Nonlinear Control Framework for Gimbal and Multirotor in Target Tracking

Jae Hun Lee

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Randal W. Beard, Chair
Timothy W. McLain
D. J. Lee

Department of Electrical and Computer Engineering

Brigham Young University

ABSTRACT

Nonlinear Control Framework for Gimbal and Multirotor in Target Tracking

Jae Hun Lee
Department of Electrical and Computer Engineering, BYU
Master of Science

This thesis presents some existing gimbal and UAV control algorithms as well as novel algorithms developed as the extensions of the existing ones. The existing image-based visual servoing algorithms for both gimbal and UAV require the depth information to the object of interest. The depth information is not measurable when only monocular camera is used for tracking. This thesis is the result of contemplation to the question: how to remove the necessity of the depth to be measured. A novel gimbal algorithm using adaptive control is developed and presented with simulation and hardware results. Although the estimated depth using the algorithm cannot be used as reliable depth information, the target tracking objective is met. Also, a new UAV control algorithm for target following is developed and presented with simulation results. This algorithm does not require the depth to the target or the UAV altitude to be measured because it exploits the unit vectors to the target and to the optical axis.

Keywords: UAV, multirotor, gimbal, adaptive control, backstepping control, target tracking

ACKNOWLEDGMENTS

First and foremost, I would like to thank my family: my wife, Jihwa and my two sons, Woojoo and Suho for their support and encouragement throughout my graduate study.

I give a special thank to my advisor Dr. Randy Beard for being a great and patient advisor. He has guided me to be more capable researcher, engineer, thinker, and learner. Especially, I would like to acknowledge that he has introduced the non-linear approaches for the gimbal and UAV control. I also acknowledge the other members of my graduate committee members: Dr. Timothy McLain and Dr. D.J. Lee for teaching concepts that are important and closely related to my research.

I also would like to acknowledge Jerel Nielsen, Jeff Millard, Parker Lusk, Jacob White, Josh Sakamaki, and Mark Petersen for the engagement as a research group. The discussions I had with them has helped me greatly to make progress on my research.

Lastly, I thank all the members of the BYU MAGICC lab to create a wonderful learning opportunity and fun place to work and study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

Unmanned Aerial Systems (UAS) have become a popular platform for both research in academia and civil applications like filming, search and rescue, surveillance, and entertainment. UAS are advantageous for surveillance and target tracking because better visual awareness can be achieved with an airborne camera. Cameras are the most popular sensor on a UAS because of cost, weight, and because they are a rich source of information. For this reason, vision-based target tracking on UAS is an active area of research. This thesis presents some existing control methods as well as novel control schemes for gimbal and UAV in a Visual Servoing context [1].

Visual servoing is another name that is often used to refer to the vision-based robot control. It is a technique to manipulate the robot using information from visual sensors. There are two main types of visual servo control: Position-based visual servo control (PBVS) and Image-based visual servo control (IBVS) [2]. PBVS defines the error in the inertial frame. Thus, PBVS requires reconstructing the 3D inertial coordinates of the feature points from a 2D image. As one can imagine, the reconstructed 3D inertial coordinates can be inaccurate when camera calibration is not done correctly. Another disadvantage of the PBVS, perhaps the most undesirable for the matter that this thesis concerns, is that since the PBVS generates the robot commands in the inertial frame, the objects of interest may leave the camera field of view [3]. On the other hand, it is important to not lose the target from the camera field of view for tracking application which is the main advantage of the IBVS. In general, IBVS is used to manipulate the robot by calculating the desired servo velocity using image features directly (see Figure 1.1). Thus, since this thesis focuses on how to keep a target of interest in sight of a system operator, the IBVS framework fits well with vision-based gimbal control in Chapter 2 and UAV control in Chapter 3, 4 of this thesis.

Gimbal is widely used device for flying drones with a camera because they stabilize the camera from vibration and movement of the platform. Stabilizing the camera is important in order to take high-quality, smooth videos and photos. For this reason, most of the high-end commercial

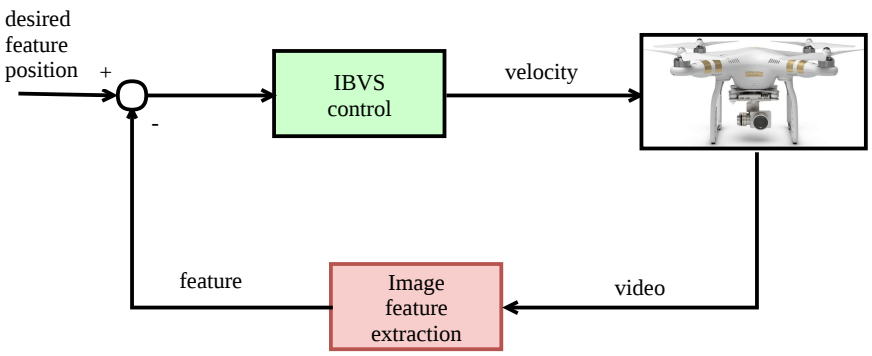


Figure 1.1: Image-based visual servoing (IBVS) structure

drones are equipped with a gimbal. For autonomous systems, a gimbal becomes more important because it adds maneuverability to the camera to provide more visual awareness and to keep the object of interest in the camera field of view. There are two cases when the object of interest can be centered on the optical axis. The first is when we know the relative location of the target to the camera. The second is when we know the target location in the image. For many practical applications, it is not feasible to assume that we know the target location in the global frame. Thus, image-based gimbal control is the more practical and realistic way to achieve the goal. In Chapter 2, we introduce several image-based gimbal control schemes.

Vision-based target tracking has been studied for decades. For example, fixed-wing applications are found in [4]–[8], and multirotor applications are in [9]–[12]. Tracking using a gimbaled camera is studied in [5], [6], [13], and tracking with a fixed camera is studied in [4], [8]–[12]. However, a common assumption that those studies make is that some information about the target, such as color [11], [12], shape [10] or pattern [14] is known or provided to the tracking algorithm. Thus, they require the user to specify what to track, or to provide the algorithm with a template image of the target in order for the tracker to be activated.

For example, the work in [4] demonstrates that a target can be kept in the camera field of view by constraining the roll angle of a small fixed-wing UAV in the presence of wind. However, the tracking method uses artificial color information and assumes that the target is static in the world frame. In [7], the tracking algorithm utilizes zero-mean normalized cross correlation to detect and locate the object of interest in the image, and therefore needs to be initialized by the user drawing a box around the target or with a template image of the target. Alternatively, the

system described in [9] can follow any user specified target in an outdoor environment while the UAS maintains fixed distance to the target using the OpenTLD tracker [15]. Occlusions are also well handled due to the machine learning algorithm of the OpenTLD. The advantage of the tracker in [15] is that it does not require any previous knowledge about the target of interest and is able to track a great variety of objects. However, the system in [9] is strictly designed to track only one target at a time and needs a different tracking framework to extend to the multiple target tracking scenario. Also, the user has to draw a bounding box to initialize the track while trying not to include much of the background. Alternatively, reference [10] shows impressive results in following a fast moving target using a receding-horizon control scheme that minimizes the velocity error during the initial transience. Still, the system in [10] is limited to detecting and localizing spherical shaped objects of known size.

The work presented in Chapter 3 overcomes many of these limitations and assumptions by using the recursive random sample consensus (R-RANSAC) algorithm that was first introduced in [16] and that was developed to track multiple dynamic targets in clutter. The algorithm has been applied to problems like RADAR tracking [16], [17] and UAV Sense and Avoid [18]. It has also been applied to vision based scenarios in which R-RANSAC is used to track multiple moving objects from a camera mounted on both static and mobile platforms [19], [20]. Chapter 3 extends our previous work and is the first attempt to close the feedback loop of a UAS around the R-RANSAC vision based tracking algorithm. The system presented here is unique in terms of tracking multiple objects that are in the camera field of view. Also, through the hardware demonstration, it is validated that the system can track realistic targets in unstructured environments with satisfactory performance. All operations in the hardware results are autonomous except for selecting the desired target ID.

The control law presented in Chapter 3 is relatively simple and has limitation. It can only compute the desired UAV position when the target is moving on a flat surface and when the correct UAV altitude is available. Thus, Chapter 4 introduces new control algorithm to overcome the limitation by incorporating adaptive control and backstepping control techniques. With mathematical derivation and simulation result, this algorithm is promising to be used in target tracking situation when the target is moving on unleveled surface. Testing the algorithm with hardware remains as future work.

Lastly, Chapter 5 provides a brief summary of the main contents and gives some suggestions and recommendations for the future work that can extend what has been done in this research.

The main contributions of this thesis are

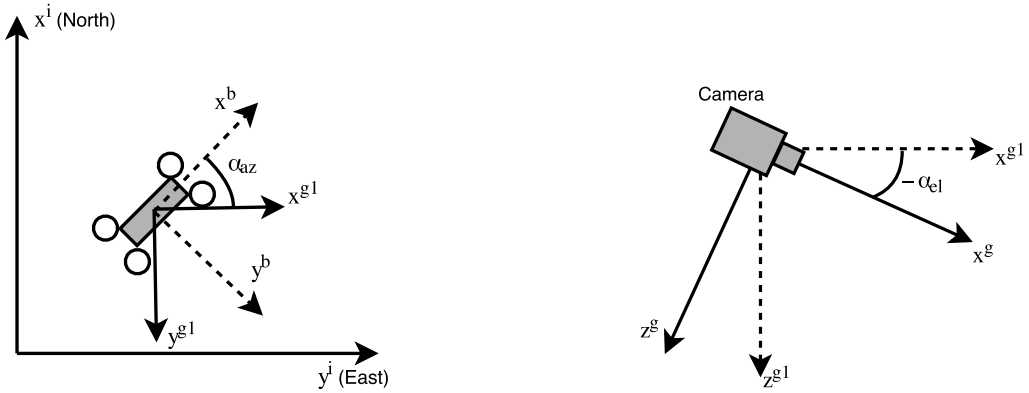
1. presenting three gimbal control algorithms including newly developed adaptive depth gimbal control.

2. integrating the system for multirotor autonomous target following using Recursive-RANSAC tracker and demonstrating the hardware results.

3. developing the unit vector visual servoing framework for UAV control in target tracking.

CHAPTER 2. GIMBAL CONTROL

In this chapter, Angle Commanding Gimbal Control (ACGC), Angular Velocity Commanding Gimbal Control (AVCGC), and Adaptive Depth Gimbal Control (ADGC) are derived and presented with some simulation and hardware result.



(a) Top view to show the relationship between the body frame and the gimbal-1 frame

(b) Side view to show the relationship between the gimbal-1 frame and the gimbal frame

Figure 2.1: Frames of interest: body frame, gimbal-1 frame, and gimbal frame

2.1 Angle Commanding Gimbal Control

2.1.1 Coordinate Frame Convention and Projective Camera Geometry

Before giving a detailed explanation of the gimbal control algorithms, we describe the coordinate frame convention and projective camera geometry. Note that the coordinate frame convention follows the convention from chapter 13 of [21]. We assume that the origins of the gimbal and camera frames are at the center of mass (COM) of the UAV. This is a reasonable assumption because the distance between the camera, gimbal and UAV COM is negligible compared to the

distance between the UAV COM and the target. Also, note that all coordinate frames and the direction of rotation follow the right-hand rule. There are four frames of interests: the body frame of the UAV denoted by $\mathscr{F}^b = (x^b, y^b, z^b)$, the gimbal-1 frame denoted by $\mathscr{F}^{g1} = (x^{g1}, y^{g1}, z^{g1})$, the gimbal frame denoted by $\mathscr{F}^g = (x^g, y^g, z^g)$, and the camera frame denoted by $\mathscr{F}^c = (x^c, y^c, z^c)$. The gimbal-1 frame can be obtained by rotating $\mathscr{F}^b$ about z^b axis by α_{az} which we call the gimbal azimuth angle. The gimbal frame can be obtained by rotating $\mathscr{F}^{g1}$ about y^{g1} by α_{el} which we call the gimbal elevation angle. Note that pointing the gimbal toward the ground is a negative elevation angle (See Fig. 2.1). The camera frame is defined so that the z^c axis points along the optical axis. Thus, the rotation from body frame to gimbal-1 frame can be expressed as

$$R_b^{g1}(\alpha_{az}) = \begin{bmatrix} \cos\alpha_{az} & \sin\alpha_{az} & 0 \\ -\sin\alpha_{az} & \cos\alpha_{az} & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.1}$$

Similarly, the rotation from gimbal-1 frame to gimbal frame can be expressed as

$$R_{g1}^{g}(\alpha_{el}) = \begin{bmatrix} \cos\alpha_{el} & 0 & -\sin\alpha_{el} \\ 0 & 1 & 0 \\ \sin\alpha_{el} & 0 & \cos\alpha_{el} \end{bmatrix}. \tag{2.2}$$

Combining equations 2.1 and 2.2, we get the rotation from the body frame $\mathscr{F}^b$ to gimbal frame $\mathscr{F}^g$ as

$$R_b^g(\alpha_{az}, \alpha_{el}) = R_{g1}^g(\alpha_{az}) R_b^{g1}(\alpha_{el}) = \begin{bmatrix} \cos\alpha_{el}\cos\alpha_{az} & \cos\alpha_{el}\sin\alpha_{az} & -\sin\alpha_{el} \\ -\sin\alpha_{az} & \cos\alpha_{az} & 0 \\ \sin\alpha_{el}\cos\alpha_{az} & \sin\alpha_{el}\sin\alpha_{az} & \cos\alpha_{el} \end{bmatrix}. \tag{2.3}$$

The camera frame, often called the optical frame, follows the common computer vision convention: x-axis is directed to the right, y-axis is directed down and the z-axis is directed out from the camera optical sensor (See Fig. 2.2). Thus, the rotation from the gimbal frame to the

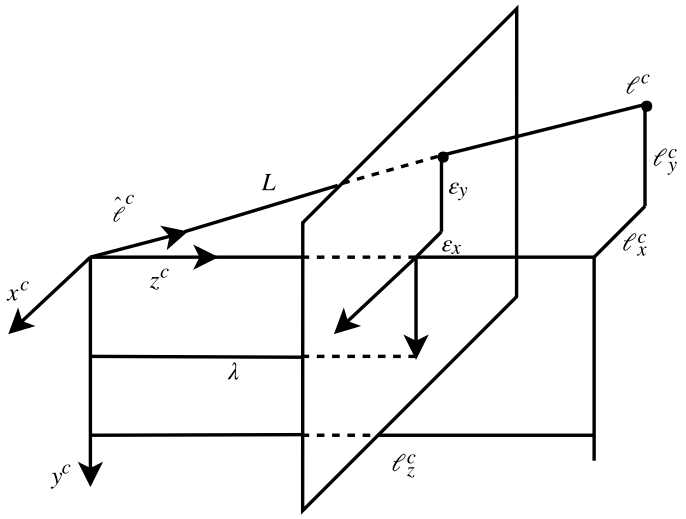


Figure 2.2: Camera frame and visual aid for projective camera model

camera frame is fixed and is given by

$$R_g^c = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}. \tag{2.4}$$

The basic idea of the projective camera model is that the 3D world is projected onto a 2D image plane that is orthogonal to z^c axis with distance being the focal length λ from the origin of the optical axis (See Fig. 2.2). Because of this projection, we lose the distance information to the target which introduces many problems in computer vision and estimation. The focal length of the camera is the distance between the optical sensor and the lens, and it can be obtained through camera calibration. Since common camera calibration gives the focal length in the units of pixels, the camera's field of view angle (M) can also be obtained as

$$M = 2\tan^{-1}\left(\frac{\varepsilon_{max}}{\lambda}\right) \tag{2.5}$$

where ε_{max} is the maximum pixel value from the center of the image assuming the image is square.

2.1.2 Derivation

Let the unit vector from the origin of the camera frame to the target (i.e. the line of sight vector or LOS vector) in the camera frame be defined as

$$\hat{\ell}^c = \frac{1}{L}\begin{pmatrix}\varepsilon_x\\ \varepsilon_y\\ \lambda\end{pmatrix} = \frac{1}{\sqrt{\varepsilon_x^2+\varepsilon_y^2+\lambda^2}}\begin{pmatrix}\varepsilon_x\\ \varepsilon_y\\ \lambda\end{pmatrix} = \begin{pmatrix}\hat{\ell}_x^c\\ \hat{\ell}_y^c\\ \hat{\ell}_z^c\end{pmatrix}. \tag{2.6}$$

The angle commanding gimbal control algorithm computes the gimbal azimuth and elevation angles that would make the optical axis align with the unit LOS vector. By transforming the unit LOS vector (2.6) into the body frame, we get the desired optical axis as

$$\hat{\ell}^b = R_g^b R_c^g \hat{\ell}^c. \tag{2.7}$$

By equating the equation (2.7) to the unit optical axis vector transformed into the body frame, we can compute the desired azimuth and elevation commands as

$$\hat{\ell}^b = \begin{pmatrix}\hat{\ell}_x^b\\ \hat{\ell}_y^b\\ \hat{\ell}_z^b\end{pmatrix} = R_g^b(\alpha_{az}^d, \alpha_{el}^d)R_c^g \begin{pmatrix}0\\0\\1\end{pmatrix} \tag{2.8}$$

$$= \begin{bmatrix}\cos\alpha_{el}^d\cos\alpha_{az}^d & -\sin\alpha_{az}^d & \sin\alpha_{el}^d\cos\alpha_{az}^d\\ \cos\alpha_{el}^d\sin\alpha_{az}^d & \cos\alpha_{az}^d & \sin\alpha_{el}^d\sin\alpha_{az}^d\\ -\sin\alpha_{el}^d & 0 & \cos\alpha_{el}^d\end{bmatrix}\begin{bmatrix}0&0&1\\1&0&0\\0&1&0\end{bmatrix}\begin{pmatrix}0\\0\\1\end{pmatrix} \tag{2.9}$$

$$= \begin{pmatrix}\cos\alpha_{el}^d\cos\alpha_{az}^d\\ \cos\alpha_{el}^d\sin\alpha_{az}^d\\ -\sin\alpha_{el}^d\end{pmatrix}. \tag{2.10}$$

Solving for α_{az}^d and α_{el}^d as

$$\alpha_{az}^d = \tan^{-1}\left(\frac{\hat{\ell}_y^b}{\hat{\ell}_x^b}\right) \tag{2.11}$$

$$\alpha_{el}^d = -\sin^{-1}\left(\hat{\ell}_z^b\right), \tag{2.12}$$

results in the gimbal commands that place the object of interest at the center of the image.

