On Writing an MS Thesis

Second Line of Title if Necessary

Three Line Limit

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

On Writing an MS Thesis
Second Line of Title if Necessary
Three Line Limit

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

The Unmanned Aerial System (UAS) is an emerging technology in a variety of application such as surveillance, search and rescue, package delivery and target tracking. In particular, target tracking can make a good use of a UAS platform with an on-board camera because the platform can adjust its altitude in order to get the desired camera field of view (FOV) and image quality of the target. This thesis presents some existing gimbal and UAV control algorithms as well as novel algorithms developed as the extensions of the existing ones.

Keywords: thesis template, dissertation template, technical writing

ACKNOWLEDGMENTS

Students may use the acknowledgments page to express appreciation for the committee members, friends, or family who provided assistance in research, writing, or technical aspects of the dissertation, thesis, or selected project. Acknowledgments should be simple and in good taste.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

This is an example of the introduction. It's pretty simple and shows off some of the basic commands.

1.1 First Section

This part shows how you can divide things into sections.

1.1.1 First Subsection

Also into subsections.

1.1.2 Second Subsection

Which really helps organization and automatically gets added to the Table of Contents and gets linked to by the hyperref package.

1.2 Citation Example

One of the greatest parts about LaTeX is BibTeX. You can just call the \cite command and it will organize the whole references section for you as long as there is an entry in the refs.bib file (or whichever other .bib file you tell it to use; see the source for master.tex). Here is an example of citing previousworks [**?**], [**?**].

1.3 Fixed Width Figure Example

This part also shows how to include a basic figure like the ones shown in Figures 1.1 and 1.2.

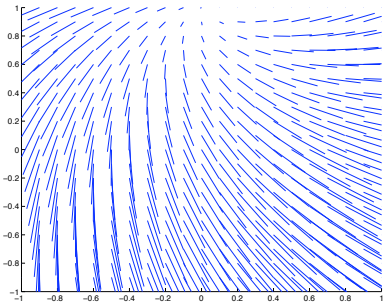


Figure 1.1: Example small width figure, showing how to use the width option.

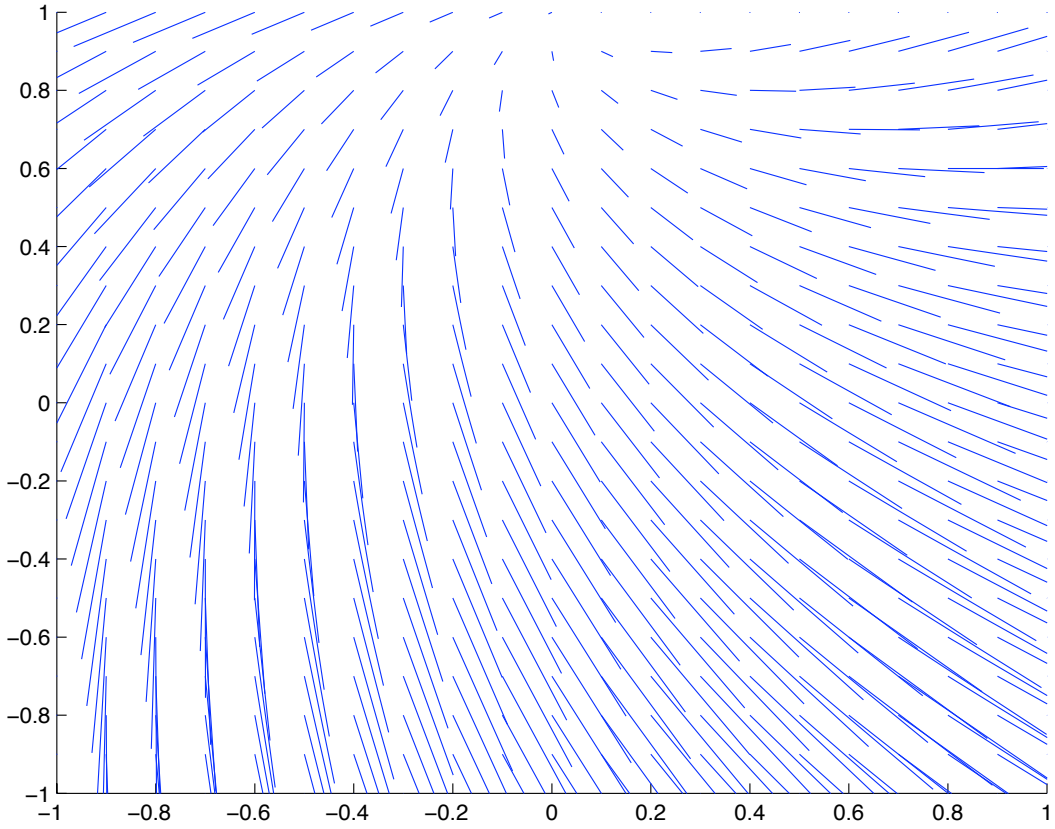


Figure 1.2: This figure is just a simple figure with a width set at 5.5 in. An example of a figure whose size depends on the width of the page is given in Figure A.1 in Section A.1 of Appendix A.

