Autonomous Landing of a Quadrotor on a Moving Platform in Outdoor Environments

Chris Choi*, Stanley Brown† and Steven L. Waslander‡

***Abstract*— The task of landing an aircraft on moving objects has been a topic of interest ever since humans took to the skies in the early twentieth century. Typically when attempting landing small, Micro UAVs such as quadrotors, the landing procedure is either done manually, or using GPS guidance to land on a stationary target. If landing on a moving target or in high wind conditions, a human operator is generally required. However this requires the human operator to have a clear line of site to both the target and the aircraft during all stages of the landing procedure. In recent years there have been several papers that propose a set of methods for landing on a slow moving platform in ideal conditions such as low wind and ideal GPS signals with little or no noise. Our contribution in this work is the demonstration of a set of algorithms that allow a quadrotor to perform a autonomous landing on a car travelling at over 20 meters per second while relying only on visual inertial data collected from and on-board gimbaled camera system. From this image data, along with on-board IMU readings, it is shown that both speed of the target and the quadrotor in the inertial frame can be directly estimated, and using this information a set of controllers and planners can be developed that allows for the quadrotor to perform a high speed landing manoeuvre in a variety of conditions. The robustness of our system is demonstrated by landing in without using GPS, poorly lit conditions, and with high winds.**

I. Introduction

Over the years there has been a growing interest in autonomous landing of quadrotors, as landing is often one of the most dangerous and risk prone parts of flight. There are now several commercial and research based implementations of autonomous landing with both the DJI Phantom and Pixhawk flight controllers being capable of automatic landing using a series control algorithms. However, there does not appear to be a reliable method of landing a quadrotor on either a moving platform or in relatively large wind disturbances.

In recent years there have been at least least 5 academic publications on the topic [**?**], [**?**], [**?**], [**?**], [**?**], [**?**] that approach the problem from a variety of angles. Overall none of these approach have completely solved the problem and most approaches only work on slow moving platforms with little or no wind disturbances.

One of the most challenging problems associated with autonomous landing is the difficulty of obtaining a reliable state estimate of the landing target relative to the quadrotor. This is due to the fact that the landing target can often go out of the camera's field of view causing the quadrotor to lose track of the target or measurements can become corrupted by vision based artifacts that include lighting, lens distortion, color, etc. In many cases GPS is not a viable option both the landing target and the quadrotor due a lack of accuracy unless an RTK GPS system is used. In many cases the use of an RTK system is not a readily available option due to cost and the lack of required equipment.

* M.A.Sc. Candidate, Mechanical and Mechatronics Engineering, University of Waterloo; c33choi@uwaterloo.ca

[2] M.A.Sc. Candidate, Mechanical and Mechatronics Engineering, University of Waterloo; s52brown@uwaterloo.ca

† Assistant Professor, Mechanical and Mechatronics Engineering, University of Waterloo; stevenw@uwaterloo.ca

II. Related Work

The autonomous landing problem can be thought of as a set of three separate control problems or stages. First the quadrotor must detect the relative position and velocity of the landing target using either GPS of visual methods. Next the quadrotor must determine a rendezvous location with the landing target and plan the required flight trajectory. Once the quadrotor has rendezvoused a final landing trajectory must be calculated between quadrotor and the landing pad.

For the perception portion of the problem there have been two main techniques use to identify the landing pad and estimate its state relative to the quadrotor. In the work of Kim, 2014 [**?**] a basic colour thresholding technique was used to identify the landing target and the MIT AprilTags library [**?**] was used to provide state estimations. In the work of Heriesse Et al., 2012 [**?**] an optical flow technique was applied to a series of images captured on-board the quadrotor to estimate the required states required to control the quadrotor and conduct autonomous landing. While both of these techniques have been shown to work in very specific cases both, require case specific parameters to be applied ahead of time, thereby limiting there general applicability.

In the case of Ling, 2014 [**?**], the brightness used in his thresholding technique must be set ahead of time and is sensitive to lighting changes. A similar issue is noted in the work of Heriesse Et al., 2012 [**?**]. Moreover in the work of Ling, there were implementation issues due to the slow state estimation rate of the AprilTag library and the AprilTag falling out of view of the camera during the final stages of decent. This meant that as the quadrotor approached the landing pad it tended to lose any measurements of its state and therefore had complete the final landing procedure by propagating the last measured state forward in time.