2.1.3 Hardware Result

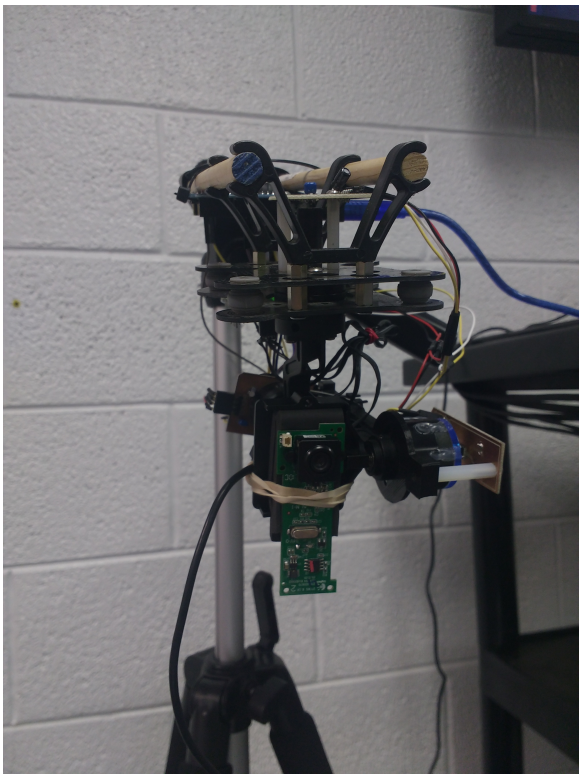
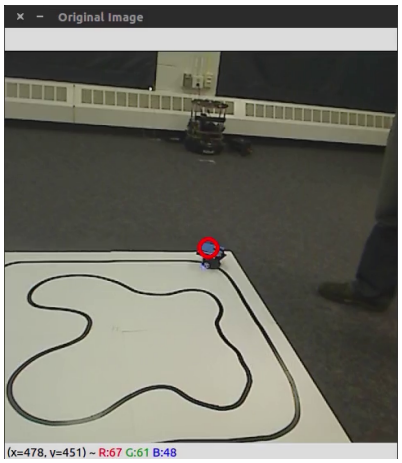
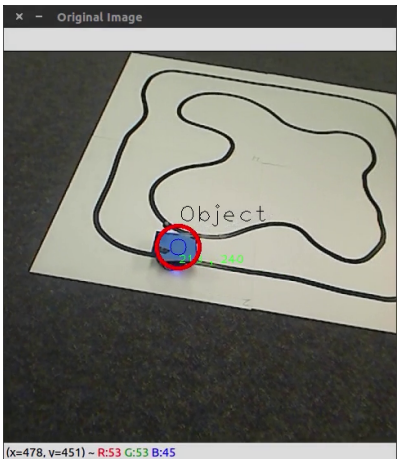


Figure 2.3: Prototype hardware to test the gimbal control algorithm

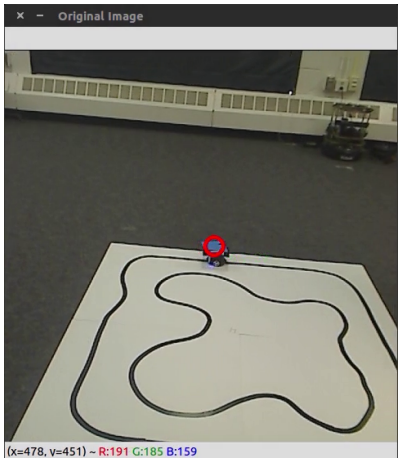
The gimbal control algorithm has been implemented and tested with simple hardware (see Figure 2.3). The major hardware component includes the BaseCam SimpleBGC 32-bit gimbal controller, a webcam, DYS 3-axis GoPro gimbal, and the AS5048B magnetic rotary encoder attached to each rotating axis to measure the gimbal angles. Note that in this hardware experiment, the roll axis of the gimbal is always commanded to be zero to model a pan-tilt gimbal. The focal length of the camera is known through the camera calibration process and the target pixel location is given by the color detection algorithm implemented in OpenCV [22]. The objective of the gimbal control is to keep the target at the center of the camera field of view (see Figure 2.4. A video of the results can be found at https://www.youtube.com/watch?v=OZ0Mg8AoAzk).



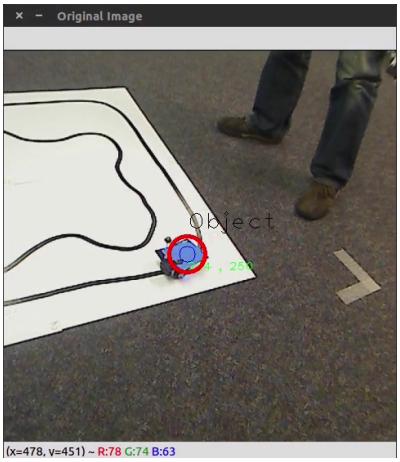
(a) when $t = 0s$



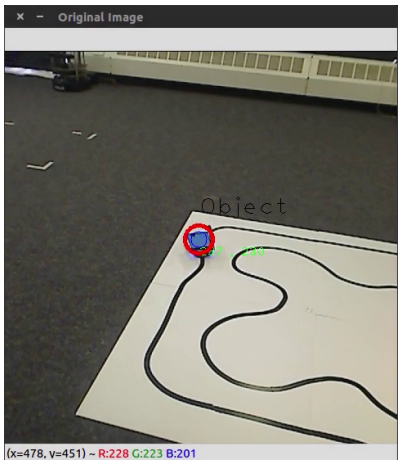
(b) when $t = 10s$



(c) when $t = 20s$



(d) when $t = 30s$



(e) when $t = 40s$

Figure 2.4: Hardware demonstration. The gimbal control algorithm is keeping the target object at the center of the camera field of view.

2.2 Angular Velocity Commanding Gimbal Control

In the previous section, a relatively simple gimbal control algorithm is presented. One disadvantage of that algorithm is that it does not take the camera velocity into consideration. When the gimbal is mounted on a stationary platform, the angle commanding controller can perform without any performance degradation. However, if the gimbal is mounted on a moving UAV, then taking the camera velocity into account in the controller becomes important. The work in [13] addresses this issue and derives an algorithm that overcomes the issue. This angular velocity commanding gimbal control algorithm is presented briefly in this section because some of the concepts in this section introduces important background for the next section.

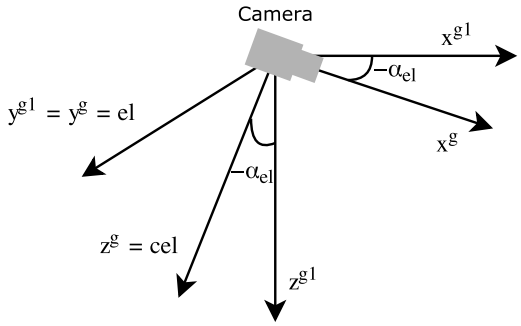


Figure 2.5: Elevation and cross-elevation axis

2.2.1 Image Jacobian

A full derivation of getting Image Jacobian matrix (often called interaction matrix) can be found in [23]. Here, we only show the final form of the Image Jacobian in the process of developing the gimbal control algorithm. Let the camera translational and rotational velocities both expressed in the camera frame be defined as

$$\mathbf{v}^c = [v_x^c, v_y^c, v_z^c]^\top \tag{2.13}$$

$$\omega^c = [\omega_x^c, \omega_y^c, \omega_z^c]^\top \tag{2.14}$$

$$\xi = [\mathbf{v}^{c\top}, \omega^{c\top}]^\top. \tag{2.15}$$

Let α_{el} be the elevation angle, and let α_{az} be the azimuth angle. Also, following the convention in Figure 2.2,

$$R_c^g = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \tag{2.16}$$

Also, let the coordinates of an image feature and its pixel velocity be defined as

$$\mathbf{s} = [u, w]^\top \tag{2.17}$$

and

$$\dot{\mathbf{s}} = [\dot{u}, \dot{w}]^\top. \tag{2.18}$$

The Image Jacobian matrix L at $\mathbf{s}$ is the relationship between the camera velocity in (2.15) and the image feature velocity in (2.18) and can be expressed as

$$\dot{\mathbf{s}} = L(\mathbf{s}, z, \lambda)\xi \tag{2.19}$$

or more explicitly

$$\begin{bmatrix} \dot{u} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} -\frac{\lambda}{z} & 0 & \frac{u}{z} & \frac{uw}{\lambda} & -\frac{\lambda^2+u^2}{\lambda} & w \\ 0 & -\frac{\lambda}{z} & \frac{w}{z} & \frac{\lambda^2+w^2}{\lambda} & -\frac{uw}{\lambda} & -u \end{bmatrix} \begin{bmatrix} v_x^c \\ v_y^c \\ v_z^c \\ \omega_x^c \\ \omega_y^c \\ \omega_z^c \end{bmatrix} \tag{2.20}$$

where z is the depth along the optical axis to the target, and λ is the focal length. Note that λ is a fixed parameter once the camera is calibrated. The equation (2.20) can be divided into two parts:

one for the camera translational velocity and the other for rotational velocity as

$$\dot{\mathbf{s}} = \begin{bmatrix} -\frac{\lambda}{z} & 0 & \frac{u}{z} \\ 0 & -\frac{\lambda}{z} & \frac{w}{z} \end{bmatrix} \mathbf{v}^c + \begin{bmatrix} \frac{uw}{\lambda} & -\frac{\lambda^2+u^2}{\lambda} & w \\ \frac{\lambda^2+w^2}{\lambda} & -\frac{uw}{\lambda} & -u \end{bmatrix} \boldsymbol{\omega}^c \tag{2.21}$$

$$= L_v(u,w,z)\mathbf{v}^c + L_\omega(u,w)\boldsymbol{\omega}^c \tag{2.22}$$

It is worth noting that only the translational term depends on the target depth z.

2.2.2 Feedback Linearization-based Visual Pointing and Tracking

The control objective is to drive an image feature to the center of the image. The horizontal error can be eliminated by moving two axis gimbal motors attached on azimuth and elevation axis. Moving only the azimuth axis cannot fully compensate for the horizontal error because it only indirectly affects z^g axis which is often referred as cross-elevation axis *cel* (see Figure 2.5). Thus, some vertical error is introduced and it needs to be compensated by moving the elevation at next time step. However, the key idea of the controller is that by exploiting ω_x^g term which is the angular velocity along the optical axis, we can find the azimuth and elevation commands for the same time step that smoothly compensates for the horizontal error without introducing the vertical error. Let the error in the image plane be

$$e(t) = s(t) - s^{ref} \tag{2.23}$$

where $s^{ref} = [0,0]^\top$ to push the image feature to the image center. We would like to drive $e(t)$ to zeros by controlling the elevation angular velocity $\omega_{EL} = \omega_y^g$ and the azimuth angular velocity $\omega_{az} = \frac{\omega_z^g}{\cos\alpha_{el}}$. Re-arranging equation (2.22) gives

$$L_\omega(u,w)\boldsymbol{\omega}^c = \dot{s} - L_v(u,w,z)\mathbf{v}^c. \tag{2.24}$$

Note that we can only command the camera angular rate where the camera linear velocity can be measured by other sensors. The null space of the matrix L_ω can be parameterized as

$$\mathcal{N}\{L_\omega\} = \{k \begin{bmatrix} u & w & \lambda \end{bmatrix}^\top\}. \tag{2.25}$$

Therefore adding (2.25) to (2.24) gives

$$L_\omega(u,w)\omega^c = \dot{s} - L_v(u,w,z)\mathbf{v}^c + L_\omega(u,w)k\begin{bmatrix} u \\ w \\ \lambda \end{bmatrix}. \tag{2.26}$$

The null space parametrization means that a rotation about the axis connecting the image feature and the origin of the optical axis does not change the coordinates of the image feature. The left pseudoinverse of L_ω is given by

$$L_\omega^\sharp = \begin{bmatrix} 0 & \frac{\lambda}{\lambda^2+u^2+w^2} \\ -\frac{\lambda}{\lambda^2+u^2+w^2} & 0 \\ \frac{w}{\lambda^2+u^2+w^2} & -\frac{u}{\lambda^2+u^2+w^2} \end{bmatrix}. \tag{2.27}$$

Multiplying both side of (2.26) by (2.27) yields

$$\omega^c = \frac{1}{z(\lambda^2+u^2+w^2)}\begin{bmatrix} \lambda^2 v_y - \lambda v_z w + \lambda \dot{w} z + ku \\ -\lambda^2 v_x + \lambda v_z u - \lambda \dot{u} z + kw \\ -\lambda u v_y + \lambda w v_x - u\dot{w} z + \dot{u} w z + k\lambda \end{bmatrix}. \tag{2.28}$$

If we require $\dot{s}^{ref}(t) = -\alpha I s(t)$ where α is a positive value, the actual value of $s(t)$ is expected to be exponentially stable which means it converges to zero. Thus,

$$\dot{\mathbf{s}}^{ref} = \begin{bmatrix} \dot{u}^{ref} \\ \dot{w}^{ref} \end{bmatrix} = -\alpha \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} u \\ w \end{bmatrix} = \begin{bmatrix} -\alpha u \\ -\alpha w \end{bmatrix}. \tag{2.29}$$

By plugging equation (2.29) into (2.28), we can compute the desired camera angular rate that guarantees the asymptotic stability for the image error as

$$\omega^{cref} = \frac{1}{z(\lambda^2+u^2+w^2)}\begin{bmatrix} \lambda^2 v_y - \lambda v_z w - \lambda \alpha w z + ku \\ -\lambda^2 v_x + \lambda v_z u + \lambda \alpha u z + kw \\ -\lambda u v_y + \lambda w v_x + k\lambda \end{bmatrix}. \tag{2.30}$$

This angular rate reference command expressed in the camera frame must be transformed into the gimbal frame in order to command each gimbal axis as

$$\omega^{gref} = R_c^g \omega^{cref} = \begin{bmatrix} \omega_x^{gref} \\ \omega_y^{gref} \\ \omega_z^{gref} \end{bmatrix} \tag{2.31}$$

$$= \frac{1}{z(\lambda^2+u^2+w^2)} \begin{bmatrix} -\lambda u v_y + \lambda w v_x + k\lambda \\ \lambda^2 v_y - \lambda v_z w - \lambda \alpha w z + k u \\ -\lambda^2 v_x + \lambda v_z u + \lambda \alpha u z + k w \end{bmatrix}. \tag{2.32}$$

Unfortunately ω^{cref} contains reference commands for three axes, but there are only two controllable axes, namely azimuth and elevation. This problem can be solved by using the free parameter k to come up with two proper motor commands. The key strategy is to select k so that $\omega_x^g = 0$. Accordingly,

$$k = u v_y - w v_x + \frac{z(\lambda^2+u^2+w^2)}{\lambda} \omega_x^g \tag{2.33}$$

where ω_x^g must be measured from a sensor such as a MEMS gyro attached to the camera. Substituting for k in ω_y^{gref} and ω_z^{gref} from the equation (2.32) with (2.33) gives

$$\omega_{el}^{ref} = \omega_y^{gref} = \frac{\lambda^2 v_y - \lambda v_z w - \lambda \alpha w z + u^2 v_y - u w v_x}{z(\lambda^2+u^2+w^2)} + \frac{\omega_x^g u}{\lambda} \tag{2.34}$$

$$\omega_{cel}^{ref} = \omega_z^{gref} = \frac{-\lambda^2 v_x + \lambda v_z u + \lambda \alpha u z + u w v_y - w^2 v_x}{z(\lambda^2+u^2+w^2)} + \frac{\omega_x^g w}{\lambda}. \tag{2.35}$$

Using the relationship

$$\omega_{az} = \frac{1}{\cos \alpha_{el}} \omega_{cel}, \tag{2.36}$$

we can compute the desired angular rate on the azimuth axis as

$$\omega_{az}^{ref} = \frac{1}{\cos \alpha_{el}} \left(\frac{-\lambda^2 v_x + \lambda v_z u + \lambda \alpha u z + u w v_y - w^2 v_x}{z(\lambda^2+u^2+w^2)} + \frac{\omega_x^g w}{\lambda} \right). \tag{2.37}$$

The system block diagram of this controller is shown in Figure 2.6. The controller presents the ideal control scheme for a two axis inertially stabilized gimbal system, but it also presents some

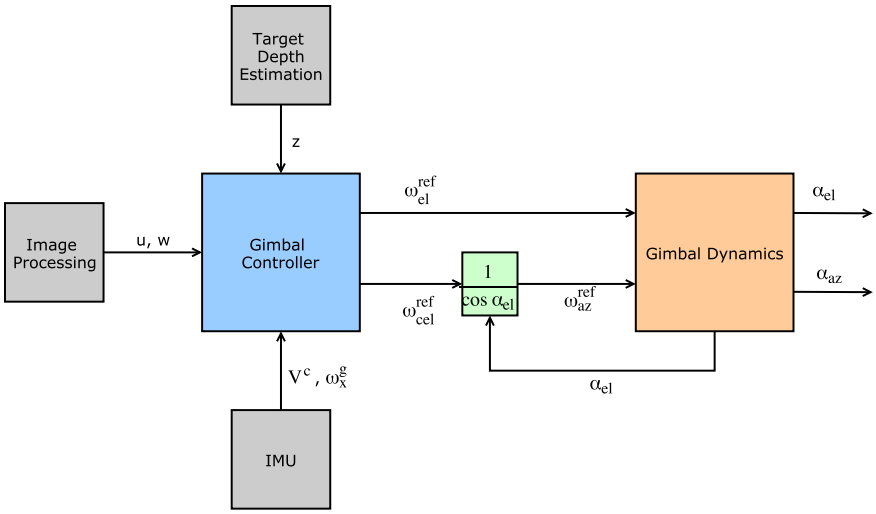


Figure 2.6: Angular velocity commanding gimbal controller block diagram

technical challenges such as estimating the depth to the target or estimating the camera translational velocity. The camera translational velocity can be either estimated by using the UAV velocity if available or be treated as a disturbance. Estimating the depth can be handled using a laser range finder if available or geolocation technique. Another option is to use the novel adaptive depth gimbal control algorithm that we introduce in the next section.

2.3 Adaptive Depth Gimbal Control

The gimbal control algorithm in the previous section provides a method to keep a target of interest in the camera's field of view with gimbal azimuth and elevation control. However, the target depth z is required to complete the Jacobian (2.20). This section examines a strategy to remove this requirement by estimating the depth online. This controller is derived using Model Reference Adaptive Control (MRAC).

2.3.1 Introduction

Model Reference Adaptive Control (MRAC) is a useful control scheme to stabilize dynamical systems even when there are unknown parameters in the system. A general MRAC closed-loop block diagram can be found in Figure 2.7 [24]. The design of an MRAC system requires a reference model that specifies how the actual system should behave. An adaptive law then adapts the

parameters using the error between the reference model and the system output. Finally, the controller computes the command for the plant using the adjusted parameters, the external command, and the system output.

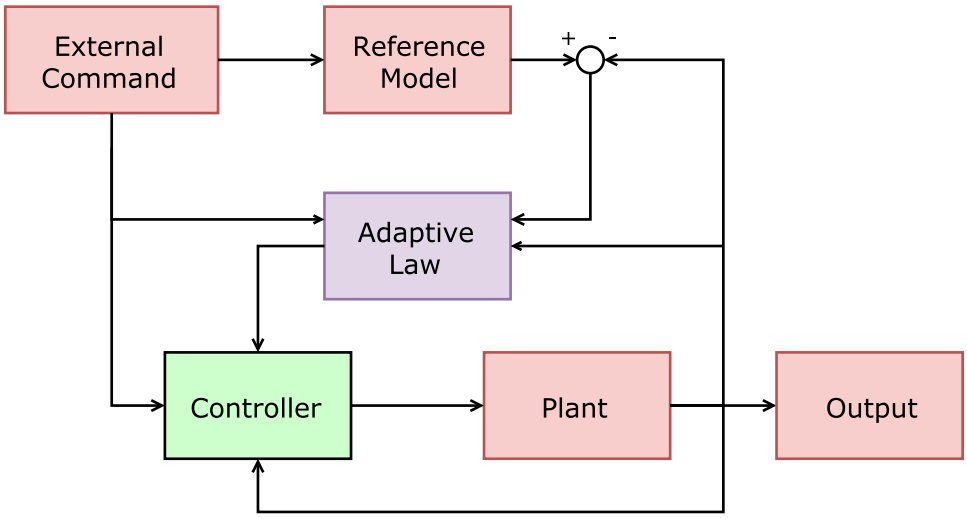


Figure 2.7: An MRAC closed-loop block diagram

2.3.2 Derivation

Consider again the Image Jacobian matrix

$$\begin{bmatrix} \dot{u} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} -\frac{\lambda}{z} & 0 & \frac{u}{z} & \frac{uw}{\lambda} & -\frac{\lambda^2+u^2}{\lambda} & w \\ 0 & -\frac{\lambda}{z} & \frac{w}{z} & \frac{\lambda^2+w^2}{\lambda} & -\frac{uw}{\lambda} & -u \end{bmatrix} \begin{bmatrix} v_x^c \\ v_y^c \\ v_z^c \\ \omega_x^c \\ \omega_y^c \\ \omega_z^c \end{bmatrix} \tag{2.38}$$

or for simplicity

$$\dot{\mathbf{s}} = L_v(u,w,z)\mathbf{v}^c + L_\omega(u,w)\omega^c. \tag{2.39}$$