1.4 Math and Equation Example, Where the Heading Is Made so Long That It Extends onto a Second Line

Here is how to use inline math mode to define lambda like this, λ, and how to declare Equations (1.1) and (1.2)

$$I_x(x,y) = \frac{\partial I(x,y)}{\partial x}, \tag{1.1}$$

$$I_y(x,y) = \frac{\partial I(x,y)}{\partial y}. \tag{1.2}$$

Or you can create equation arrays like

$$\alpha = \beta^{\gamma} \tag{1.3}$$

$$x = \frac{1}{\alpha} \tag{1.4}$$

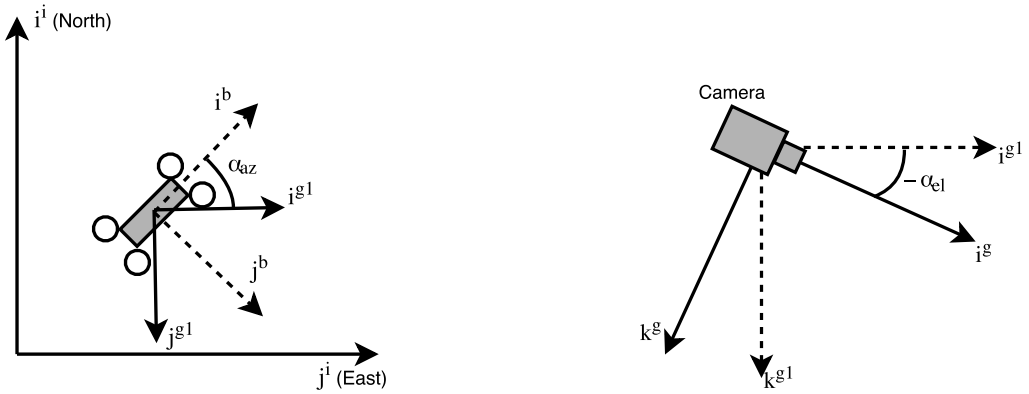
$$y = \sqrt{\left|\frac{\gamma}{\beta}\right|} \tag{1.5}$$

$$\zeta = x^{y}. \tag{1.6}$$

The lines in the array can be referenced by saying things like: In Eqn. (1.4) we show a wonderful equation, but it's not nearly as amazing as Eqn. (1.6).

CHAPTER 2. GIMBAL CONTROL

Gimbal is a widely used device for flying drones with camera because it stabilizes the camera from vibration and movement of the platform. Stabilizing camera is important in order to take high-quality, smooth videos and photos. For this reason, most of high-end commercial drones are equipped with gimbal. For autonomous systems, gimbal becomes more important because it adds maneuverability to the camera to provide with more visual awareness and to keep the object of interest in the camera field of view. There are two ways to keep the object of interest in the camera field of view: First, when we know the relative location of the target to the camera. Second, when we know the target location in image. For many practical applications, it is not feasible to assume that we know the target location in global frame. Thus, image-based gimbal control becomes more practical and realistic way to achieve the goal. In this chapter, we introduce multiple image-based gimbal control schemes.



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

(b) Side view to show the relationship between gimbal-1 frame and 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, it is worth to set up the coordinate frame convention and projective camera geometry. Note that the coordinate frame convention here is following the convention from chapter 13 of [1]. We assume that the origins of gimbal and camera frames are on the same point as the center of mass (COM) of the UAV. This is a reasonable assumption because the distance between camera, gimbal and the 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 $\mathscr{F}^b = (i^b, j^b, k^b)$, the gimbal-1 frame denoted by $\mathscr{F}^{g1} = (i^{g1}, j^{g1}, k^{g1})$, the gimbal frame denoted by $\mathscr{F}^g = (i^g, j^g, k^g)$, and the camera frame denoted by $\mathscr{F}^c = (i^c, j^c, k^c)$. The gimbal-1 frame can be obtained by rotating $\mathscr{F}^b$ about k^b axis by α_{az} which we call the gimbal azimuth angle. The gimbal frame can be obtained by rotating $\mathscr{F}^{g1}$ about j^{g1} axis by $-\alpha_{el}$ which we call the gimbal elevation angle. The negative sign is added because gimbal tilts down where tilting up is positive rotation (See Fig. 2.1). The camera frame is following the same direction as the pixel value grows in image with k^c axis pointing straight out of the camera sensor. 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