Examining the related literature on the controllers used in solving the the autonomous landing problem, the majority of literature either uses a set of PID or a Linear Quadratic

Controllers (LQC) to command and control the quadrotor at all stages of flight. A LQC was thoroughly explored in Friis Et al., 2009 [**?**] but the authors admit even though LQC performed slightly better than the PID controller the difference was minor and may be attributed to the additional time spent tuning the LQC. In [**?**], a PID controller was developed to land on a vertically oscillating platform no lateral movement. While interesting, the solution took over 1 minute to transition from hovering over the landing pad to landing. Additionally the experiments did not seem to account for pitch or roll of the platform.

In this project we focus developing a set of controllers that produce the final landing trajectory and using a AprilTags [**?**] to provide the estimated state of the quadrotor relative to the landing pad. We seek to develop both a more robust and rapid method of determining the state of the quadrotor and landing pad system and to develop a set of PID controllers that can be used to control the quadrotor based on the measured state information.

III. Contribution

- Robust Landing
- AprilTag estimation with Kalman Filter
- AprilTag windowing to speed up detection
- Light Invariant AprilTag detection
- Tracking Controller
- Trajectory Planning

IV. Problem Formulation

V. Camera Transforms

VI. Gimbal Transforms

VII. Landing Target Estimation

Kalman Filter constant acceleration

A. Adaptive Image Processing

As discussed previously, the low rate of the AprilTag library when processing images of 640 by 480 pixels results in an update rate of approximately 3 to 5 fps on the on-board computers used on the target quadrotor, which is too low to be used for controlling the quadrotor. Building upon who improved the AprilTag estimation rate by reducing the brightness of the image such that the majority of the image is black, we implemented an adaptive down sampling of the image size as the estimated distance between camera and AprilTag decreased. In this work two image down sampling sizes 320×280 (half resolution) and 160×140 (quarter resolution), were used. Whenever the distance between the camera and the AprilTag is less than 1.5 and 3 meters, the images are down sampled to the two resolutions respectively. If the camera is further than 3 meters from the AprilTag, the native resolution of 640×480 is used. This change only affected the image processing time when the quadrotor is within 3 meters of the quad and has the desired effect of maintaining a high AprilTag estimation rate.

B. Adaptive Image Windowing

If the AprilTag in the image space does not translate too suddenly, one can make the assumption that the previous detected AprilTag will approximately be in the same approximate image coordinates. A bounding box around the AprilTag can be formed and used to feed into the AprilTag detector.

C. AprilTag Inception

To address both the decreasing state update rate as a function of proximity and reduce the probability of losing sight of the AprilTag during landing a secondary AprilTag is embedded in the larger AprilTag, we term this technique AprilTag Inception (patent pending). This secondary AprilTag is assigned different family ID to avoid confusion during detection, and is placed at the center of the larger primary AprilTag. Whenever the secondary AprilTag is captured in the image, the adaptive windowing method is set to track only the secondary AprilTag rather than the larger one, which reduces the portion of the image that must be processed in with each image capture. An example of this procedure is highlighted in Figure.

VIII. Light Invariant AprilTag

IX. Quadrotor Control

X. Trajectory Planning

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{bmatrix} = \begin{bmatrix} x & \dot{x} & z & \dot{z} & \theta \end{bmatrix} \in \mathbb{R}^5 \tag{1}$$

$$\mathbf{x}_d = \begin{bmatrix} x_d & \dot{x_d} & z_d & \dot{z_d} & \theta_d \end{bmatrix} \tag{2}$$

$$\mathbf{u}(t) = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \frac{F_T}{m} \\ \omega \end{bmatrix} \in \mathbb{R}^2 \tag{3}$$

$$\begin{bmatrix} \ddot{x} \\ \ddot{z} \\ \theta \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ u_1 \\ u_2 \end{bmatrix} + \begin{bmatrix} 0 \\ -g \\ 0 \end{bmatrix} \tag{4}$$

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \\ \dot{x_5} \end{bmatrix} = f(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} x_2 \\ u_1 \sin(x_5) \\ x_4 \\ u_1 \cos(x_5) - g \\ u_2 \end{bmatrix} \tag{5}$$

$$\mathbf{u}(t) \in \mathbf{U} = \tag{6}$$

XI. System Architecture

XII. Experiment Results

XIII. Conclusions and Future Work