Since we desire that the target is pushed to the center of the image plane, the reference model can be set as

$$\dot{\mathbf{s}}^{ref} = A\mathbf{s}^{ref} \tag{2.40}$$

where $\mathbf{s} = [u, w]^\top$ and A is Hurwitz (i.e. every eigenvalue of A has strictly negative real part). It is clear that the reference model in (2.40) is globally asymptotically stable. From the image jacobian matrix, ω^c is the control input, because it is common situation to track the target with gimbal movement rather than platform movement. Thus, equation (2.38) can be manipulated into a more convenient form for MRAC design as

$$\dot{\mathbf{s}} = \frac{1}{z}\begin{bmatrix} -\lambda v_x^c + u v_z^c \\ -\lambda v_y^c + w v_z^c \end{bmatrix} + \begin{bmatrix} \frac{uw}{\lambda} & -\frac{\lambda^2+u^2}{\lambda} & w \\ \frac{\lambda^2+w^2}{\lambda} & -\frac{uw}{\lambda} & -u \end{bmatrix} \begin{bmatrix} \omega_x^c \\ \omega_y^c \\ \omega_z^c \end{bmatrix} \tag{2.41}$$

$$= \beta\varphi + L_\omega U \tag{2.42}$$

where U is the control input and $\beta = \frac{1}{z}$. If β were known, then the ideal control input would be

$$U = L_\omega^\sharp(-\beta\varphi + K\mathbf{s}) \tag{2.43}$$

where $L_\omega^\sharp$ is the right pseudoinverse of L_ω and K is a negative definite tuning matrix. Then, the subsequent dynamics would become

$$\dot{\mathbf{s}} = K\mathbf{s} \tag{2.44}$$

which is globally asymptotically stable. However, since β is an unknown quantity and needs to be adapted, a more realistic control input is

$$U = L_\omega^\sharp(-\hat{\beta}\varphi + K\mathbf{s}) \tag{2.45}$$

where $\hat{\beta}$ is the estimate of β. Plugging this control input into the equation (2.42) gives

$$\dot{\mathbf{s}} = \beta\varphi - \hat{\beta}\varphi + K\mathbf{s} \tag{2.46}$$

$$= \tilde{\beta}\varphi + K\mathbf{s} \tag{2.47}$$

where $\tilde{\beta} = \beta - \hat{\beta}$. Let the error be defined as

$$\mathbf{e} = \mathbf{s} - \mathbf{s}^{ref}. \tag{2.48}$$

Taking derivative and equating K to A gives

$$\dot{\mathbf{e}} = \dot{\mathbf{s}} - \dot{\mathbf{s}}^{ref} \tag{2.49}$$

$$= \tilde{\beta}\varphi + K\mathbf{s} - A\mathbf{s}^{ref} \tag{2.50}$$

$$= \tilde{\beta}\varphi + A\mathbf{e}. \tag{2.51}$$

Using the variables of interest, a Lyapunov function candidate can be constructed as

$$V = \frac{1}{2}\mathbf{e}^\top\mathbf{e} + \frac{1}{2\gamma_\beta}\tilde{\beta}^2. \tag{2.52}$$

Taking the derivative yields

$$\dot{V} = \mathbf{e}^\top\dot{\mathbf{e}} + \frac{1}{\gamma_\beta}\tilde{\beta}\dot{\tilde{\beta}} \tag{2.53}$$

$$= \mathbf{e}^\top(\tilde{\beta}\varphi + A\mathbf{e}) + \frac{1}{\gamma_\beta}\tilde{\beta}\dot{\tilde{\beta}}. \tag{2.54}$$

Assuming that the inverse depth β is constant or slowly moving ($\dot{\beta} = 0$, $\dot{\tilde{\beta}} = -\dot{\hat{\beta}}$), the above equation becomes

$$\dot{V} = \mathbf{e}^\top A\mathbf{e} + \tilde{\beta}(\mathbf{e}^\top\varphi - \frac{\dot{\hat{\beta}}}{\gamma_\beta}). \tag{2.55}$$

Since $\dot{\hat{\beta}}$ is a design parameter, its value can be selected as

$$\dot{\hat{\beta}} = \gamma_\beta\mathbf{e}^\top\varphi. \tag{2.56}$$

Then, the equation (2.54) becomes

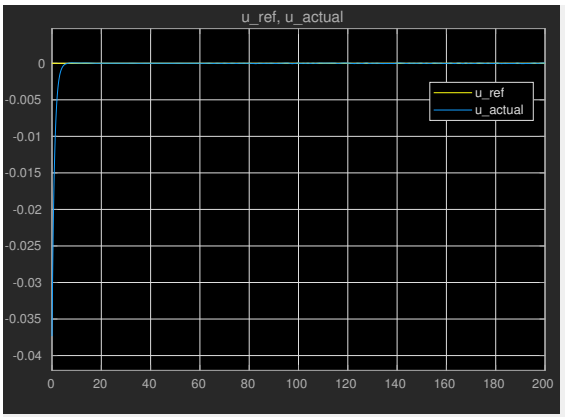
$$\dot{V} = \mathbf{e}^\top A\mathbf{e}. \tag{2.57}$$

Asymptotic stability of the system follows from the fact that A is Hurwitz.

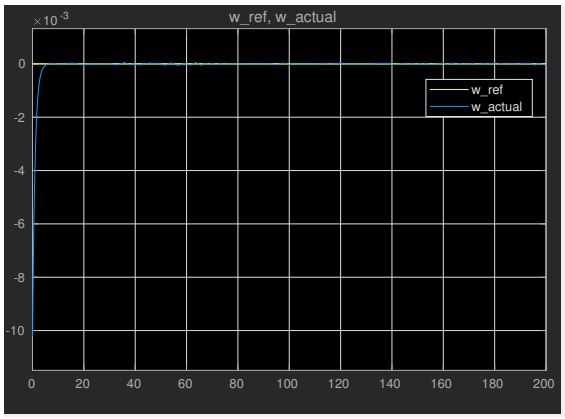
2.3.3 Simulation

First, the reference model $\mathbf{s}^{ref}$, in equation (2.40) is selected as $[0,0]$ over time. The objective of the MRAC controller is to drive the actual image dynamics with unknown quantity z as close as it can to the reference model. In simulation, a square image with 800 pixel width and height is used. The camera field of view is set to 60 degrees and one pixel is set to one millimeter. Thus, we can obtain the focal length and image width in metric units. The initial image coordinates are randomly selected and these coordinates are expected to converge to the origin as the reference model. In reality, the depth z is a varying quantity because of the relative movement between the camera and the target. However, in this simulation z is initialized to an arbitrary value between 30 and 300, and is changed randomly with increment ± 2 every second. In order to simulate the translational velocity of the UAV, v_x^c and v_y^c are initialized to zeros and changed with increment ± 2 every other second. v_z^c is kept zero assuming that the camera is not directly flying into nor away from the target. Figure 2.8 shows that the controller keeps the target in the center of the image which is the primary objective. Figure 2.9 presents the angular velocity commands computed by the adaptive depth gimbal controller and the depth estimation. Based on Figure 2.9, the estimation of the uncertain parameter seems to converge to its true value. However, note that this is rather a special case. In normal cases, only uniform boundness of the uncertain parameters is guaranteed [24].

In addition, this gimbal control is simulated on a multirotor equipped with a pan-tilt gimbal. The multirotor is moving and suppose that its translational velocity is available. Also, the target is moving on the ground. The adaptive depth gimbal controller takes the pixel location of the target from a camera at a $30Hz$ frame rate, and its goal is to push the target to the center of the image. The simulated multirotor and camera view are presented in Figure 2.10. Additional information regarding the simulation can be found in Figure 2.11. These plots are rather typical for MRAC: meeting the control objective without a guarantee that the uncertain parameter will converge to its true value.

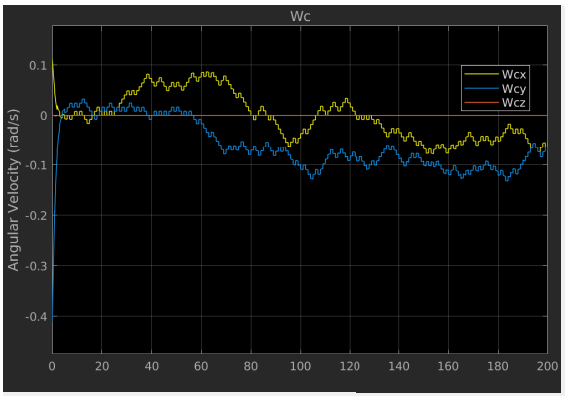


(a) Reference image coordinate u_{ref} and the system output u

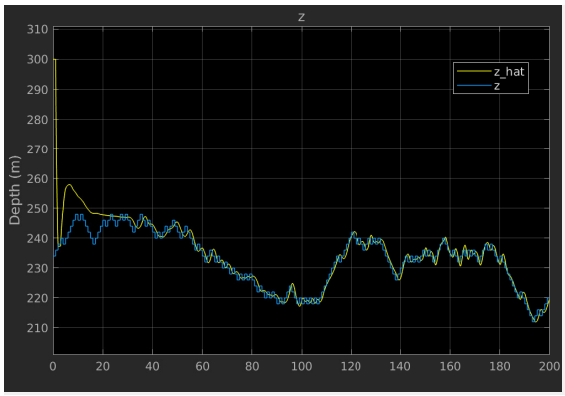


(b) Reference image coordinate w_{ref} and the system output w

Figure 2.8: The simulation result for the adaptive depth gimbal control. The system output u and w are converging to the reference model output u_{ref} and w_{ref}.



(a) Angular velocity commands of the controller

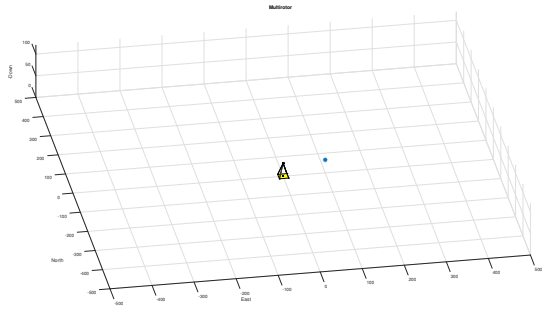


(b) Online depth estimation, $\hat{z}$

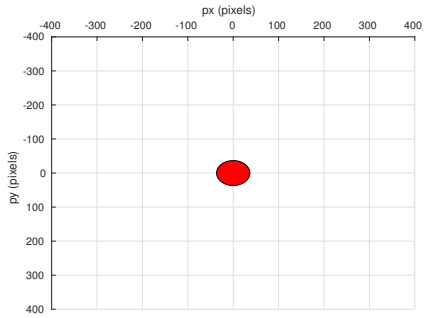
Figure 2.9: Angular velocity commands of the adaptive depth gimbal controller. Note that only two commands are used, since it is a pan-tilt gimbal. The depth z estimate using MRAC.

2.3.4 Hardware

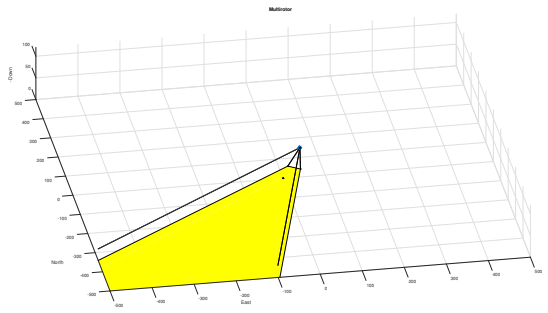
The adaptive depth gimbal controller is tested on the custom pan-tilt gimbal shown in Figure 2.12. The gimbal consists of a uEye UI-3250-LE-C-HQ camera, BaseCam SimpleBGC 32-bit gimbal controller, CUI AMT203 ABS SPI encoder, Quanum 4008 precision brushless gimbal motor, and 3D printed housing (See Figure 2.13). The system block diagram can be found in Figure 2.14. The Arduino acts as a communication bridge by receiving the current azimuth and elevation angle from the encoder and passes them to the onboard computer. Also, it passes the angular



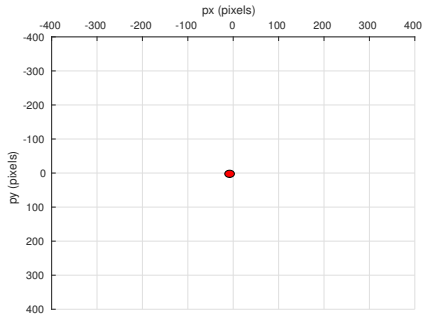
(a) Multirotor at $t = 0s$. The blue star indicates the desired multirotor position for the position PID controller for moving camera platform.



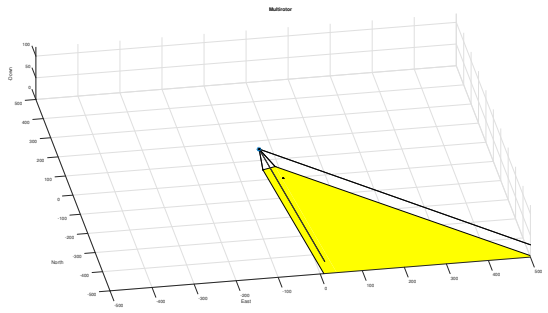
(b) Camera view at $t = 0s$



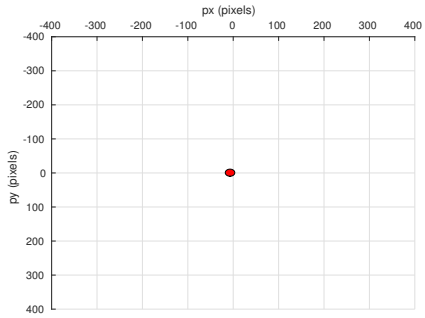
(c) Multirotor at $t = 60s$



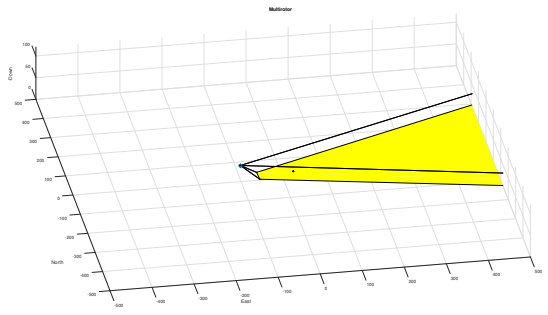
(d) Camera view at $t = 60s$



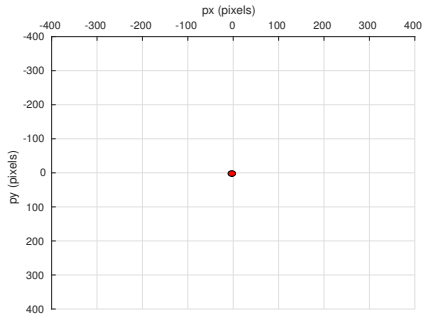
(e) Multirotor at $t = 120s$



(f) Camera view at $t = 120s$

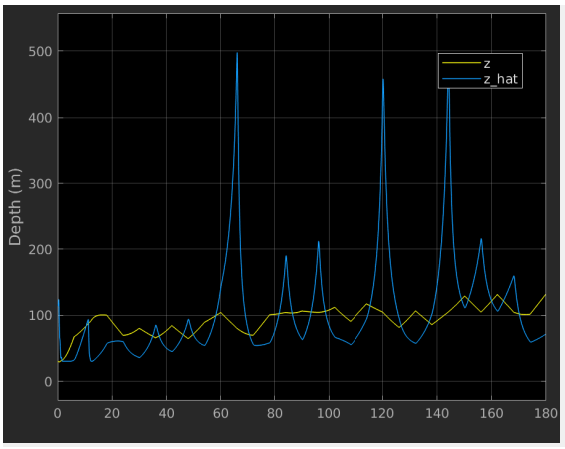


(g) Multirotor at $t = 180s$

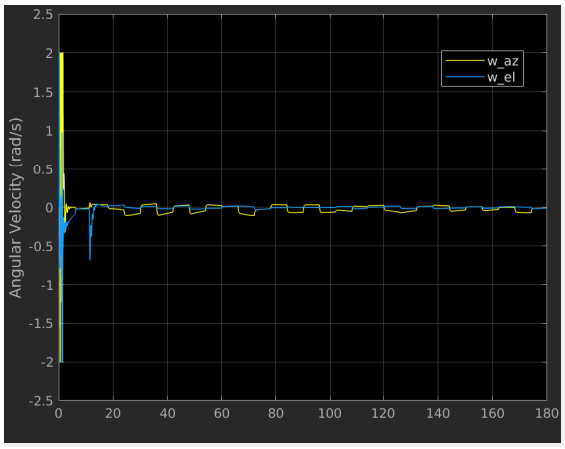


(h) Camera view at $t = 180s$

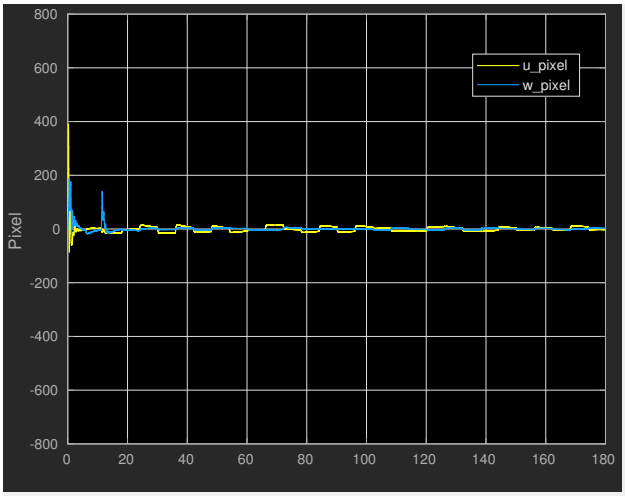
Figure 2.10: Multirotor simulation with camera view. The gimbal pointing objective is well achieved.



(a) Depth z and its estimate $\hat{z}$. Uncertain parameter converging to true value is not guaranteed.



(b) Angular velocity gimbal azimuth and elevation commands



(c) Pixel value u and w. They are maintained around the center of the image.

Figure 2.11: Uncertain parameter estimation, gimbal angular velocity commands from the controller, and where target lies in the image.

velocity commands from the onboard computer to the gimbal controller. A tracking algorithm running on the onboard computer receives an image from the camera and outputs the pixel location of the target. The onboard computer also runs the controller that computes the gimbal angular velocity and passes it to the Arduino. A series of snapshots that shows the gimbal pointing result with the custom hardware and the adaptive depth gimbal controller are shown in Figure 2.15. Also, a video showing the same result can be seen at https://www.youtube.com/watch?v=VMfvQkhD9-o. In this particular result, the camera is static which means that v^c is set to zero.