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

The last two equations from the equation 2.2 allow to use just the angle between i^{g1} and i^g without having to add the negative sign before the angle value. Combining the equations 2.1 and 2.2, we

get the rotation from the body frame $\mathscr{F}^b$ to gimbal frame $\mathscr{F}^g$

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

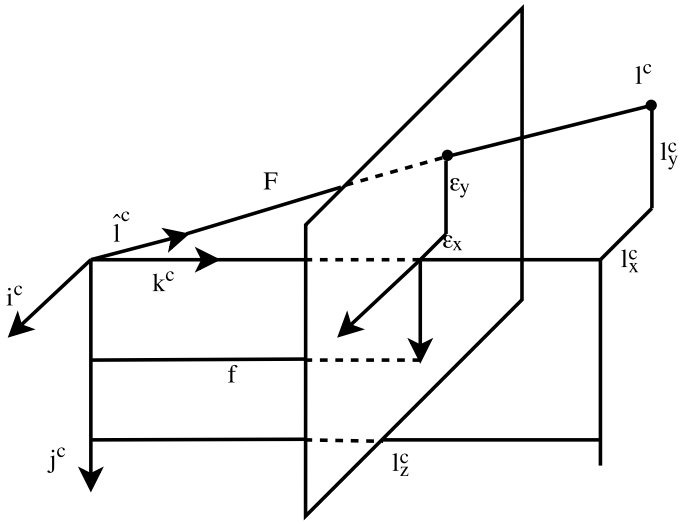


Figure 2.2: Camera frame and visual aid for projective camera model

The camera frame, often referred as optical frame, is following the common computer vision convention: i-axis grows to the right, j-axis grows to the bottom and k-axis grows to the out from the camera optical sensor (See Fig. 2.2). Thus, the rotation from the gimbal frame to the camera frame is fixed once a camera is mounted on a gimbal 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 real 3D world is projected onto a 2D image plane that is orthogonal to k^c axis with distance being the focal length f from the origin of the optical axis (See Fig. 2.2). Because of this projection, we lose the distance information to the target which has introduced wide research problems in computer vision and estimation area. The

focal length of the camera is the distance between optical sensor and lens, and it can be obtained through camera calibration. Since common camera calibration gives the focal length in the unit of pixel, the camera's field of view angle (M) can also be obtained as

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

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

2.1.2 Derivation

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

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

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 direction as shown below

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

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

$$\hat{\ell}^b = \begin{pmatrix} \hat{\ell}^b_x \\ \hat{\ell}^b_y \\ \hat{\ell}^b_z \end{pmatrix} = R^b_g(\alpha^d_{az}, \alpha^d_{el}) R^g_c \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{2.8}$$

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

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

If we solve for α^d_{az} and α^d_{el} as

$$\alpha^d_{az} = \tan^{-1}\left(\frac{\hat{\ell}^b_y}{\hat{\ell}^b_x}\right) \tag{2.11}$$

$$\alpha^d_{el} = -\sin^{-1}\left(\hat{\ell}^b_z\right), \tag{2.12}$$

they become the gimbal commands to place the object of interest at the center of image.

2.1.3 Hardware Testing Result

The angle commanding control algorithm has been implemented and tested with simple hardware (see Figure 2.3). The major hardware component includes BaseCam SimpleBGC 32-bit gimbal controller, a webcam, DYS 3-axis GoPro gimbal, and AS5048B magnetic rotary encoder attached to each rotating axis to get the exact gimbal angles. Note that in this hardware experiment, the roll axis of gimbal is always commanded to be zero to pan-tilt camera gimbal. The focal length of camera is known through the camera calibration process and the target pixel location is given by the color detection algorithm implemented using OpenCV [2]. The gimbal control algorithm is keeping the target, a line following ground robot, at the center of the camera field of view (see Figure 2.4. Also, the result video can be found at https://www.youtube.com/watch?v=OZ0Mg8AoAzk).

