot mounted with an AprilTag.

For target detection and tracking, this project utilizes the AprilTag 2 fiducial markers, which provide position tracking and orientation tracking.1

The AprilTag data is processed in a computer program that uses the AprilTag detected fiducial markers to calculate the target’s position and orientation relative to the drone.

Figure 4: Flowchart of data. Camera input is streamed to the computer for processing and velocity calculations before being sent back to the quadcopter.

Framework
The program itself is written largely in Python and uses the Robot Operating System (ROS) middleware for cross-platform deployment and communication protocols. ROS is a research and industry standard for robotic systems and can be thought of as a network of nodes talking to each other through ‘topics’.

Figure 5: Map of the ROS network for this project. Custom nodes are in black; existing ROS packages are in blue (see inline citations).

Figure 6: Quadcopter successfully aligned with an AprilTag.

Software
The software portion of this project can be broken down into multiple levels.

State Machine:
At the highest level lies the states of operation. Each state represents a distinct behavior. The State Machine regulates which state controls the quadcopter at any given point in time. To change the dominant state, an event needs to trigger a change.

Figure 7: The camera stream from the quadcopter overlaid with the axes of the AprilTag.

The Motor Controllers:
The software controls three (3) degrees of freedom on the quadcopter, linear x (forward and back), linear y (left and right), and angular z (yaw).

Each of the controllers implements a potential controller which provides a method for calculating the “potential” at a location, \( U(q) \), based upon how far the input value, \( q \), is from the goal. The “potential” at a location is the sum of both the “potential” generated by the goal along with the “potential” generated by any obstacles along the way.4

\[
U(q) = U_{\text{Goal}}(q) + U_{\text{Obstacle}}(q)
\]

The potential of the goal, \( U_{\text{Goal}} \), decreases at a slower rate as the input is closer to the input at the goal, eventually creating a bowl like structure.

The potential of an obstacle, \( U_{\text{Obstacle}} \), increases at a faster rate as the input gets closer to said obstacle, creating a wall like structure.

There are two different versions of the motor controllers implemented:

• First-degree (position based controllers)
  \( v_1(q) = \min\left(-U(q), \frac{-U_{\text{Goal}}(q)}{U_{\text{Obstacle}}(q)}\right) \)

• Second-degree (position and velocity based controllers)
  \( v_1(q) = v_1(q) - \varepsilon_{\text{PosVelo}} \)

Figure 8: The computer visualization of the quadcopter’s location and orientation relative to the AprilTag to the left.

Figure 9: The input values that the control uses to calculate the desired velocity for that axis.

Conclusions + Future
The second degree controller significantly outperforms the first degree controller. While the first degree controller is prone to overshooting, the second degree controller is capable of staying in close proximity to the goal, even when the tag moves erratically.

This project has been implemented to be easily expanded without modifying core programs and functionality. New controllers, image detection software and additional states are welcome. Possible extensions include:

• Multi-robot system where a “symphony” of quadcopters “conducted” by a single AprilTag
• Multi-target system with a single quadcopter keeping tabs on multiple AprilTags
• Natural image detection such as detecting wildlife, vehicles, or other drones.

References

Acknowledgements
Special thanks to Lisa Liu and Nicole Calero for helping me get started with the Parrot Bebop technology and providing support throughout my time here. And to Eric Wong for helping me with the hardware aspects of the ground robot.