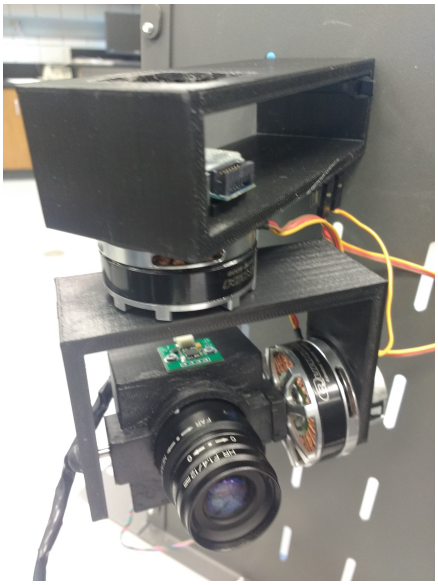


Figure 2.12: A custom pan-tilt camera gimbal

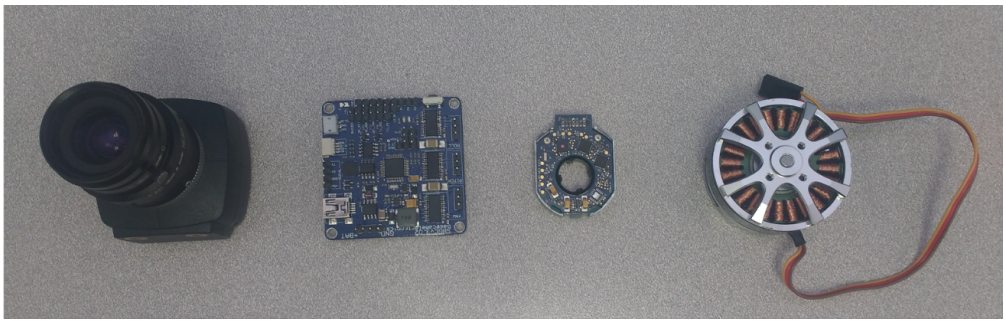


Figure 2.13: A custom pan-tilt camera gimbal

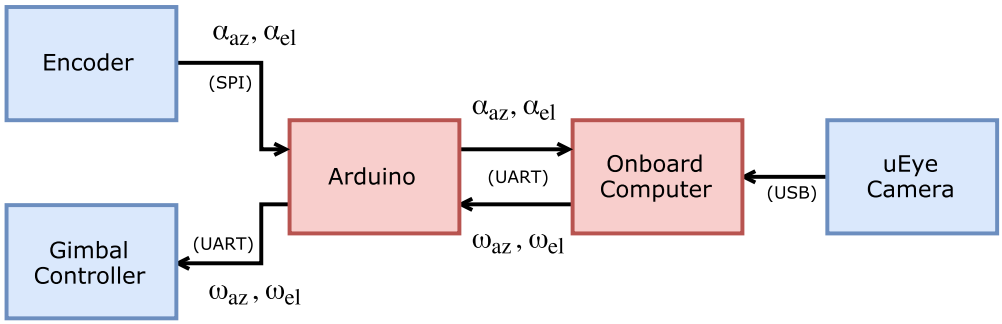
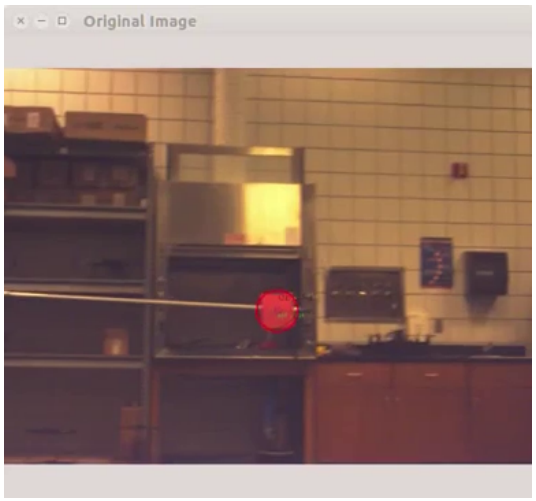
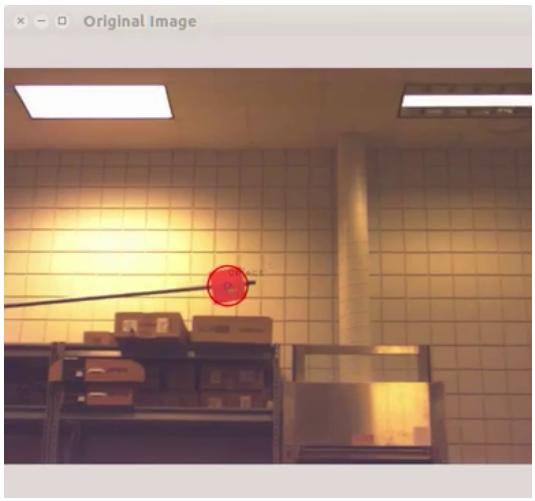


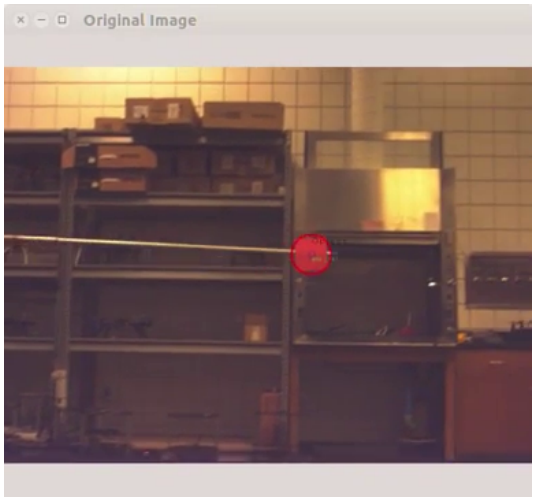
Figure 2.14: Custom gimbal block diagram



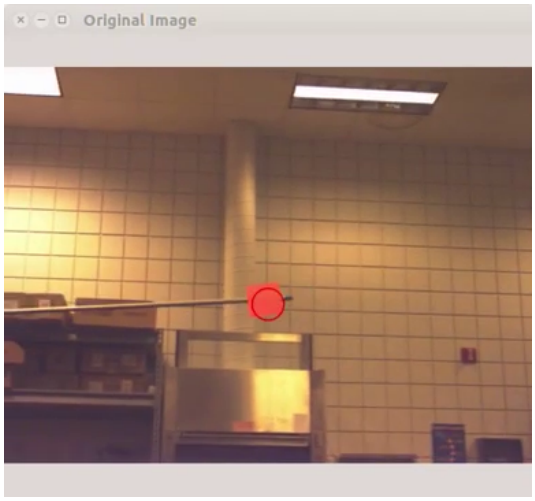
(a) At $t = 0s$



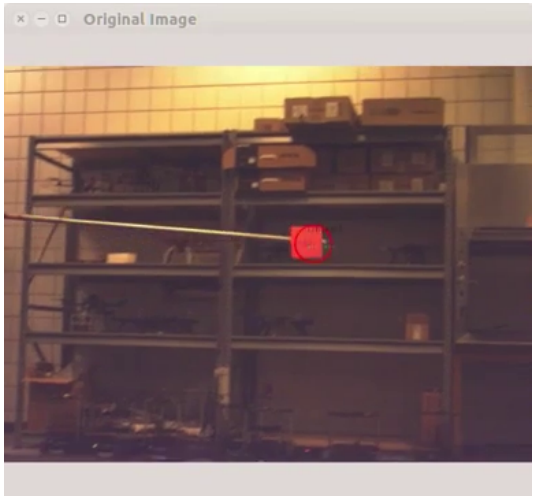
(b) At $t = 10s$



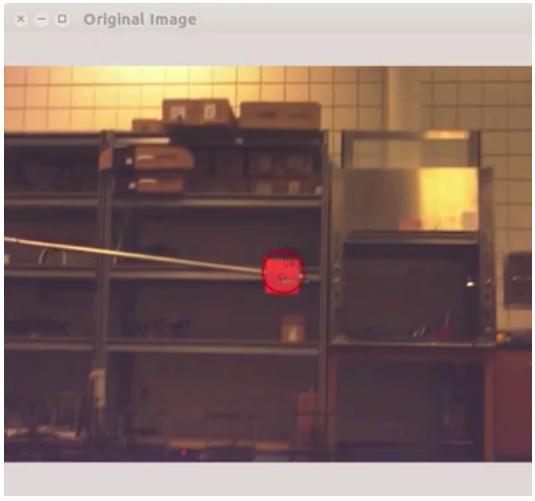
(c) At $t = 20s$



(d) At $t = 30s$



(e) At $t = 40s$



(f) At $t = 50s$

Figure 2.15: Adaptive depth gimbal control result on the custom-built hardware.

2.4 Conclusion

There are a few controller options to keep a target in the camera field of view using a gimbal. Depending on whether the camera is moving or stationary, either angle commanding gimbal control (ACGC) or angular velocity commanding gimbal control (AVCGC) can be used. The advantage of the ACGC is that it is easy to implement and that it does not require the depth information. The AVCGC is expected to perform better than ACGC for a moving camera since the AVCGC take the moving platform into account. However, the AVCGC requires knowledge of the depth z. When there is no way to reliably measure or estimate the depth z, the adaptive depth gimbal control (ADGC) can be used. A summary is shown in Table 2.1.

ACGC	AVCGC	ADGC
Easy to implement Designed for static camera	Designed for moving camera Requires the knowledge of the depth z	Designed for moving camera Works without knowing z

Table 2.1: Summary of gimbal control schemes

CHAPTER 3. AUTONOMOUS TARGET FOLLOWING SYSTEM

This chapter presents a vision-based target tracking and following system using a monocular camera on an Unmanned Aerial System (UAS). The Recursive-RANSAC (R-RANSAC) tracker tracks multiple moving objects in the camera field of view and the proposed controller is capable of following a particular target selected by a user while keeping the target in the center of the image. The main contribution of this work is that multiple objects can be tracked without imposing restrictions such as color, shape, etc. Also, the hardware test shows that the system is able to follow a target autonomously in a real-world outdoor environment.

3.1 System Overview

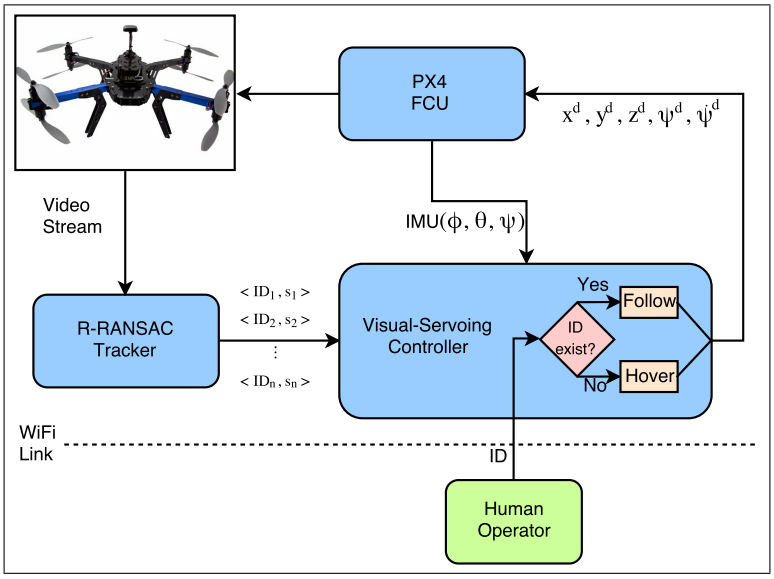


Figure 3.1: System Architecture. The R-RANSAC tracker produces a set of target ID numbers and corresponding pixel locations. The visual-servoing controller outputs the desired position, heading, and yaw rate based on the pixel location of the requested target.

The R-RANSAC tracker and the visual-servoing controller are major subsystems as shown in Figure 3.1 and they communicate using the Robot Operating System (ROS) framework. The Pixhawk flight controller is used with the PX4 firmware for the autopilot. The R-RANSAC tracker is responsible for tracking moving objects in the image sequence and outputs a vector of normalized image coordinates with unique ID assigned to each track. Let the normalized pixel coordinates be defined as

$$s = [\varepsilon_x, \varepsilon_y] \tag{3.1}$$

where ε_x, ε_y are normalized pixel coordinates. Combined with IDs, a vector of track information is defined as

$$T = [< ID_1, s_1 >, < ID_2, s_2 >, \ldots, < ID_n, s_n >]. \tag{3.2}$$

The human operator assigns which target the UAV is to follow by sending a target ID number using the ground station. The controller checks to see if the target with the same ID given by the human operator exists among tracks. If the target exists, the controller keeps the target in the center of the image by commanding yaw rate and forward, backward motion of the UAV. Otherwise, the controller holds the UAV's current position until it receives another target ID existing among tracks.

3.2 Recursive-RANSAC Tracker

This section describes the visual detection and tracking framework. The objective of the visual tracker is to reliably track all targets in the field of view such that the ground operator can select a desired ID number for visual servoing. All elements of target tracking are required to be autonomous, without a priori knowledge of the number of targets in the field of view. A key requirement is track continuity (persistent track ID numbers) in order for the system to achieve good following performance. An ID-loss event requires the ground operator to select the new target ID when the track is re-initialized which leads to undesirable flight behavior. No detection aids such as color segmentation or truth data are available to the controller, meaning that target detection and state estimation must be robust in standard flight environments.

Difference imaging reveals motion in the field of view by warping the previous image into the current image and comparing the two frames. This approach was used because it tends to be more robust in the presence of noisy homography transforms and image imperfections experienced

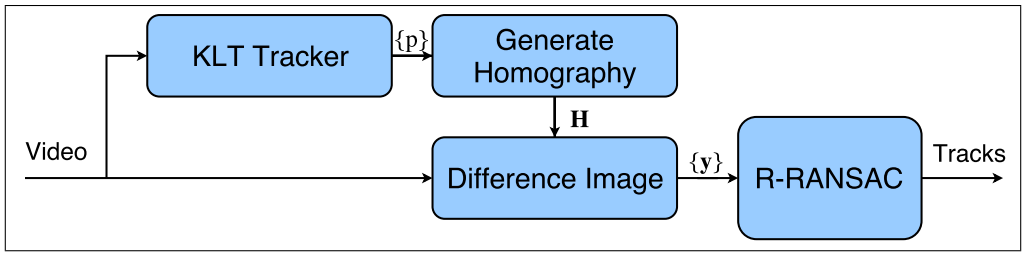


Figure 3.2: This figure illustrates the detection framework used to generate measurements used by R-RANSAC. The KLT tracker creates point correspondences between frames which are used to calculate a homography. The difference image detects motion in the frame and creates position measurements of potential targets.

by rolling shutter cameras in the presence of vibration. Our flight demonstration used entry-level hardware such as a webcam without gimbal stabalization.

This form of motion detection requires knowledge of the homography as seen in Figure 3.2. The KLT algorithm is used to create point correspondences across the image and the homography is generated from these points using a RANSAC method.

R-RANSAC is an MTT algorithm that generates many hypothesis trajectories based upon an assumed dynamic model of the targets and the set of recent measurements. By elevating models that surpass a threshold of inlier measurements, the algorithm is capable of tracking many targets with missed detections in clutter [16].

For each new set of measurements, those that are inliers to existing models are used to perform a Kalman update using a probabalistic data association (PDA) filter. For each measurement that is an outlier to all existing models, a new model is generated by sampling trajectories based on the recent history of measurements. The trajectory hypothesis with the most inliers is selected as a new model and the inlier measurements are used to estimate the target state estiamate and error covariance for the current timestep. Additional operations perform model merging and pruning in order to eliminate unlikely models.

By using the difference image measurements and R-RANSAC, track ID numbers are produced for targets in the field of view and can be used for visual servoing operations.

3.3 UAV Control

The visual tracking system in the previous section provides the control algorithm with a vector of normalized image coordinates for every track in the camera field of view. The control algorithm activates follow mode when there exists a target with the ID that a human operator has assigned for following. When the given ID is not found in the vector that the tracker provides, the control algorithm commands the UAV to hold its position until another target ID that exists among the tracks is assigned to follow. In this section, the control algorithm is described in more detail.

3.3.1 Coordinate Frame Convention

Before giving a detailed explanation of the control algorithm, it is worth clarifying our assumptions and the coordinate frames used.

First, an East-North-Up (ENU) coordinate frame is used as opposed to the common North-East-Down (NED) coordinates for UAV in order to match the frame convention used in the `mavros` package in ROS [25]. Let $\mathscr{F}^i$ be the inertial frame, which in this case coincides with the ENU frame, and let $\mathscr{F}^v$ be the vehicle frame that is translated to the UAV center of mass, with the same orientation as $\mathscr{F}^i$. Vehicle-1 frame, $\mathscr{F}^{v1}$ indicates the frame that is only rotated about the z-axis of $\mathscr{F}^v$ by ψ, the heading angle of the multirotor. The rotation matrix from $\mathscr{F}^v$ to $\mathscr{F}^{v1}$ can be expressed as R_v^{v1}. Other involved frames are the optical, camera, and body frames expressed as $\mathscr{F}^o$, $\mathscr{F}^c$, $\mathscr{F}^b$, respectively.

Second, a flat-earth model is used to properly scale the target position relative to the camera in $\mathscr{F}^{v1}$ and we assume that we have access to the correct altitude information.

Third, the displacement between the center of mass of the UAV and the focal point of the camera is ignored since it is negligible compared to the distance between the camera and the target.

Fourth, we will rely on the GPS position controller on the autopilot. The visual-servoing controller in this work computes the desired multirotor position in order to follow the target and send the position command to the autopilot. The control algorithm computes the desired multirotor velocity command in the vehicle-1 frame, but it is converted to a position command as a matter of convenience to interface with the Pixhawk autopilot.

3.3.2 Forward and heading motion control

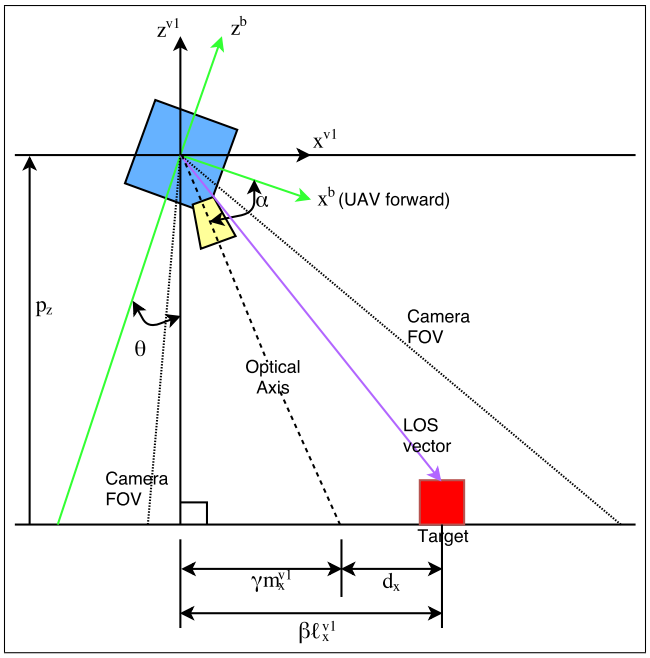


Figure 3.3: Side view of the multirotor.

The first step of the motion control is to transform the line of sight (LOS) vector to the target from $\mathscr{F}^o$ to $\mathscr{F}^{v1}$. Let

$$\ell^o = [\varepsilon_x^o, \varepsilon_y^o, 1]^\top \tag{3.3}$$

where ℓ^o is the line of sight vector in $\mathscr{F}^o$ placed at the focal length of 1. Let also the unit vector along the optical axis in $\mathscr{F}^o$ be defined as

$$\mathbf{m}^o = [0, 0, 1]^\top. \tag{3.4}$$

By applying sequential transformations to ℓ^o and $\mathbf{m}^o$, we get

$$\ell^{v1} = R_b^{v1}(\phi, \theta) R_c^b(\alpha) R_o^c \ell^o = [\ell_x^{v1}, \ell_y^{v1}, \ell_z^{v1}]^\top \tag{3.5}$$

$$\mathbf{m}^{v1} = R_b^{v1}(\phi, \theta) R_c^b(\alpha) R_o^c \mathbf{m}^o = [m_x^{v1}, m_y^{v1}, m_z^{v1}]^\top \tag{3.6}$$

where R_c^b is a matrix with fixed values depending on how the camera is mounted with respect to $\mathscr{F}^b$, and R_b^{v1} is a matrix requiring the roll and pitch angles of the multirotor. The vector ℓ^{v1} is the displacement of the target relative to the multirotor and $\mathbf{m}^{v1}$ is the optical axis in $\mathscr{F}^{v1}$. The LOS vector ℓ^{v1} and $\mathbf{m}^{v1}$ do not have proper scalings due to the unknown depth information to the target in $\mathscr{F}^o$, but can be recovered using the altitude of the camera. Let

$$\beta = \frac{p_z}{\ell_z^{v1}} \tag{3.7}$$

$$\gamma = \frac{p_z}{m_z^{v1}} \tag{3.8}$$

where p_z is the altitude of the multirotor. Then, the desired forward position from the current multirotor position can be computed as

$$d_x = \beta \ell_x^{v1} - \gamma m_x^{v1}. \tag{3.9}$$

This d_x may be decomposed into east and north components as

$$d_n = d_x \sin(\psi) \tag{3.10}$$

$$d_e = d_x \cos(\psi), \tag{3.11}$$

where ψ is the heading of the multirotor. These north and east components are added to the current multirotor east and north positions, and the sum is sent to the autopilot position controller.

It is more suitable to compensate for a target moving horizontally in the image plane by adjusting the multirotor's heading than through lateral motion. Thus, a yaw rate command ω_z can be computed as

$$\omega_z = \eta \ell_y^{v1}, \tag{3.12}$$

where $\eta > 0$ is a control gain.