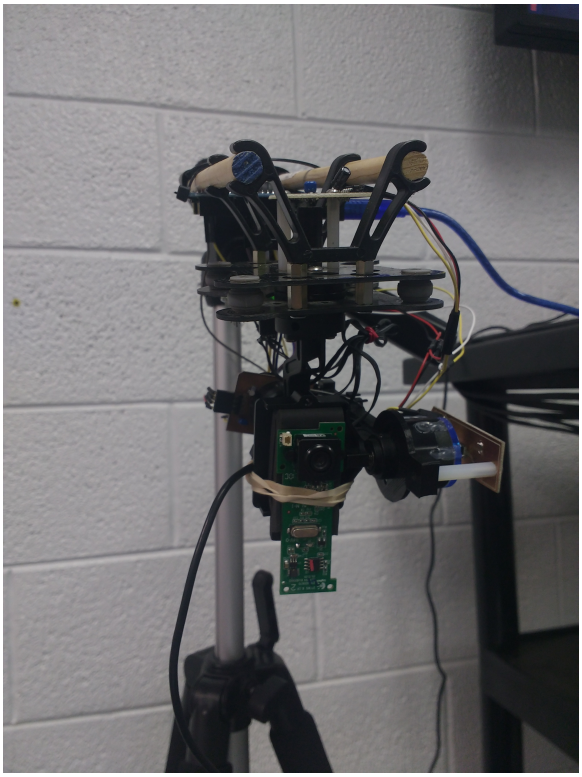


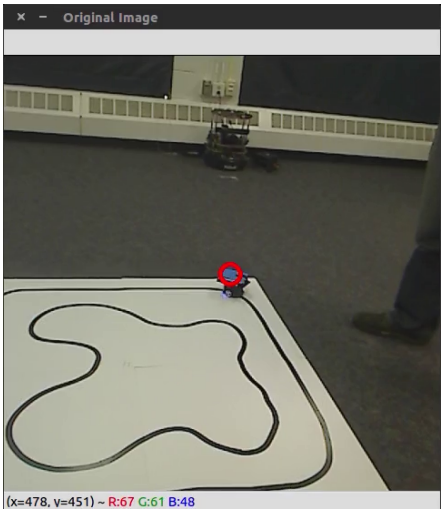
Figure 2.3: Prototype hardware to test gimbal control algorithm

2.2 Angular Velocity Commanding Gimbal Control

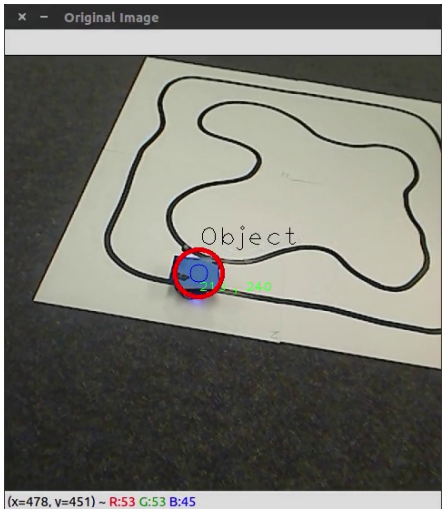
In the previous section, a relatively simple gimbal control algorithm is presented. One disadvantage of it is that it does not take the camera velocity into consideration. When gimbal is mounted on a stationary platform, the angle commanding controller can perform without any performance degradation. However, if the gimbal is mounted on UAV, taking the camera velocity into account in the controller becomes important. The work in [3] addresses this issue and derives the control 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.

2.2.1 Image Jacobian

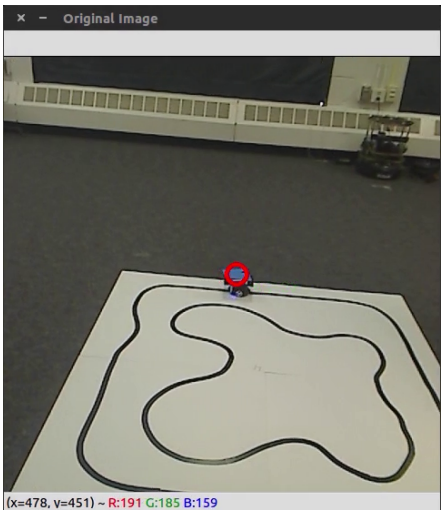
A full derivation of getting Image Jacobian matrix (often called interaction matrix) can be found in [4]. Here, we only show the final form of Image Jacobian in the process of developing the gimbal control algorithm. Let the camera translational and rotational velocities both expressed



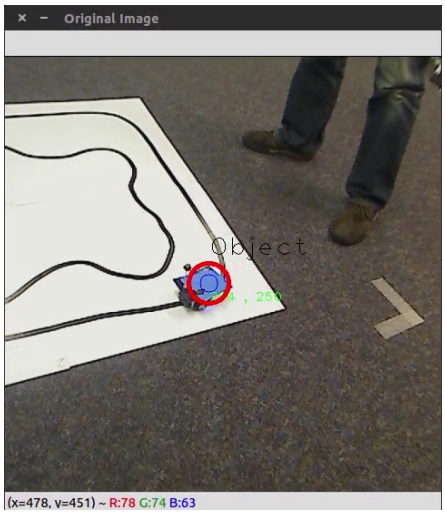
(a) when $t = 0s$



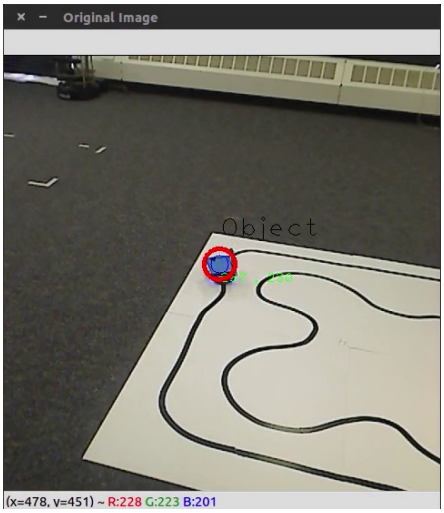
(b) when $t = 10s$



(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.