3.4 Experiments and Results

The hardware is comprised of a 3DR X8 multirotor platform, a small form factor Gigabyte BRIX GB-BXi7-4500 (no GPU), low-cost USB camera, ELP-USBFHD01M-L21 (rolling shutter), and Pixhawk with PX4 firmware. The proposed system was tested in an outdoor environment to track realistic targets (people). We demonstrate the ability to follow one of the multiple tracked objects while switching the target of interest in real-time. The tracker does not have any prior information about what tracks look like and how they might move. The hardware result shows that the R-RANSAC tracker is able to track multiple targets and to provide the controller with proper coordinates of the targets. It also shows that the controller is able to follow one of the targets while keeping it in the camera field of view. As shown in Figure 3.4a, the human operator sees this camera view of multiple moving objects being tracked from the ground station and commands the multirotor to follow a target of interest by sending the correct track ID (Figure 3.4b). The hardware test lasted about 1 minute and in the middle (t=35s) the operator commanded the multirotor to follow a different target (Figure 3.4d). Until another track ID is commanded by the operator, the multirotor keeps following the current assigned target (Figure 3.4e).

Figure 3.5 and 3.6 show the effort of the multirotor trying to place the target being followed in the center of the image plane. In Figure 3.5 and Figure 3.6, the numbered events listed in Figure 3.4 are illustrated in the image plane. A track with ID 51 is initialized by the R-RANSAC tracker, and later the multirotor was commanded to follow track 51. The R-RANSAC tracker detected another moving object in the camera field of view and started to track the object with ID 65 while the multirotor was still following track 51. After a while, the human operator switches the target of interest resulting in the multirotor following track 65. The multirotor kept following the track 65 for the rest of the experiment. Finally, Figure 3.7 illustrates the GPS coordinates of the multirotor to show its movement and heading during the experiment.

3.5 Conclusion

In this chapter, a novel vision-based target following system with the R-RANSAC tracker is presented with hardware demonstration. The experimental result shows the feasibility of the real-time system in a realistic outdoor environment. With the R-RANSAC tracker, multiple moving

(a) Track ID 51 initiated by R-RANSAC tracker (t=0s)

(b) The human operator has commanded the UAV to follow track ID 51 (t=13s)

(c) Track ID 65 initiated by R-RANSAC tracker (t=27s)

(d) The human operator has commanded the UAV to follow track ID 65 (t=35s)

(e) A snapshot of the track ID 65 being followed (t=60s)

Figure 3.4: Camera view at various events

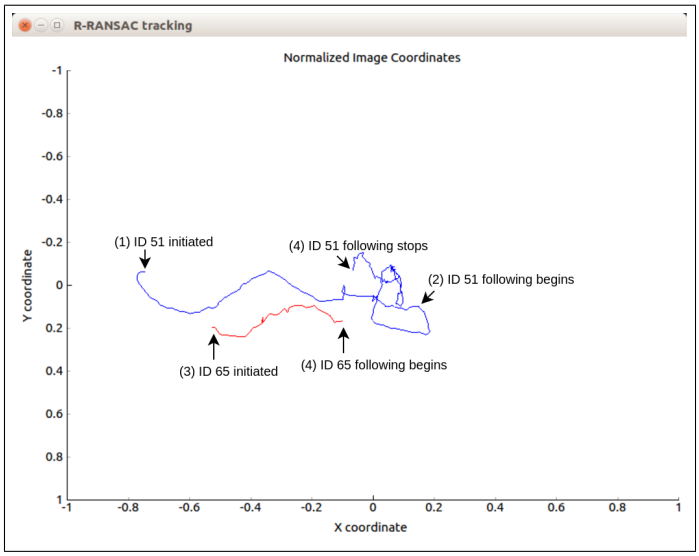


Figure 3.5: Tracks movement in the normalized image plane. Each event (1)-(4) corresponds to camera view in 3.4a-3.4d respectively. Until the command to follow ID 65, the multirotor keeps the track ID 51 from leaving the camera view.

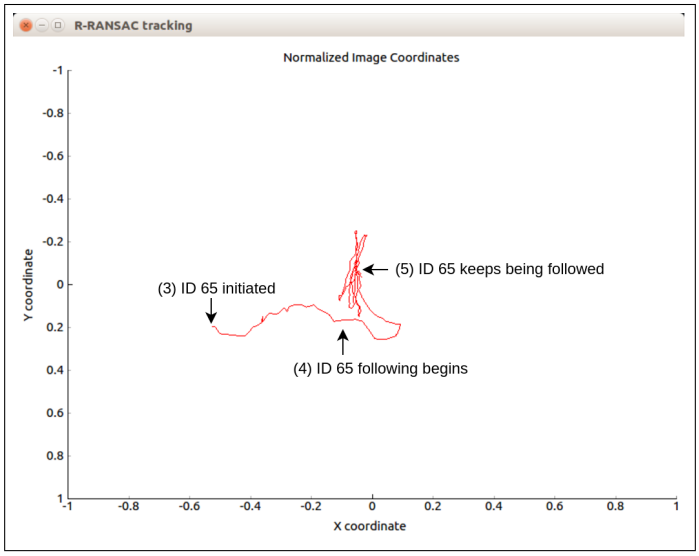


Figure 3.6: The movement of track ID 65 in the normalized image plane. Each event (3)-(5) corresponds to camera view in 3.4c-3.4e respectively. The controller keeps the track ID 65 in the camera field of view after receiving the command to do so from the human operator.

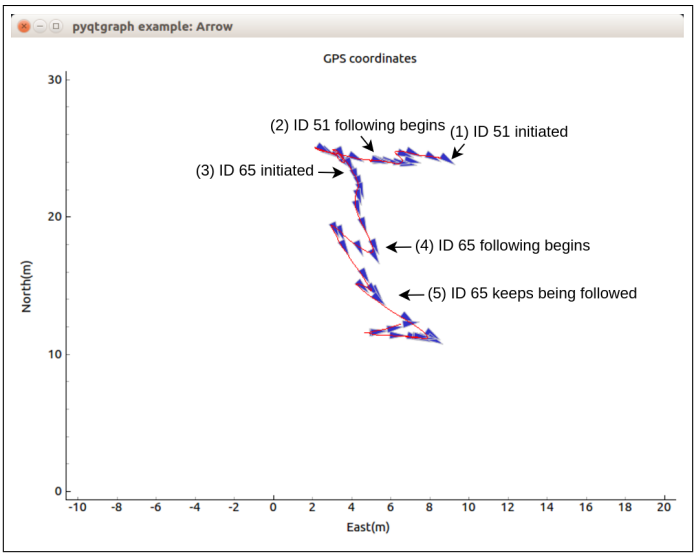


Figure 3.7: Multirotor GPS footage and heading corresponding to camera view in 3.4a-3.4e respectively.

objects in the camera view are tracked without having to know their colors or shapes. The controller is able to follow any particular target among the tracks with minimum effort to the human operator. The human operator is only expected to send a track ID number to the controller in order for the multirotor to follow the target of interest. This research opens up many other potential areas of research such as keeping multiple targets in the camera field of view, human machine interaction and multi UAS coordination in multiple target tracking situations.

CHAPTER 4. UNIT VECTOR UAV VISUAL SERVOING

4.1 Motivation

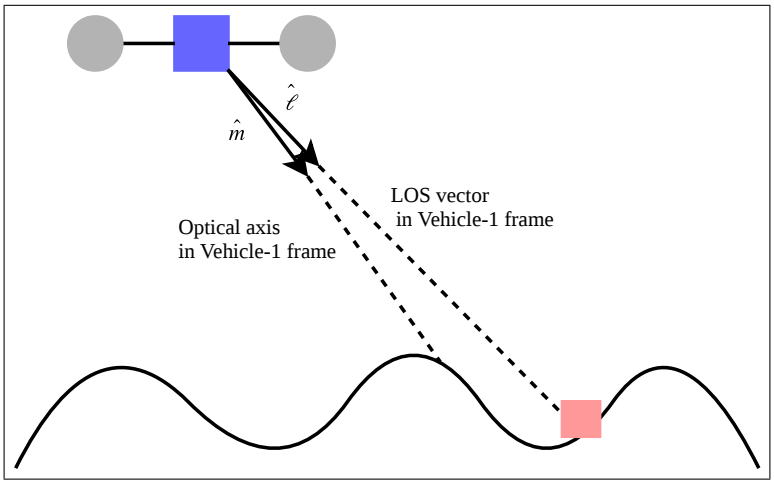


Figure 4.1: Non-flat-earth model example. The unit optical axis vector $\hat{m}$ and the unit line of sight vector $\hat{\ell}$ are key components of the controller presented in this chapter.

The UAV control algorithm introduced in the previous chapter is relatively simple and easy to implement. However, the controller makes strong assumption that can be unrealistic in some cases. For example, it assumes that the line of sight (LOS) vector to the target and the optical axis terminate on the same flat surface which is often called 'flat-earth assumption.' In that case, the altitude of the multirotor acts as a scale factor that can be used to compute how much the UAV should move forward. The more advanced UAV controller presented in this chapter overcomes the flat-earth assumption by working with the unit LOS vector and unit optical axis vector (see Figure 4.1). Also, the controller does not require the altitude of the multirotor to be known because the scale factor does not have to be recovered.

4.2 Unit Vector UAV Visual Servoing Derivation for Simple UAV Dynamics

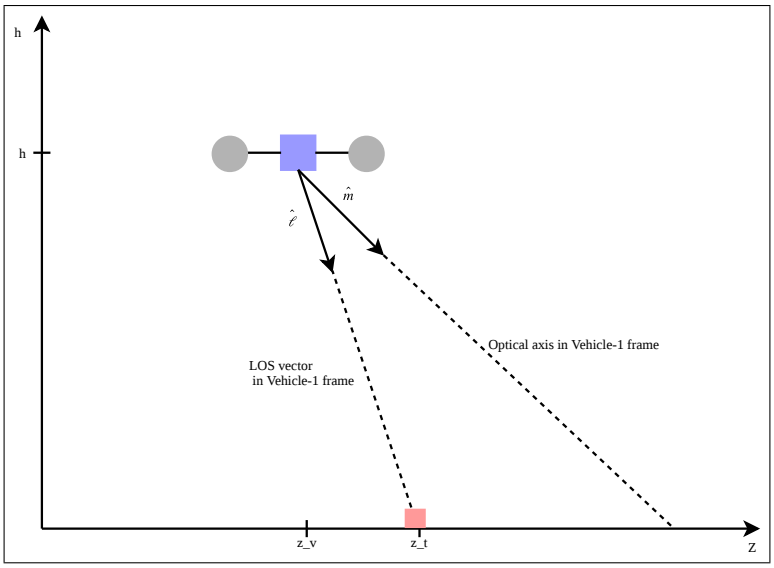


Figure 4.2: Graphical overview of the problem

As a first step in developing the control algorithm, we start with the simple vehicle dynamics

$$\ddot{z}_v = u_1 \tag{4.1}$$

$$\ddot{h} = u_2 \tag{4.2}$$

where z_v is multirotor's horizontal position and h is its altitude. A required input to the proposed controller is the measured unit LOS vector $\hat{\ell}$. Let $\hat{m}$ be the unit vector aligned with the optical axis, and let the projection matrix onto the null space of $\hat{m}$ be

$$P_{\hat{m}} = (I - \hat{m}\hat{m}^\top). \tag{4.3}$$

Selecting the coordinate frame as $(\hat{z},\ \hat{z}\times\hat{h},\ \hat{h})$, $\hat{m}$ can be represented as

$$\hat{m} = \begin{pmatrix} m_1 & 0 & -m_2 \end{pmatrix}^\top. \tag{4.4}$$

In that case, we have

$$P_{\hat{m}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} m_1 \\ 0 \\ -m_2 \end{pmatrix} \begin{pmatrix} m_1 & 0 & -m_2 \end{pmatrix} \tag{4.5}$$

$$= \begin{pmatrix} 1-m_1^2 & 0 & m_1 m_2 \\ 0 & 1 & 0 \\ m_1 m_2 & 0 & 1-m_2^2 \end{pmatrix}. \tag{4.6}$$

Note that the controller is developed as if the multirotor is in 3D plane even if we only show 2D dynamics. The objective is to drive the horizontal component of $P_{\hat{m}}\hat{\ell}$ to zero. If we let

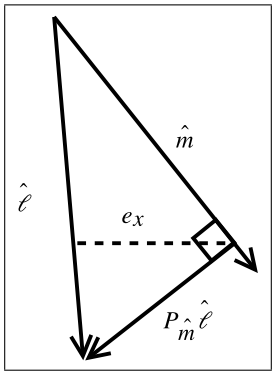


Figure 4.3: Projection onto the null space of the optical axis unit vector

$$\hat{e}_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^\top, \tag{4.7}$$

the horizontal component of $P_{\hat{m}}\hat{\ell}$ which is denoted by e_x can be expressed as

$$e_x = \hat{e}_1^\top P_{\hat{m}} \hat{\ell}. \tag{4.8}$$

Note that since $\hat{m}$ is fixed and known and $\hat{\ell}$ is measured, e_x can also be measured. Also, since $\dot{\hat{\ell}}$ can be approximated by differentiating numerically,

$$\dot{e}_x = \hat{e}_1^\top P_{\hat{m}} \dot{\hat{\ell}} \tag{4.9}$$

is measurable. Meanwhile, the line of sight vector is

$$\ell = \begin{pmatrix} z_t \\ 0 \\ 0 \end{pmatrix} - \begin{pmatrix} z_v \\ 0 \\ h \end{pmatrix}. \tag{4.10}$$

Differentiating equation (4.10) gives

$$\dot{\ell} = \begin{pmatrix} \dot{z}_t \\ 0 \\ 0 \end{pmatrix} - \begin{pmatrix} \dot{z}_t \\ 0 \\ \dot{h} \end{pmatrix}, \tag{4.11}$$

and

$$\ddot{\ell} = \begin{pmatrix} \ddot{z}_t \\ 0 \\ 0 \end{pmatrix} - \begin{pmatrix} \ddot{z}_v \\ 0 \\ \ddot{h} \end{pmatrix}. \tag{4.12}$$

Assuming that the velocity of the target and the UAV altitude (i.e. $u_2 = 0$) are not changing yields

$$\ddot{\ell} = \begin{pmatrix} -\ddot{z}_v \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} -u_1 \\ 0 \\ 0 \end{pmatrix}. \tag{4.13}$$

Thus, combining equations (4.6), (4.7) and (4.13) results in

$$\hat{e}_1^\top P_{\hat{m}} \ddot{\ell} = -(1 - m_1^2) u_1. \tag{4.14}$$

Let

$$L = \|\ell\|, \tag{4.15}$$

be the unknown depth between the camera and the target, or the norm of the LOS vector. Then, the unit LOS vector is

$$\hat{\ell} = \frac{\ell}{L}. \tag{4.16}$$

Differentiating equation (4.16) gives

$$\dot{\hat{\ell}} = \frac{\dot{\ell}L - \ell\dot{L}}{L^2} = \frac{\dot{\ell}}{L} - \hat{\ell}\left(\frac{\dot{L}}{L}\right). \tag{4.17}$$

Differentiating again gives

$$\ddot{\hat{\ell}} = \frac{\ddot{\ell}L - \dot{\ell}\dot{L}}{L^2} - \dot{\hat{\ell}}\left(\frac{\dot{L}}{L}\right) - \hat{\ell}\left(\frac{\ddot{L}L - \dot{L}^2}{L^2}\right) \tag{4.18}$$

$$= \frac{\ddot{\ell}}{L} - \left(\frac{\dot{\ell}}{L}\right)\left(\frac{\dot{L}}{L}\right) - \dot{\hat{\ell}}\left(\frac{\dot{L}}{L}\right) - \hat{\ell}\left(\frac{\ddot{L}}{L}\right) + \hat{\ell}\left(\frac{\dot{L}}{L}\right)^2. \tag{4.19}$$

Plugging in for $\frac{\dot{\ell}}{L} = \dot{\hat{\ell}} + \hat{\ell}\left(\frac{\dot{L}}{L}\right)$ from equation (4.17) gives

$$\ddot{\hat{\ell}} = \frac{\ddot{\ell}}{L} - 2\dot{\hat{\ell}}\left(\frac{\dot{L}}{L}\right) - \hat{\ell}\left(\frac{\ddot{L}}{L}\right). \tag{4.20}$$

Differentiating equation (4.9) and using equation (4.20) for $\ddot{\hat{\ell}}$ yields

$$\ddot{e}_x = \hat{e}_1^\top P_{\hat{m}} \ddot{\hat{\ell}} \tag{4.21}$$

$$= \hat{e}_1^\top P_{\hat{m}} \left(\frac{\ddot{\ell}}{L} - 2\dot{\hat{\ell}}\left(\frac{\dot{L}}{L}\right) - \hat{\ell}\left(\frac{\ddot{L}}{L}\right)\right) \tag{4.22}$$

$$= \frac{1}{L}(\hat{e}_1^\top P_{\hat{m}} \ddot{\ell}) + \frac{\dot{L}}{L}(-2\hat{e}_1^\top P_{\hat{m}} \dot{\hat{\ell}}) + \frac{\ddot{L}}{L}(-\hat{e}_1^\top P_{\hat{m}} \hat{\ell}). \tag{4.23}$$

Substituting (4.14) for $\hat{e_1}^\top P_{\hat{m}} \ddot{\ell}$ results in

$$\ddot{e}_x = -\frac{1}{L}(1 - m_1^2)u_1 + \frac{\dot{L}}{L}(-2\hat{e}_1^\top P_{\hat{m}} \dot{\hat{\ell}}) + \frac{\ddot{L}}{L}(-\hat{e}_1^\top P_{\hat{m}} \hat{\ell}). \tag{4.24}$$

Defining the unknown quantities

$$\beta_1 \triangleq \frac{1}{L}, \quad \beta_2 \triangleq \frac{\dot{L}}{L}, \quad \beta_3 \triangleq \frac{\ddot{L}}{L} \tag{4.25}$$

and the measured quantities

$$\phi_1 = 1 - m_1^2, \quad \phi_2 = -2\hat{e}_1^\top P_{\hat{m}} \dot{\hat{\ell}}, \quad \phi_3 = -\hat{e}_1^\top P_{\hat{m}} \hat{\ell}. \tag{4.26}$$

we can rewrite

$$\ddot{e}_x = -\beta_1 \phi_1 u_1 + \beta_2 \phi_2 + \beta_3 \phi_3. \tag{4.27}$$

In order to drive e_x to zero, define

$$s = \dot{e}_x + k e_x \tag{4.28}$$

where $k > 0$. The reason why the equation (4.28) is selected is because when $s \to 0$,

$$\dot{e}_x = -k e_x \tag{4.29}$$

which makes e_x asymptotically stable. Thus, if we can find the control input u_1 that makes $s \to 0$, e_x will converge to zero. Differentiating equation (4.28) and substituting (4.27) for $\ddot{e}_x$ yields

$$\dot{s} = \ddot{e}_x + k\dot{e}_x \tag{4.30}$$

$$= -\beta_1 \phi_1 u_1 + \beta_2 \phi_2 + \beta_3 \phi_3 + k\dot{e}_x. \tag{4.31}$$

If β_1, β_2, and β_3 are known, then the ideal control input would be

$$u_1 = \frac{1}{\beta_1 \phi_1}(\beta_2 \phi_2 + \beta_3 \phi_3 + k\dot{e}_x + \alpha s) \tag{4.32}$$

where $\alpha > 0$. Then,

$$\dot{s} = -\alpha s \tag{4.33}$$

which makes s asymptotically stable. However, since we do not know β_1, β_2, and β_3, we use the control input

$$u_1 = \frac{1}{\hat{\beta}_1 \phi_1}(\hat{\beta}_2 \phi_2 + \hat{\beta}_3 \phi_3 + k\dot{e}_x + \alpha s) \tag{4.34}$$

where $\hat{\beta}_1$, $\hat{\beta}_2$, and $\hat{\beta}_3$ are the estimates of β_1, β_2, and β_3 respectively. In that case, we have

$$\dot{s} = -\beta_1\phi_1 u_1 + \hat{\beta}_1\phi_1 u_1 - \hat{\beta}_1\phi_1 u_1 + \beta_2\phi_2 + \beta_3\phi_3 + k\dot{e}_x \tag{4.35}$$

$$= -(\beta_1 - \hat{\beta}_1)\phi_1 u_1 + (\beta_2 - \hat{\beta}_2)\phi_2 + (\beta_3 - \hat{\beta}_3)\phi_3 - \alpha s \tag{4.36}$$

$$= -\tilde{\beta}_1\phi_1 u_1 + \tilde{\beta}_2\phi_2 + \tilde{\beta}_3\phi_3 - \alpha s \tag{4.37}$$

$$= \tilde{B}^\top \Phi - \alpha s \tag{4.38}$$

where

$$B \triangleq \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix}, \quad \hat{B} \triangleq \begin{pmatrix} \hat{\beta}_1 \\ \hat{\beta}_2 \\ \hat{\beta}_3 \end{pmatrix}, \quad \tilde{B} \triangleq B - \hat{B}, \quad \Phi \triangleq \begin{pmatrix} -\phi_1 u_1 \\ \phi_2 \\ \phi_3 \end{pmatrix}. \tag{4.39}$$