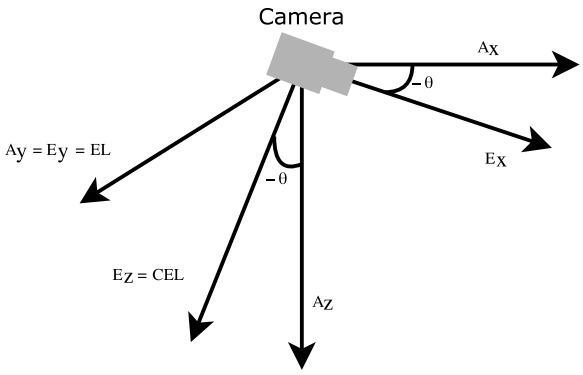


Figure 2.5: Elevation and cross-elevation axis

in the camera frame be defined as

$$\mathbf{v}_C = [v_{Cx}, v_{Cy}, v_{Cz}]^\top \tag{2.13}$$

$$\omega_C = [\omega_{Cx}, \omega_{Cy}, \omega_{Cz}]^\top \tag{2.14}$$

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

In this section, $[A]$ stands for azimuth frame, $[E]$ stands for elevation frame, and $[C]$ stands for camera frame (see Figure 2.5). θ indicates the elevation angle and ψ indicates the azimuth angle. Also, following the convention in Figure 2.2,

$$R_C^E = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \tag{2.16}$$

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

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

and

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

Image Jacobian matrix L shows 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

$$\begin{bmatrix} \dot{u} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} -\frac{\lambda}{z} & 0 & \frac{u}{z} & \frac{uw}{\lambda} & -\frac{\lambda^2+u^2}{\lambda} & w \\ 0 & -\frac{\lambda}{z} & \frac{w}{z} & \frac{\lambda^2+w^2}{\lambda} & -\frac{uw}{\lambda} & -u \end{bmatrix} \begin{bmatrix} v_{Cx} \\ v_{Cy} \\ v_{Cz} \\ \omega_{Cx} \\ \omega_{Cy} \\ \omega_{Cz} \end{bmatrix} \tag{2.20}$$

where z is the depth along the optical axis to where the target lies 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 to note that only the translational part depends on the target depth z.

2.2.2 Feedback Linearization-based Visual Pointing and Tracking

The controller's objective is to drive an image feature to the center of image. The horizontal error can be eliminated by moving two axis gimbal motors attached on azimuth and elevation axis. Moving only the azimuth axis can not compensate for the horizontal error because it only indirectly affects E_z axis which is often referred as cross-elevation axis CEL. 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 exploiting ω_{Ex} term in a clever way we can find the azimuth

and elevation commands for the same time step that smoothly compensate 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 derive (2.23) to zeros by controlling $\omega_{EL} = \omega_{Ey}$ and $\omega_{AZ} = \frac{\omega_{Ez}}{\cos\theta}$. Manipulating the equation (2.22) gives

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

Note that our interest is to find camera angular rate command only and camera translational rate can be measured when UAV is flying independent from the gimbal controller. The null space of the matrix L_ω can be parameterized as

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

and adding (2.25) to (2.24) makes

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

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

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

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

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

This angular rate reference commands expressed in the camera frame must be transformed into the elevation frame in order to make them direct command for each gimbal axis as

$$\omega_E^{ref} = R_C^E \omega_C^{ref} = \begin{bmatrix} \omega_{Ex}^{ref} \\ \omega_{Ey}^{ref} \\ \omega_{Ez}^{ref} \end{bmatrix} \tag{2.31}$$

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

It is questionable that ω_C^{ref} has commands for three axis, but there are only two controllable axis, namely ω_{Ey} and ω_{Ez}. This problem can be solved by using the free parameter k to come up with two proper motor commands. The key strategy of picking k is to select the k value that does not

require any change from the current ω_{Ex} value as

$$k = uv_y - wv_x + \frac{z(\lambda^2 + u^2 + w^2)}{\lambda}\omega_{Ex} \tag{2.33}$$

which indicates that ω_x must be measured from a sensor such as MEMS gyro attached to the camera. Substituting for k in ω_{Ey}^{ref} and ω_{Ez}^{ref} from the equation (2.32) with (2.33) gives

$$\omega_{EL}^{ref} = \omega_{Ey}^{ref} = \frac{\lambda^2 v_y - \lambda v_z w - \lambda \alpha w z + u^2 v_y - uwv_x}{z(\lambda^2 + u^2 + w^2)} + \frac{\omega_{Ex} u}{\lambda} \tag{2.34}$$

$$\omega_{CEL}^{ref} = \omega_{Ez}^{ref} = \frac{-\lambda^2 v_x + \lambda v_z u + \lambda \alpha u z + uwv_y - w^2 v_x}{z(\lambda^2 + u^2 + w^2)} + \frac{\omega_{Ex} w}{\lambda}. \tag{2.35}$$

Using the relationship

$$\omega_{AZ} = \frac{1}{\cos\theta}\omega_{CEL}, \tag{2.36}$$

we can compute the desired angular rate on azimuth axis instead of cross-elevation axis as

$$\omega_{AZ}^{ref} = \frac{1}{\cos\theta}\left(\frac{-\lambda^2 v_x + \lambda v_z u + \lambda \alpha u z + uwv_y - w^2 v_x}{z(\lambda^2 + u^2 + w^2)} + \frac{\omega_{Ex} w}{\lambda}\right). \tag{2.37}$$

The system block diagram of this controller can be found in Figure 2.6. The controller presents

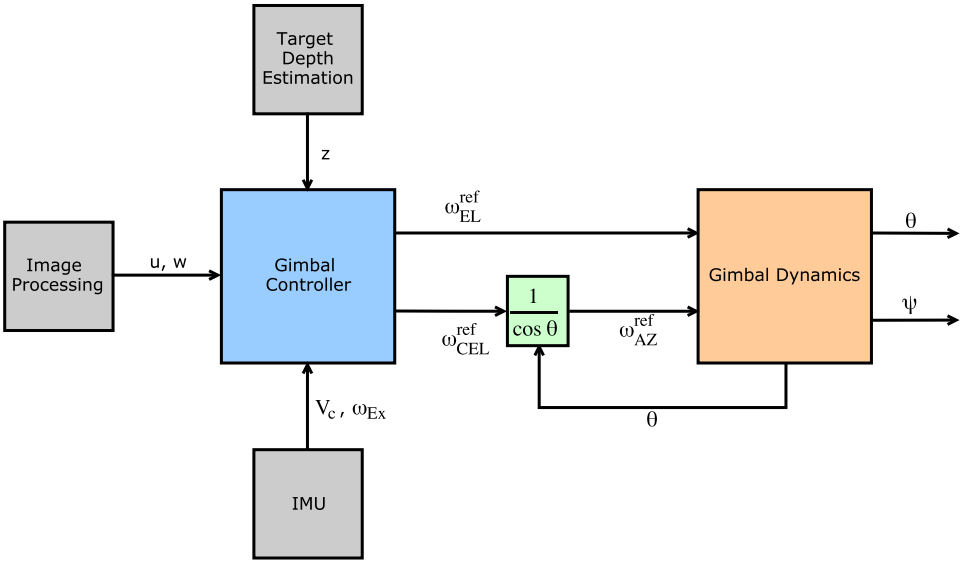


Figure 2.6: Angular velocity commanding gimbal controller block diagram

the ideal control scheme for two axis inertially stabilized gimbal system, but it also presents some 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 disturbance. Estimating the depth can be handled using a laser range finder if available or geolocation technique or adaptive depth gimbal control algorithm which is elaborated in the next section.

2.3 Adaptive Depth Gimbal Control

The gimbal control algorithm in the previous section provides with a great way to keep a target of interest in the camera's field of view with gimbal azimuth and elevation incorporating control effort rather than decoupled controls for azimuth and elevation separately. On top of the framework established in the previous section, this section examines a way to remove the necessity of externally feeding the depth, z term in the Image Jacobian equation (2.20). Instead, the controller estimates the depth online as a part of the controller while the control objective of pointing to the target is still being accomplished. This controller is inspired by a non-linear adaptive control technique called 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 [5]. In designing of MRAC system, it is required to have a reference model and we desire that the actual system behaves as similar as it can to the reference model. An adaptive law works on online parameter adaptation using the error between the reference model output and the system output, external command, and the system output. Finally, the controller computes the command for the plant using the adjusted parameters, external command, and the system output.