Selecting the Lyapunov function

$$V = \frac{1}{2}s^2 + \frac{1}{2}\tilde{B}^\top \Gamma^{-1}\tilde{B}, \tag{4.40}$$

and differentiating with respect to time gives

$$\dot{V} = s\dot{s} + \tilde{B}^\top \Gamma^{-1}\dot{\tilde{B}} \tag{4.41}$$

$$= s(\tilde{B}^\top \Phi - \alpha s) + \tilde{B}^\top \Gamma^{-1}\dot{\tilde{B}} \tag{4.42}$$

$$= -\alpha s^2 + \tilde{B}^\top (s\Phi + \Gamma^{-1}\dot{\tilde{B}}). \tag{4.43}$$

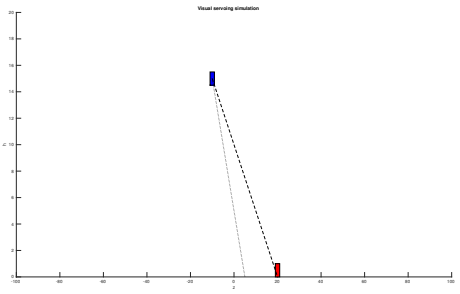
Assuming B is roughly constant or slowly varying (i.e. $\dot{B} = 0$), we have that $\dot{\tilde{B}} = -\dot{\hat{B}}$ implying that

$$\dot{V} = -\alpha s^2 + \tilde{B}^\top (s\Phi - \Gamma^{-1}\dot{\hat{B}}). \tag{4.44}$$

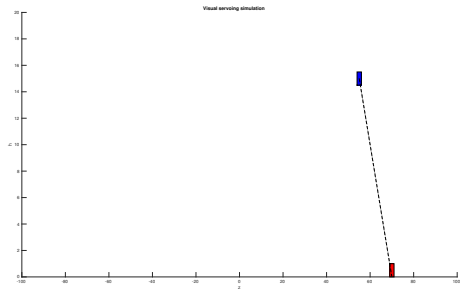
Therefore, by choosing $\dot{\hat{B}} = s\Gamma\Phi$, gives

$$\dot{V} = -\alpha s^2 \tag{4.45}$$

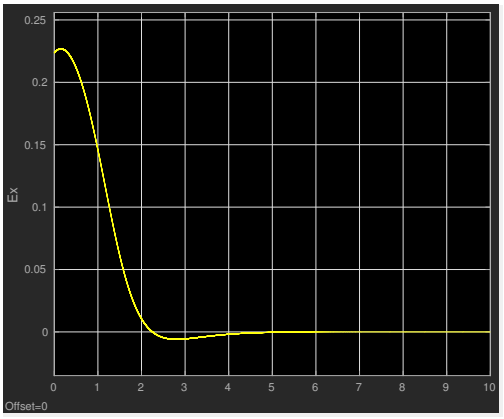
which implies that s is asymptotically stable. Thus, the original objective of stabilizing e_x is achieved. This control scheme is simulated in Simulink and Figures 4.4 and 4.5 show the performance of the controller for one initial condition.



(a) when $t = 0s$



(b) when $t = 10s$



(c) The horizontal error between the unit LOS vector and the unit optical axis vector converges to zero.

Figure 4.4: Simple UAV dynamics visual servoing Simulink simulation. The blue square is flying UAV at constant altitude and the red square is a target on the ground moving at $5m/s$. The initial UAV and target positions are [-10, 15] and [20, 0] respectively. Tuning parameters are set to $k = 1$, $\Gamma = I_3$ (identity matrix), and $\alpha = 1000$.

4.3 Backstepping Controller Derivation for Multirotor Dynamics

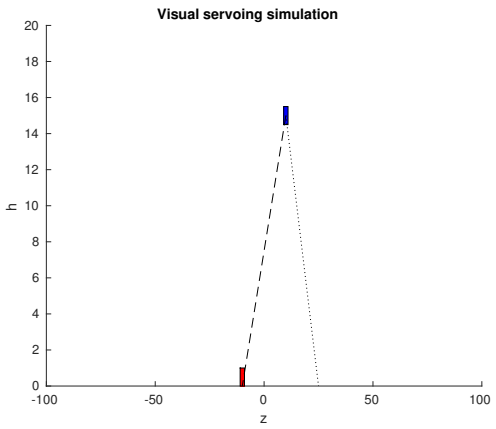
4.3.1 Derivation

The multirotor dynamics expressed in vehicle-1 frame [26] is given by

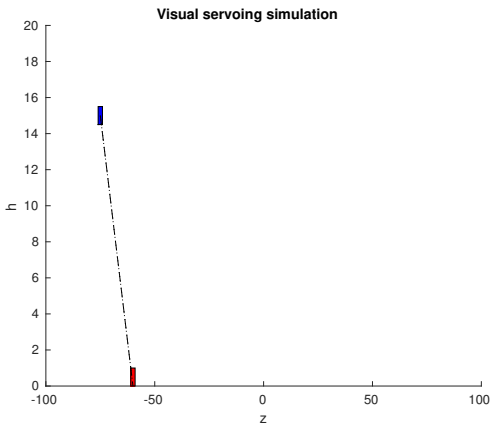
$$\ddot{p}_x = -\cos\phi \sin\theta \frac{F}{m} \tag{4.46}$$

$$\ddot{p}_y = \sin\phi \frac{F}{m} \tag{4.47}$$

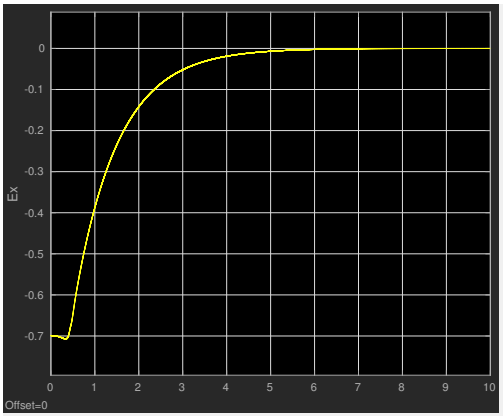
$$\ddot{p}_z = g - \cos\phi \cos\theta \frac{F}{m} \tag{4.48}$$



(a) when $t = 0s$



(b) when $t = 10s$



(c) The horizontal error between the unit LOS vector and the unit optical axis vector converges to zero.

Figure 4.5: Simple UAV dynamics visual servoing Simulink simulation. The blue square is flying UAV at constant altitude and the red square is a target on the ground moving at $-5m/s$. The initial UAV and target positions are [10, 15] and [-10, 0] respectively. Tuning parameters are set to $k = 1$, $\Gamma = I_3$ (identity matrix), and $\alpha = 1000$.

$$\ddot{\phi} = \frac{1}{J_x}\tau_\phi \tag{4.49}$$

$$\ddot{\theta} = \frac{1}{J_y}\tau_\theta \tag{4.50}$$

$$\ddot{\psi} = \frac{1}{J_z}\tau_\psi \tag{4.51}$$

where m is the mass of multirotor and J_x, J_y, and, J_z are the moments of inertia. Also, F and τ are force and torque produced by the propellers. For our specific control interest, we are only concerned with forward and backward motion of the multirotor involving only the equation (4.46)

and (4.50). We also assume that the altitude of the multirotor is controlled with a separate altitude controller and the roll of multirotor, ϕ, is controlled to be zero which means that the multirotor is restricted from moving sideways. When the target moves left or right, the multirotor yaws to keep the target in the camera's view and it is assumed that there is a controller for this. Applying the above conditions and using small angle approximation to linearize, the equation (4.46) becomes

$$\ddot{p}_x = -\frac{F_e}{m}\theta \tag{4.52}$$

where F_e is the force to keep the multirotor in a constant altitude hover when the roll and pitch angles are zero. The line of sight vector is given by

$$\ell = \begin{pmatrix} p_x^{tar} \\ 0 \\ 0 \end{pmatrix} - \begin{pmatrix} p_x^{uav} \\ 0 \\ p_z^{uav} \end{pmatrix}. \tag{4.53}$$

Differentiating (4.53) we get

$$\dot{\ell} = \begin{pmatrix} \dot{p}_x^{tar} \\ 0 \\ 0 \end{pmatrix} - \begin{pmatrix} \dot{p}_x^{uav} \\ 0 \\ \dot{p}_z^{uav} \end{pmatrix}, \tag{4.54}$$

and

$$\ddot{\ell} = \begin{pmatrix} \ddot{p}_x^{tar} \\ 0 \\ 0 \end{pmatrix} - \begin{pmatrix} \ddot{p}_x^{uav} \\ 0 \\ \ddot{p}_z^{uav} \end{pmatrix}. \tag{4.55}$$

Referring back to (4.6), (4.7) and (4.55) we get that

$$\hat{e}_1^\top P_{\hat{m}} \ddot{\ell} = (1-m_1^2)(\ddot{p}_x^{tar} - \ddot{p}_x^{uav}) + m_1 m_2 \ddot{p}_z^{uav} \tag{4.56}$$

$$= (1-m_1^2)\frac{F_e}{m}\theta \tag{4.57}$$

assuming that the multirotor keeps its altitude constant and the target does not accelerate. Notice that

$$P_{\hat{m}} = \begin{pmatrix} 1-m_1^2 & 0 & -m_1 m_2 \\ 0 & 1 & 0 \\ -m_1 m_2 & 0 & 1-m_2^2 \end{pmatrix} \tag{4.58}$$

which is different from (4.6) because now the z component of the optical axis unit vector in vehicle-1 frame aligns with z axis of the vehicle-1 frame (i.e. NED coordinate frame). However, this turns out to not affect the controller derivation. Again, the control objective is to drive e_x to zero and this can be achieved by driving $s = \dot{e}_x + k e_x$ to zero. Recalling (4.27) we get

$$\ddot{e}_x = \frac{1}{L}(1-m_1^2)\frac{F_e}{m}\theta + \frac{\dot{L}}{L}(-2\hat{e}_1^\top P_{\hat{m}}\dot{\hat{\ell}}) + \frac{\ddot{L}}{L}(-\hat{e}_1^\top P_{\hat{m}}\hat{\ell}) \tag{4.59}$$

$$= \beta_1\phi_1\frac{F_e}{m}\theta + \beta_2\phi_2 + \beta_3\phi_3 \tag{4.60}$$

where

$$\beta_1 \triangleq \frac{1}{L}, \quad \beta_2 \triangleq \frac{\dot{L}}{L}, \quad \beta_3 \triangleq \frac{\ddot{L}}{L} \tag{4.61}$$

and

$$\phi_1 \triangleq 1-m_1^2, \quad \phi_2 \triangleq -2\hat{e}_1^\top P_{\hat{m}}\dot{\hat{\ell}}, \quad \phi_3 \triangleq -\hat{e}_1^\top P_{\hat{m}}\hat{\ell}. \tag{4.62}$$

Since we have all necessary components, we can get

$$\dot{s} = \ddot{e}_x + k\dot{e}_x \tag{4.63}$$

$$= \beta_1\phi_1\frac{F_e}{m}\theta + \beta_2\phi_2 + \beta_3\phi_3 + k\dot{e}_x. \tag{4.64}$$

Manipulating (4.64) yields

$$\dot{s} = \beta_1\phi_1\frac{F_e}{m}\theta + \beta_2\phi_2 + \beta_3\phi_3 + k\dot{e}_x + \hat{\beta}_1\phi_1\frac{F_e}{m}\theta + \hat{\beta}_2\phi_2 + \hat{\beta}_3\phi_3 - \hat{\beta}_1\phi_1\frac{F_e}{m}\theta - \hat{\beta}_2\phi_2 - \hat{\beta}_3\phi_3 \tag{4.65}$$

$$= (\beta_1-\hat{\beta}_1)\phi_1\frac{F_e}{m}\theta + (\beta_2-\hat{\beta}_2)\phi_2 + (\beta_3-\hat{\beta}_3)\phi_3 + k\dot{e}_x + \hat{\beta}_1\phi_1\frac{F_e}{m}\theta + \hat{\beta}_2\phi_2 + \hat{\beta}_3\phi_3 \tag{4.66}$$

$$= \tilde{\beta}_1\phi_1\frac{F_e}{m}\theta + \tilde{\beta}_2\phi_2 + \tilde{\beta}_3\phi_3 + k\dot{e}_x + \hat{\beta}_1\phi_1\frac{F_e}{m}\theta + \hat{\beta}_2\phi_2 + \hat{\beta}_3\phi_3 \tag{4.67}$$

$$= \tilde{B}^\top\Phi + k\dot{e}_x + \hat{\beta}_1\phi_1\frac{F_e}{m}\theta + \hat{\beta}_2\phi_2 + \hat{\beta}_3\phi_3 \tag{4.68}$$

where

$$B \triangleq \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix}, \quad \hat{B} \triangleq \begin{pmatrix} \hat{\beta}_1 \\ \hat{\beta}_2 \\ \hat{\beta}_3 \end{pmatrix}, \quad \tilde{B} \triangleq B - \hat{B}, \quad \Phi \triangleq \begin{pmatrix} \phi_1 \frac{F_e}{m} \theta \\ \phi_2 \\ \phi_3 \end{pmatrix}. \tag{4.69}$$

Define the Lyapunov function candidate as

$$V_1 = \frac{1}{2} s^2 + \frac{1}{2} \tilde{B}^\top \Gamma^{-1} \tilde{B} \tag{4.70}$$

and take the derivative to get

$$\dot{V}_1 = s\dot{s} + \tilde{B}^\top \Gamma^{-1} \dot{\tilde{B}} \tag{4.71}$$

$$= s(\tilde{B}^\top \Phi + k\dot{e}_x + \hat{\beta}_1 \phi_1 \frac{F_e}{m} \theta + \hat{\beta}_2 \phi_2 + \hat{\beta}_3 \phi_3) + \tilde{B}^\top \Gamma^{-1} \dot{\tilde{B}} \tag{4.72}$$

$$= s(k\dot{e}_x + \hat{\beta}_1 \phi_1 \frac{F_e}{m} \theta + \hat{\beta}_2 \phi_2 + \hat{\beta}_3 \phi_3) + s\tilde{B}^\top \Phi + \tilde{B}^\top \Gamma^{-1} \dot{\tilde{B}}. \tag{4.73}$$

Assuming that B is constant or slowly changing, then $\dot{\tilde{B}} = -\dot{\hat{B}}$, and equation (4.73) becomes

$$\dot{V}_1 = s(k\dot{e}_x + \hat{\beta}_1 \phi_1 \frac{F_e}{m} \theta + \hat{\beta}_2 \phi_2 + \hat{\beta}_3 \phi_3) + s\tilde{B}^\top \Phi - \tilde{B}^\top \Gamma^{-1} \dot{\hat{B}}. \tag{4.74}$$

By selecting $\dot{\hat{B}} = s\Gamma\Phi$, the above equation further becomes

$$\dot{V}_1 = s(k\dot{e}_x + \hat{\beta}_1 \phi_1 \frac{F_e}{m} \theta + \hat{\beta}_2 \phi_2 + \hat{\beta}_3 \phi_3). \tag{4.75}$$

For some cases, θ is the direct control input that can be commanded making the problem less complicated. However, for the multirotor dynamics, θ is indirectly controlled by torque τ_θ which leads to the development of backstepping control technique. The backstepping control is especially useful when the system has cascaded structure such as this multirotor case by using some of the state variables as pseudo controls to backstep until the final external control input is reached. With backstepping, a Lyapunov function is derived for the entire system [27], [28]. We start by adding and subtracting a pseudo control ξ_1 to equation (4.75) as

$$\dot{V}_1 = s(k\dot{e}_x + \hat{\beta}_1 \phi_1 \frac{F_e}{m} \theta + \hat{\beta}_2 \phi_2 + \hat{\beta}_3 \phi_3 + \xi_1 - \xi_1). \tag{4.76}$$

By selecting $\xi_1 = -k\dot{e}_x - \hat{\beta}_2\phi_2 - \hat{\beta}_3\phi_3 - k_1 s$,

$$\dot{V}_1 = -k_1 s^2 + s(\hat{\beta}_1\phi_1\frac{F_e}{m}\theta - \xi_1). \tag{4.77}$$

Define a new Lyapunov function candidate as

$$V_2 = V_1 + \frac{1}{2}(\hat{\beta}_1\phi_1\frac{F_e}{m}\theta - \xi_1)^2, \tag{4.78}$$

and taking derivative, we get

$$\dot{V}_2 = \dot{V}_1 + (\hat{\beta}_1\phi_1\frac{F_e}{m}\theta - \xi_1)(\phi_1\frac{F_e}{m}(\dot{\hat{\beta}}_1\theta + \hat{\beta}_1\dot{\theta}) - \dot{\xi}_1) \tag{4.79}$$

$$= -k_1 s^2 + (\hat{\beta}_1\phi_1\frac{F_e}{m}\theta - \xi_1)(s + \phi_1\frac{F_e}{m}(\dot{\hat{\beta}}_1\theta + \hat{\beta}_1\dot{\theta}) - \dot{\xi}_1 + \xi_2 - \xi_2). \tag{4.80}$$

Assigning $\xi_2 = \dot{\xi}_1 - s - \phi_1\frac{F_e}{m}\dot{\hat{\beta}}_1\theta - k_2(\hat{\beta}_1\phi_1\frac{F_e}{m}\theta - \xi_1)$ leads to

$$\dot{V}_2 = -k_1 s^2 - k_2(\hat{\beta}_1\phi_1\frac{F_e}{m}\theta - \xi_1)^2 + (\hat{\beta}_1\phi_1\frac{F_e}{m}\theta - \xi_1)(\phi_1\frac{F_e}{m}\hat{\beta}_1\dot{\theta} - \xi_2). \tag{4.81}$$

Finally, define a new Lyapunov function as

$$V_3 = V_2 + \frac{1}{2}(\phi_1\frac{F_e}{m}\hat{\beta}_1\dot{\theta} - \xi_2)^2, \tag{4.82}$$

and differentiate to get

$$\dot{V}_3 = \dot{V}_2 + (\phi_1\frac{F_e}{m}\hat{\beta}_1\dot{\theta} - \xi_2)(\phi_1\frac{F_e}{m}\dot{\hat{\beta}}_1\dot{\theta} + \phi_1\frac{F_e}{m}\hat{\beta}_1\ddot{\theta} - \dot{\xi}_2) \tag{4.83}$$

$$= -k_1 s^2 - k_2(\hat{\beta}_1\phi_1\frac{F_e}{m}\theta - \xi_1)^2 + (\phi_1\frac{F_e}{m}\hat{\beta}_1\dot{\theta} - \xi_2)(\hat{\beta}_1\phi_1\frac{F_e}{m}\theta - \xi_1 + \phi_1\frac{F_e}{m}\dot{\hat{\beta}}_1\dot{\theta} + \phi_1\frac{F_e}{m}\hat{\beta}_1\ddot{\theta} - \dot{\xi}_2). \tag{4.84}$$

Using the relationship between the angular acceleration and torque in (4.50), the above equation becomes

$$\dot{V}_3 = -k_1 s^2 - k_2(\hat{\beta}_1\phi_1\frac{F_e}{m}\theta - \xi_1)^2 + (\phi_1\frac{F_e}{m}\hat{\beta}_1\dot{\theta} - \xi_2)(\hat{\beta}_1\phi_1\frac{F_e}{m}\theta - \xi_1 + \phi_1\frac{F_e}{m}\dot{\hat{\beta}}_1\dot{\theta} + \phi_1\frac{F_e}{m}\hat{\beta}_1\frac{1}{J_y}\tau_\theta - \dot{\xi}_2) \tag{4.85}$$

Notice that τ_θ finally showed up in the equation (4.85) which we can directly control. Thus, by setting

$$\tau_\theta = \frac{J_y m}{\phi_1 F_e \hat{\beta}_1}(\xi_1 + \dot{\xi}_2 - \hat{\beta}_1 \phi_1 \frac{F_e}{m}\theta - \phi_1 \frac{F_e}{m}\dot{\hat{\beta}}_1 \dot{\theta} - k_3(\phi_1 \frac{F_e}{m}\hat{\beta}_1 \dot{\theta} - \xi_2)), \tag{4.86}$$

gives

$$\dot{V}_3 = -k_1 s^2 - k_2(\hat{\beta}_1 \phi_1 \frac{F_e}{m}\theta - \xi_1)^2 - k_3(\phi_1 \frac{F_e}{m}\hat{\beta}_1 \dot{\theta} - \xi_2)^2. \tag{4.87}$$

Because of the cascaded structure of the backstepping control, the fact that $V_3(t) \to 0$ means that $V_1(t)$ also converges to zero.