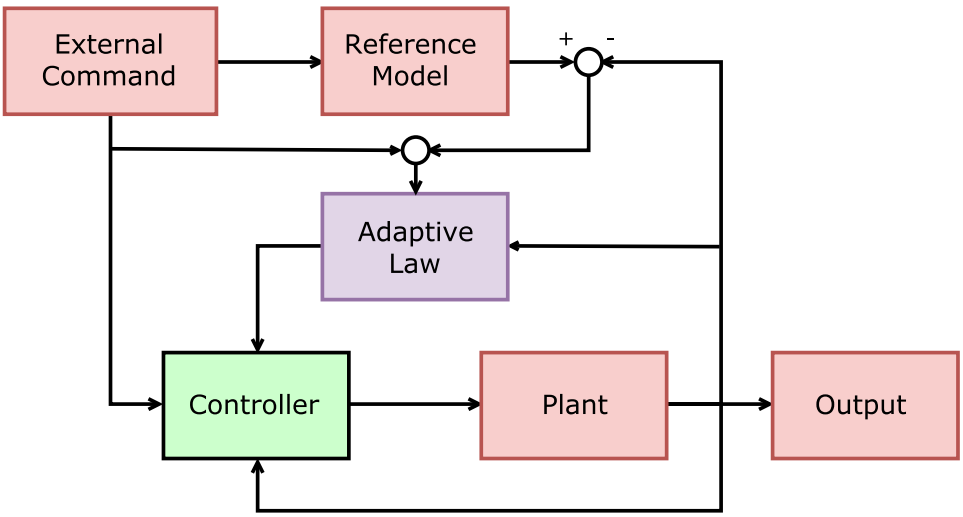


Figure 2.7: An MRAC closed-loop block diagram

2.3.2 Derivation

Let's revisit the Image Jacobian matrix

$$\begin{bmatrix} \dot{u} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} -\frac{\lambda}{z} & 0 & \frac{u}{z} & \frac{uw}{\lambda} & -\frac{\lambda^2+u^2}{\lambda} & w \\ 0 & -\frac{\lambda}{z} & \frac{w}{z} & \frac{\lambda^2+w^2}{\lambda} & -\frac{uw}{\lambda} & -u \end{bmatrix} \begin{bmatrix} v_{Cx} \\ v_{Cy} \\ v_{Cz} \\ \omega_{Cx} \\ \omega_{Cy} \\ \omega_{Cz} \end{bmatrix} \tag{2.38}$$

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 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 or 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, the equation (2.38) can be manipulated into more convenient form for MRAC design as

$$\dot{\mathbf{s}} = \frac{1}{z}\begin{bmatrix} -\lambda v_{Cx} + u v_{Cz} \\ -\lambda v_{Cy} + w v_{Cz} \end{bmatrix} + \begin{bmatrix} \frac{uw}{\lambda} & -\frac{\lambda^2+u^2}{\lambda} & w \\ \frac{\lambda^2+w^2}{\lambda} & -\frac{uw}{\lambda} & -u \end{bmatrix} \begin{bmatrix} \omega_{Cx} \\ \omega_{Cy} \\ \omega_{Cz} \end{bmatrix} \tag{2.41}$$

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

where U is the control input and β is the uncertain parameter. If it is supposed that β is known, 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 negative definite matrix that the designer chooses. Then, the subsequent dynamics would become

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

which is globally asymptotically stable system. However, since β is unknown quantity and needs to be adapted, 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 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 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

$$\dot{V} = \mathbf{e}^\top(\tilde{\beta}\varphi + A\mathbf{e}) - \frac{1}{\gamma_\beta}\tilde{\beta}\dot{\hat{\beta}} \tag{2.55}$$

$$= \mathbf{e}^\top A\mathbf{e} + \tilde{\beta}(\mathbf{e}^\top\varphi - \frac{\dot{\hat{\beta}}}{\gamma_\beta}). \tag{2.56}$$

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

$$\dot{\hat{\beta}} = \gamma_\beta\mathbf{e}^\top\varphi. \tag{2.57}$$

Then, the equation (2.54) becomes

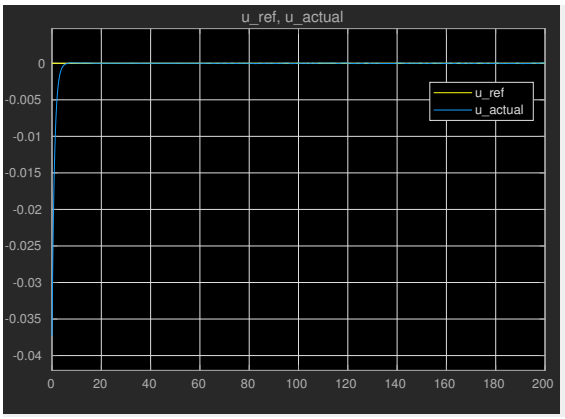
$$\dot{V} = \mathbf{e}^\top A\mathbf{e}. \tag{2.58}$$

Recalling that A is Hurwitz, the asymptotic stability of the system is proved.