4.3.2 Simulation

The backstepping control algorithm derived above is simulated in Simulink with more realistic multirotor dynamics that can be found in [26]. The simulation only tests the forward motion control of the multirotor by having separate PID controller to keep roll and yaw motions unmoved. There are two types of backstepping controller tested. Both cases the same backstepping controller is used and it only requires the unit optical axis vector and unit LOS vector as inputs to compute the multirotor pitch torque output. The first is using the inertial LOS vector and normalize it to get the unit LOS vector shown in Figure 4.6. This is rather unrealistic because it is usually not feasible to know the target location in the inertial frame to get the inertial LOS vector. However, this is still tested as an intermediate step and is used to test the validity of the backstepping control algorithm itself. The simulation results with various target velocity $0m/s$, $5m/s$, and $-5m/s$ can be found in Figure 4.8, 4.9, and 4.10 respectively. Figure 4.8 shows that the backstepping controller is capable of keeping the static target in the camera's field of view using the unit vector visual servoing framework. Also, the horizontal error between the unit LOS vector and the unit optical axis vector both in the vehicle-1 frame converges to zero. Figure 4.9 and 4.10 show that the backstepping controller can keep the target moving at $5m/s$ and $-5m/s$ in the camera's view. In both cases, the target is not placed at the center of image because the horizontal error is computed in the vehicle-1 frame meaning that the pitch of multirotor is not compensated. In the future with further research, the target can be placed at the image center by adding a simple pitch gimbal.

The second backstepping controller is the actual simulation of the hardware demonstration. When the multirotor is equipped with monocular camera only, the multirotor has no other way to

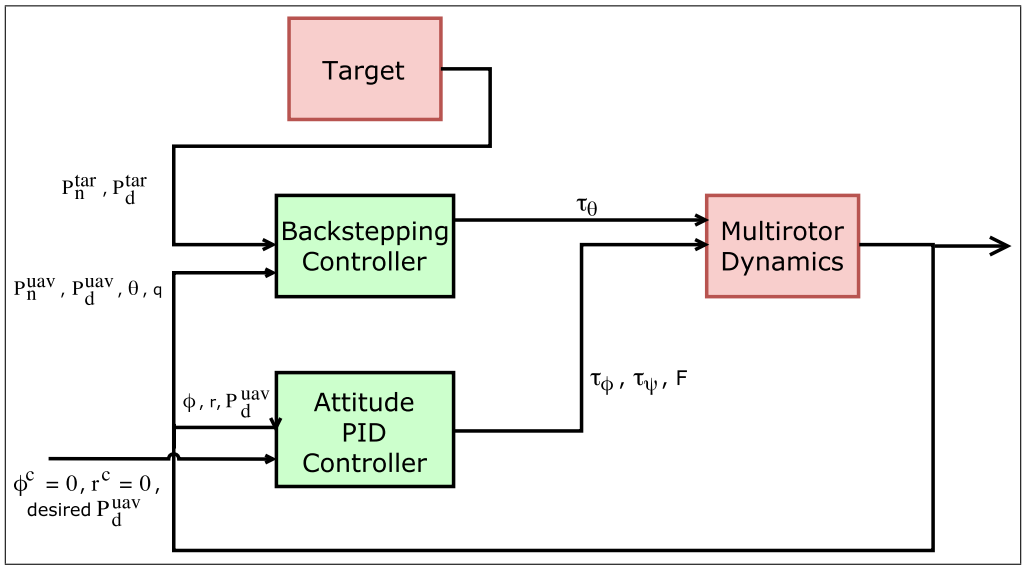


Figure 4.6: Control system diagram for the inertial line of sight vector backstepping controller. In this configuration, the backstepping controller needs the positions of multirotor and target.

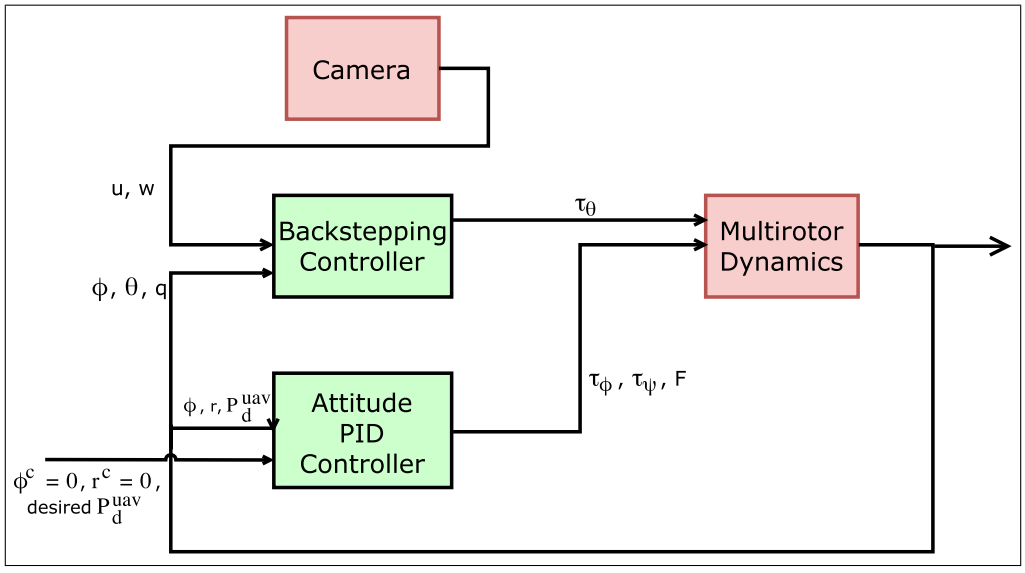
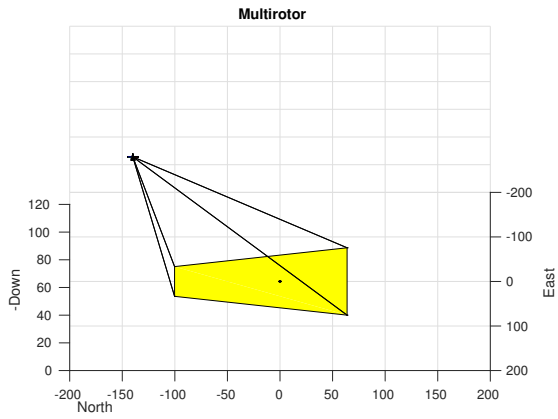
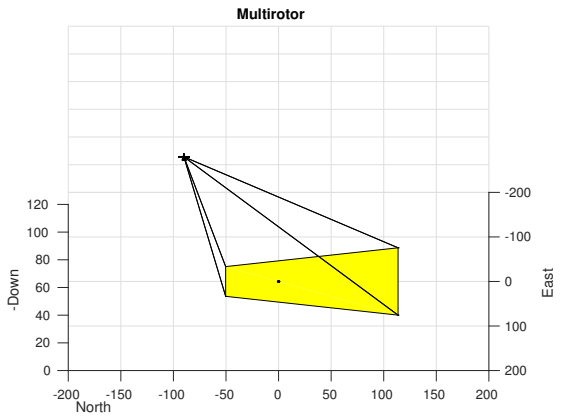


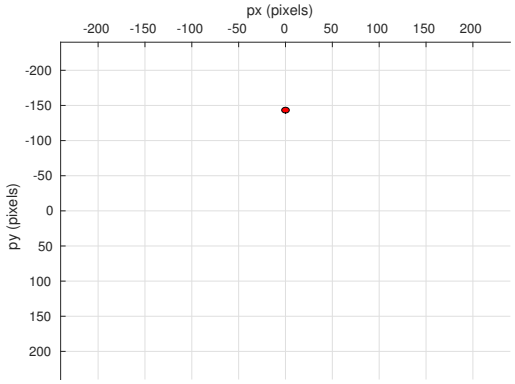
Figure 4.7: Control system diagram for the image-based backstepping controller. Note that the backstepping controller only requires the image coordinates of the target.



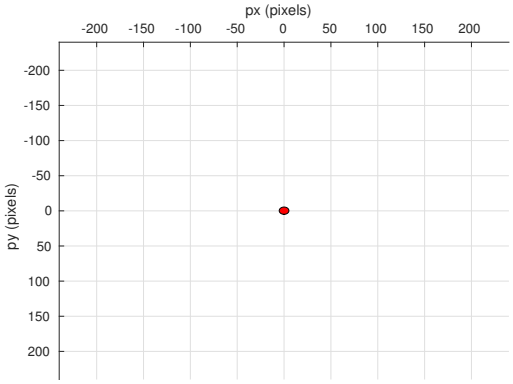
(a) Multirotor and ground target when $t = 0s$



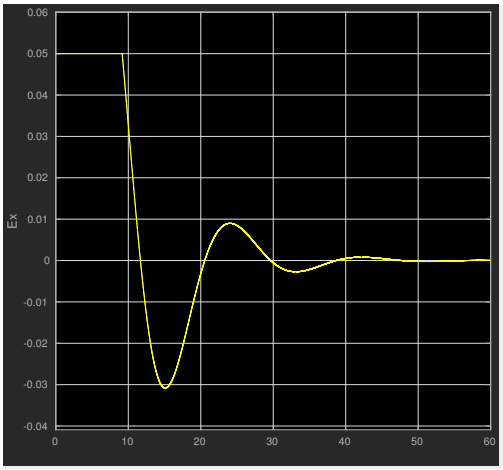
(b) when $t = 60s$



(c) Camera view when $t = 0s$

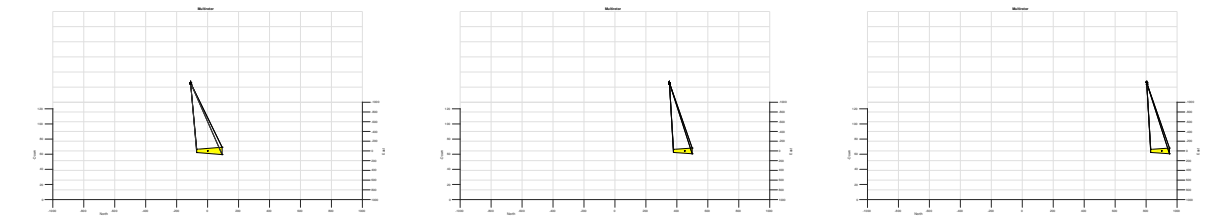


(d) when $t = 60s$

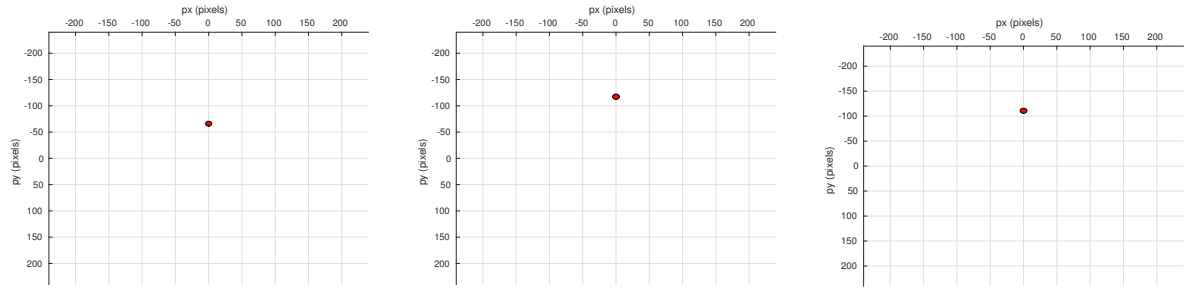


(e) The horizontal error (e_x) between the unit LOS vector and the unit optical axis vector both in the vehicle-1 frame converges to zero. It saturates at 0.05 to prevent the multirotor from commanding too large torque command.

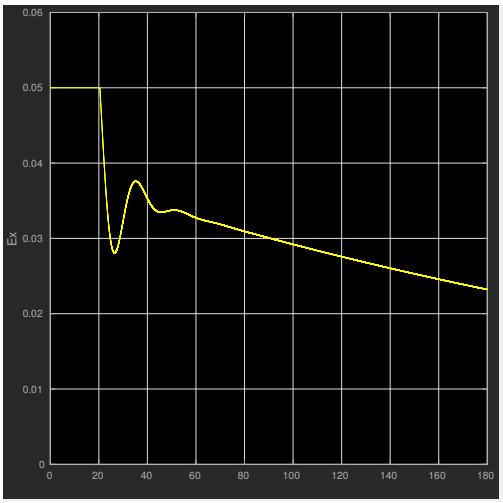
Figure 4.8: Simulation result for the backstepping control using the inertial LOS vector. The ground target is static ($0m/s$). The initial UAV and target positions are [-140, 0, -90] and [0, 0, 0] respectively. Tuning parameters are set to $k = 0.12$, $k_1 = 1$, $k_2 = 1$, $k_3 = 1$, and $\Gamma = 0.01 * I_3$ (identity matrix).



(a) when $t = 0s$ (b) when $t = 90s$ (c) when $t = 180s$

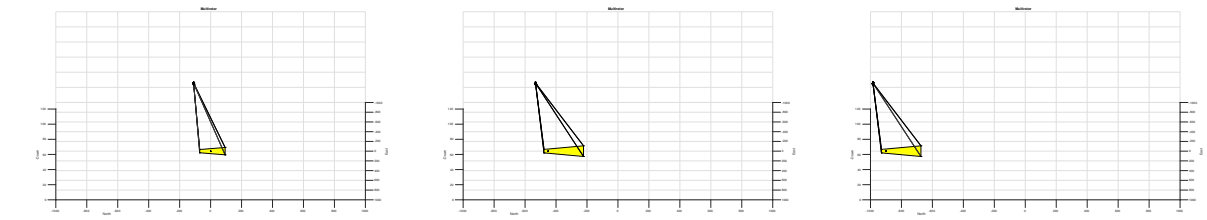


(d) when $t = 0s$ (e) when $t = 90s$ (f) when $t = 180s$

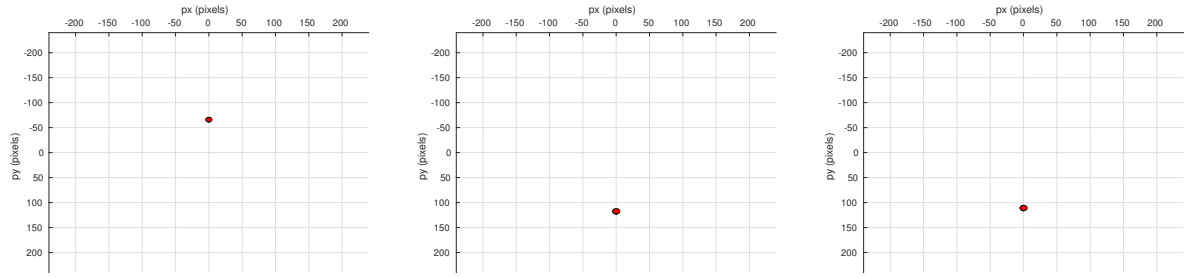


(g) The horizontal error (e_x) between the unit LOS vector and the unit optical axis vector both in the vehicle-1 frame converges to zero. It saturates at 0.05 to prevent the multirotor from commanding too large torque command.

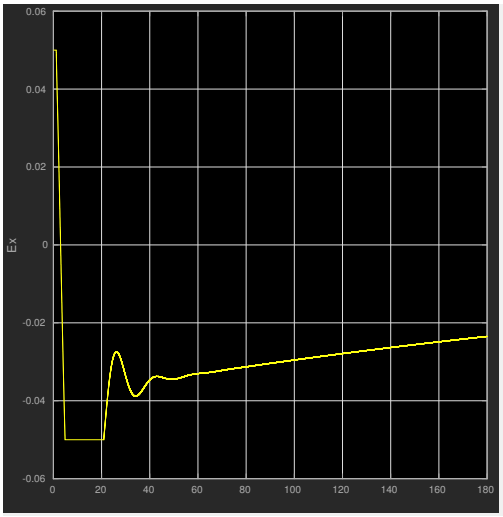
Figure 4.9: Simulation result for the backstepping control using the inertial LOS vector. The ground target is moving at the speed of $5m/s$. The initial UAV and target positions are [-110, 0, -90] and [0, 0, 0] respectively. Tuning parameters are set to $k = 0.12$, $k_1 = 1$, $k_2 = 1$, $k_3 = 1$, and $\Gamma = 0.01 * I_3$ (identity matrix). In this case, the target is not placed at the center of image because the horizontal error is computed in the vehicle-1 frame meaning that the pitch of multirotor is not compensated.



(a) when $t = 0s$ (b) when $t = 90s$ (c) when $t = 180s$



(d) when $t = 0s$ (e) when $t = 90s$ (f) when $t = 180s$

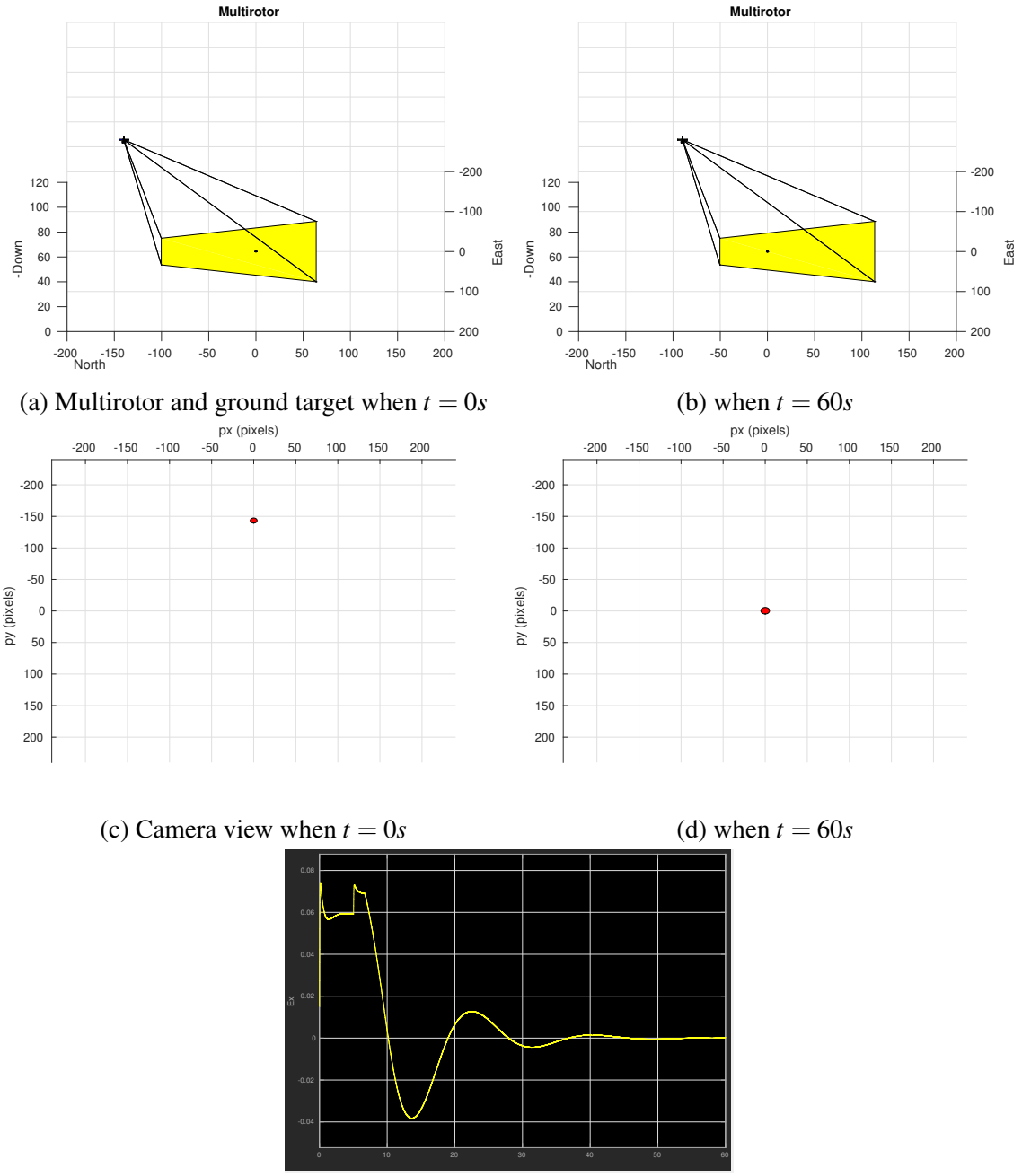


(g) The horizontal error (e_x) between the unit LOS vector and the unit optical axis vector both in the vehicle-1 frame converges to zero. It saturates at 0.05 to prevent the multirotor from commanding too large torque command.

Figure 4.10: Simulation result for the backstepping control using the inertial LOS vector. The ground target is moving at the speed of $-5m/s$. The initial UAV and target positions are [-110, 0, -90] and [0, 0, 0] respectively. Tuning parameters are set to $k = 0.12$, $k_1 = 1$, $k_2 = 1$, $k_3 = 1$, and $\Gamma = 0.01 * I_3$ (identity matrix). In this case, the target is not placed at the center of image because the horizontal error is computed in the vehicle-1 frame meaning that the pitch of multirotor is not compensated.

know where the target is relative to itself but through the information received from the camera. Thus, the second controller is only using the camera information (target pixel location) to control the multirotor (See Figure 4.7). The camera is simulated to output its information at $30Hz$ while the backstepping controller is evaluated at $100Hz$. This caused the horizontal error e_x to be quite noisy which eventually makes the controller unstable. Thus, a low-pass filter is added internally in the backstepping controller to smooth e_x. The simulation results for a static target, moving targets at $5m/s$ and $-5m/s$ can be found in Figure 4.11, 4.12, and 4.13 respectively. In Figure 4.11, the target is kept in the camera FOV and the horizontal error converges to zero. Figure 4.12 and 4.13 show not much different results from Figure 4.9 and 4.10 showing almost identical target location in the image and the horizontal error trend over time.

In conclusion, based on the comparison between the inertial LOS vector ($100Hz$ update rate) and the image LOS vector ($30Hz$ update rate) as inputs to the backstepping controller, it turns out that there is no significant performance decrease using the sparse image information compared to using the inertial LOS vector directly as long as a simple low-pass filter is implemented internally in the backstepping controller to smooth the jittery horizontal error (e_x). Thus, the proposed controller can be tested in hardware using a multirotor equipped with monocular camera.



(a) Multirotor and ground target when $t = 0s$

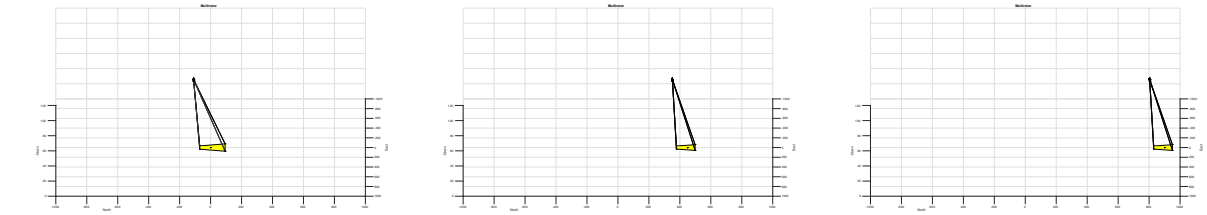
(b) when $t = 60s$

(c) Camera view when $t = 0s$

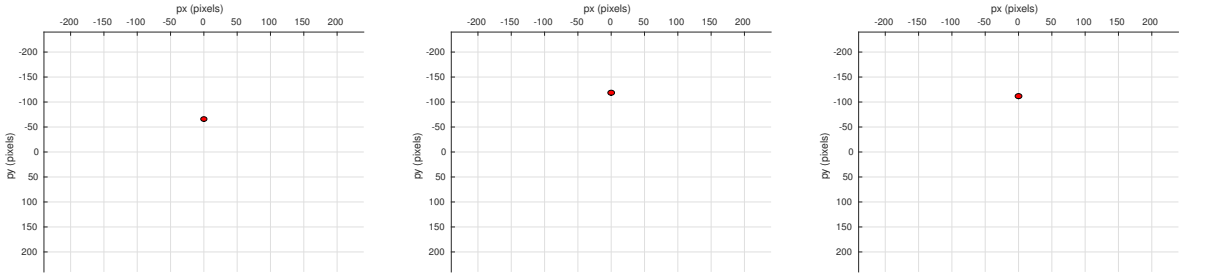
(d) when $t = 60s$

(e) The horizontal error (e_x) between the normalized target pixel coordinates and the unit optical axis vector both in the vehicle-1 frame converges to zero. Note that the value is low-pass filtered.

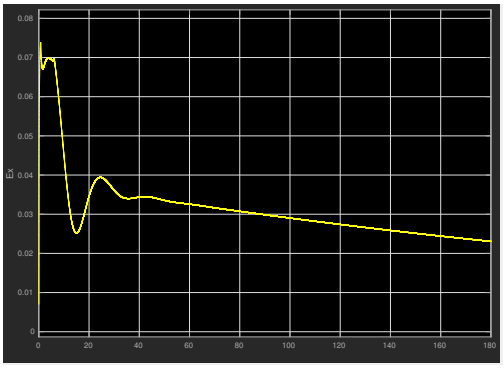
Figure 4.11: Simulation result for the backstepping control using the normalized target pixel coordinates. The ground target is static ($0m/s$). The initial UAV and target positions are [-140, 0, -90] and [0, 0, 0] respectively. Tuning parameters are set to $k = 0.12$, $k_1 = 1$, $k_2 = 1$, $k_3 = 1$, and $\Gamma = 0.01 * I_3$ (identity matrix).



(a) when $t = 0s$ (b) when $t = 90s$ (c) when $t = 180s$

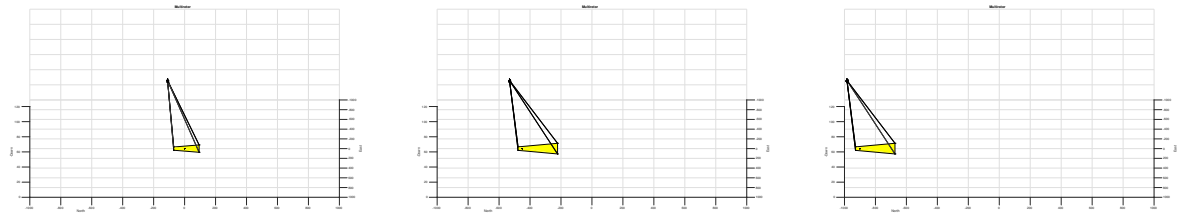


(d) when $t = 0s$ (e) when $t = 90s$ (f) when $t = 180s$

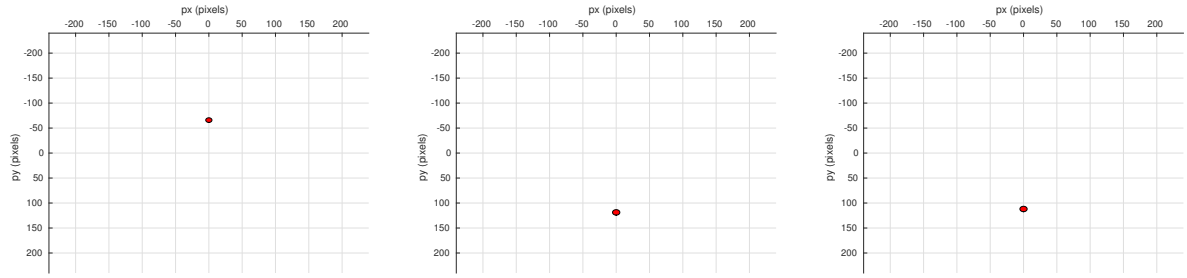


(g) The horizontal error (e_x) between the normalized target pixel coordinates and the unit optical axis vector both in the vehicle-1 frame converges to zero. Note that the value is low-pass filtered.

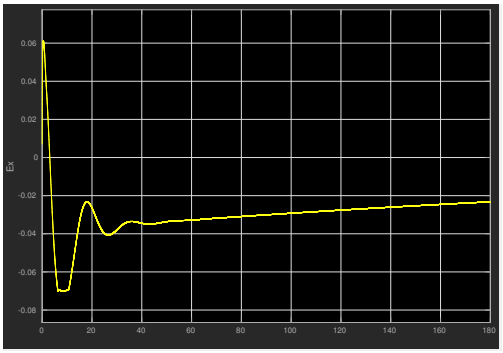
Figure 4.12: Simulation result for the backstepping control using the normalized target pixel coordinates. The ground target is moving at the speed of $5m/s$. The initial UAV and target positions are [-110, 0, -90] and [0, 0, 0] respectively. Tuning parameters are set to $k = 0.12$, $k_1 = 1$, $k_2 = 1$, $k_3 = 1$, and $\Gamma = 0.01 * I_3$ (identity matrix). In this case, the target is not placed at the center of image because the horizontal error is computed in the vehicle-1 frame meaning that the pitch of multirotor is not compensated.



(a) when $t = 0s$ (b) when $t = 90s$ (c) when $t = 180s$



(d) when $t = 0s$ (e) when $t = 90s$ (f) when $t = 180s$



(g) The horizontal error (e_x) between the normalized target pixel coordinates and the unit optical axis vector both in the vehicle-1 frame converges to zero. Note that the value is low-pass filtered.

Figure 4.13: Simulation result for the backstepping control using the normalized target pixel coordinates. The ground target is moving at the speed of $-5m/s$. The initial UAV and target positions are [-110, 0, -90] and [0, 0, 0] respectively. Tuning parameters are set to $k = 0.12$, $k_1 = 1$, $k_2 = 1$, $k_3 = 1$, and $\Gamma = 0.01 * I_3$ (identity matrix). In this case, the target is not placed at the center of image because the horizontal error is computed in the vehicle-1 frame meaning that the pitch of multirotor is not compensated.

CHAPTER 5. CONCLUSION

We have presented several gimbal control algorithms: angle commanding gimbal control, angular velocity commanding gimbal control, and adaptive depth gimbal control. Although they have their own strengths and weaknesses, the adaptive depth gimbal control is the most useful when a depth sensor is not available and the camera is on a moving platform. Thus, the algorithm is more suitable than the other algorithms on small UAV tracking application. Although a custom pan-tilt gimbal had been built, the size of the gimbal system including gimbal controller exceeded the payload capability of commonly available multirotors. For this reason, if the adaptive depth gimbal controller is to be tested on UAV, a smaller size gimbal needs to be built or search for the commercially available UAV option that gives the ability to command the angular rate of the gimbal (Not available as of December 2017).

In addition, this thesis shows the first attempt to use the visual multiple target tracking using R-RANSAC tracker to close the UAV control loop in real-time. Although the theoretical contribution is not huge, the value sits in regard of the system integration and hardware result. In this experiment, one of the shortcomings is that R-RANSAC tracker does not have good tracking continuity due to the lack of computation resource (No GPU). Often, R-RANSAC tracker assigns different ID number to the same target whenever the program is lagged by computer vision processing. This could be overcome by introducing an embedded computer with GPU such as NVIDIA Jetson TK1, TX1, or TX2. Another shortcoming is that the UAV control algorithm used in the hardware demonstration is only to follow a single target. With R-RANSAC tracker, multiple target tracking in perception domain is feasible. However, without new UAV control algorithm, real multiple target tracking system cannot be realizable. Thus, if one desires to pursue further research on 'single UAV to track multiple targets' before moving forward to 'multiple UAVs to track multiple targets', the reference to [29] may be a good starting point.

Finally, a novel UAV control algorithm for target tracking, the unit vector UAV visual servoing, has been developed. This algorithm employs an adaptive control technique to derive the unit vector framework in target tracking and employs the backstepping control technique to incorporate with common multirotor dynamics. The purpose of the algorithm is to eliminate the necessity for the UAV altitude or target depth measurements for the control algorithm. Thus, this algorithm would be more suitable than the algorithm using simple camera geometry on flat earth assumption when target is moving on non-flat surface. The unit vector UAV visual servoing is newly derived and is in early stage of its development. So far there is only Simulink simulation result and it shows that it would be worthwhile to pursue further research on this control algorithm. Since there is huge gap in Simulink simulation and actual hardware, it is recommended to test the controller on a more realistic simulator such as Gazebo. In Gazebo, not only the control algorithm can be tested, but also the whole system including the R-RANSAC tracker can be simulated. Also, it is recommended to test the robustness of the controller with different tuning parameters and various target speed, not just constant speed. Then, the tested system may be implemented and integrated in hardware.

In summary, here are some suggestions and recommendations that can be pursued in the future. The listed items may be viewed as a priority list placing the top priority on top of the list or as sequential tasks placing the thing that needs to be done before another.

- Implement the unit vector UAV visual servoing in Gazebo simulator. Test the control algorithm as well as the whole system with tracking algorithm. Tune the control parameter for more realistic multirotor and test with varying speed target. Come up with safety logic to ensure that the UAV is under control.

- Integrate the unit vector UAV visual servoing algorithm into the autonomous target following system in hardware. Test the system on non-flat ground.

- Design and build a new compact gimbal for small multirotor and test the adaptive depth gimbal control algorithm on a flying platform. Compare the result with other existing algorithms. (Optional)

- Implement the multiple target keeping in FOV algorithm from [29] and integrate into the autonomous target following system. Note that the algorithm from [29] is realistic for a multirotor equipped with at least two-axis gimbal. (Optional)

REFERENCES

[1] S. Hutchinson, G. Hager, and P. I. Corke, "A tutorial on visual servo control," *IEEE International Conference on Robotics and Automation*, vol. 12, no. 5, pp. 651–670, 1996. 1

[2] F. Chaumette and S. Hutchinson, "Visual servo control. I. Basic approaches," *IEEE Robotics and Automation Magazine*, vol. 13, no. 4, pp. 82–90, 2006. 1

[3] G. Hu, N. Gans, and W. E. Dixon, *Adaptive Visual Servo Control*. New York, NY: Springer New York, 2009, pp. 42–63. [Online]. Available: https://doi.org/10.1007/978-0-387-30440-3_3 1

[4] J. Saunders and R. W. Beard, "Visual Tracking in Wind with Field of View Constraints," *International Journal of Micro Air Vehicles*, vol. 3, no. 3, pp. 169–182, sep 2011. [Online]. Available: http://journals.sagepub.com/doi/10.1260/1756-8293.3.3.169 2

[5] R. Rysdyk, "Unmanned Aerial Vehicle Path Following for Target Observation in Wind," *Journal of Guidance, Control, and Dynamics*, vol. 29, no. 5, pp. 1092–1100, sep 2006. [Online]. Available: http://arc.aiaa.org/doi/10.2514/1.19101 2

[6] V. Dobrokhodov, I. Kaminer, K. Jones, and R. Ghabcheloo, "Vision-based tracking and motion estimation for moving targets using small UAVs," in *2006 American Control Conference*. IEEE, 2006, p. 6 pp. [Online]. Available: http://ieeexplore.ieee.org/document/1656418/ 2

[7] A. Qadir, J. Neubert, W. Semke, and R. Schultz, "On-Board Visual Tracking With Unmanned Aircraft System (UAS)," in *Infotech@Aerospace 2011*. Reston, Virigina: American Institute of Aeronautics and Astronautics, mar 2011, p. 9. [Online]. Available: http://arc.aiaa.org/doi/10.2514/6.2011-1503 2

[8] P. Theodorakopoulos and S. Lacroix, "A strategy for tracking a ground target with a UAV," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, sep 2008, pp. 1254–1259. [Online]. Available: http://ieeexplore.ieee.org/document/4650939/ 2

[9] J. Pestana, J. L. Sanchez-Lopez, P. Campoy, and S. Saripalli, "Vision based GPS-denied Object Tracking and following for unmanned aerial vehicles," in *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, oct 2013, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/document/6719359/ 2, 3

[10] J. Thomas, J. Welde, G. Loianno, K. Daniilidis, and V. Kumar, "Autonomous Flight for Detection, Localization, and Tracking of Moving Targets With a Small Quadrotor," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1762–1769, jul 2017. [Online]. Available: http://ieeexplore.ieee.org/document/7921549/ 2, 3

[11] C. Teuliere, L. Eck, and E. Marchand, "Chasing a moving target from a flying UAV," *IEEE International Conference on Intelligent Robots and Systems*, pp. 4929–4934, 2011. 2

[12] J. Kim and D. H. Shim, "A vision-based target tracking control system of a quadrotor by using a tablet computer," in *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, may 2013, pp. 1165–1172. [Online]. Available: http://ieeexplore.ieee.org/document/6564808/ 2

[13] Z. Hurak and M. Rezac, "Image-based pointing and tracking for inertially stabilized airborne camera platform," *IEEE Transactions on Control Systems Technology*, vol. 20, no. 5, pp. 1146–1159, 2012. 2, 11

[14] D. Lee, T. Ryan, and H. J. Kim, "Autonomous landing of a VTOL UAV on a moving platform using image-based visual servoing," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, may 2012, pp. 971–976. [Online]. Available: http://ieeexplore.ieee.org/document/6224828/ 2

[15] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-Learning-Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1409–1422, jul 2012. [Online]. Available: http://ieeexplore.ieee.org/document/6104061/ 3

[16] P. C. Niedfeldt and R. W. Beard, "Multiple target tracking using recursive RANSAC," in *2014 American Control Conference*. IEEE, jun 2014, pp. 3393–3398. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6859273http://ieeexplore.ieee.org/document/6859273/ 3, 29

[17] E. B. Quist, P. C. Niedfeldt, and R. W. Beard, "Radar odometry with recursive-RANSAC," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 4, pp. 1618–1630, 2016. 3

[18] J. Wikle and T. W. Mclain, "Detection and tracking of multiple intruders using the recursive-ransac algorithm state of practice," Utah Space Grant Fellowship Symposium, 2012. 3

[19] K. Ingersoll, "Vision based multiple target tracking using recursive ransac," Master's thesis, Brigham Young University, Provo, UT, 2015. 3

[20] P. Defranco, "Detecting and tracking moving objects from a small unmanned air vehicle," Master's thesis, Brigham Young University, Provo, UT, 2015. 3

[21] R. W. Beard and T. W. McLain, *Small unmanned aircraft: Theory and practice*. Princeton university press, 2012. 5

[22] Itseez, "Open source computer vision library," https://github.com/itseez/opencv, 2015. 9

[23] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modeling and control*. Wiley New York, 2006, vol. 3. 11

[24] E. Lavretsky and K. A. Wise, *State Feedback Direct Model Reference Adaptive Control*. London: Springer London, 2013, pp. 263–292. [Online]. Available: https://doi.org/10.1007/978-1-4471-4396-3_9 16, 20

[25] V. Ermakov, “mavros,” http://wiki.ros.org/mavros. 30

[26] R. Beard, “Quadrotor Dynamics and Control Rev 0.1,” 2008. 44, 50

[27] H. K. Khalil, “Noninear systems,” *Prentice-Hall, New Jersey*, vol. 2, no. 5, pp. 5–1, 1996. 48

[28] I. A. Raptis and K. P. Valavanis, *Linear and nonlinear control of small-scale unmanned helicopters*. Springer Science & Business Media, 2010, vol. 45. 48

[29] N. R. Gans, G. Hu, and K. Nagarajan, “Keeping Multiple Moving Targets in the Field,” vol. 27, no. 4, pp. 822–828, 2011. 59, 61