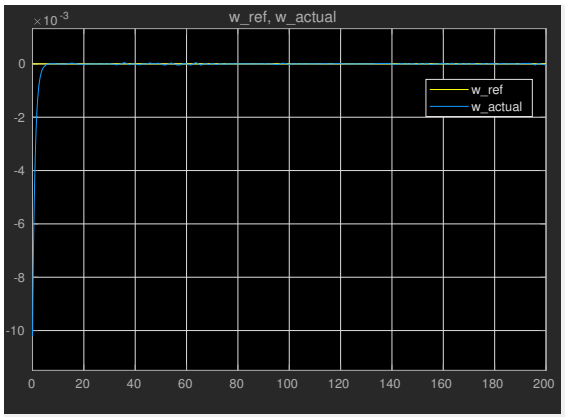
2.3.3 Simulation

First, the behavior of reference model is set as the equation (2.40) with starting image coordinates at $[0,0]$ meaning that the reference model is $[0,0]$ 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 degree and one pixel is set to one millimeter. Thus, we can obtain the focal length and image width in meter metric. 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 varying quantity because of the movement of the camera on UAV and the target. However, in this simulation z is initialized to any value between 30 and 300, and is changing with increment either 2 or -2 every second. In order to simulate the translational velocity of the UAV, v_{Cx} and v_{Cy} are initialized to zeros and changing with increment either 2 or -2 every other seconds. v_{Cz} 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 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 the observation of Figure 2.9, the estimation of uncertain parameters seem to be guaranteed to converge to the true value. However, note that this is rather special case than normal case. In normal cases, only uniform boundness of the uncertain parameters is guaranteed [5].

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 of $30Hz$ frame rate and its goal is to push the target to the center of 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. This is rather typical case of MRAC, meeting the control objective without guarantee for uncertain parameter converging to its true value.

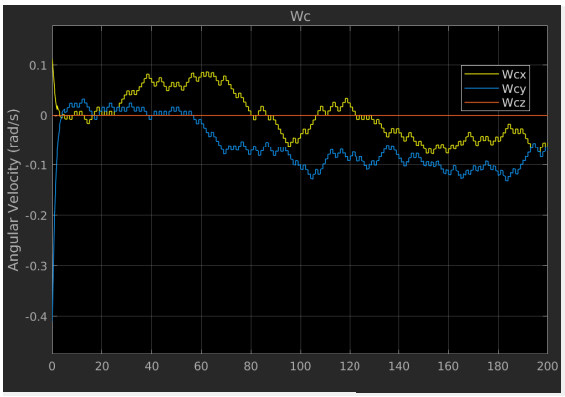


(a) Reference image coordinate u_{ref} and the system output u

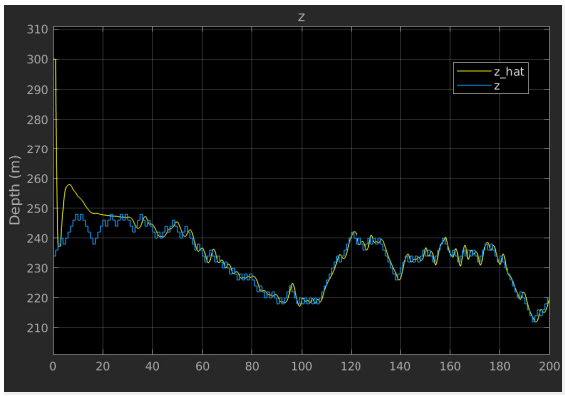


(b) Reference image coordinate w_{ref} and the system output w

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

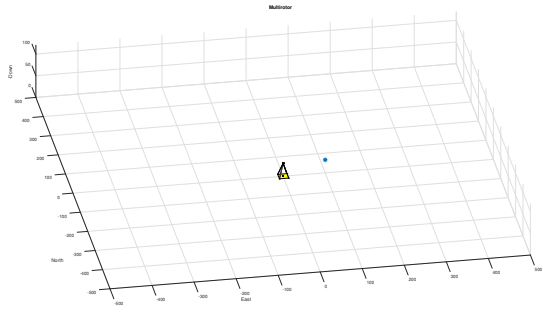


(b) Online depth estimation, $\hat{z}$

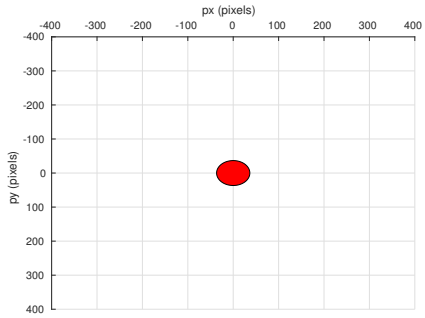
Figure 2.9: Angular velocity commands of the adaptive depth gimbal controller. Note that only two commands are actively used as if it is a pan-tilt gimbal. The depth z estimation using the adaptive law of the MRAC.

2.3.4 Hardware

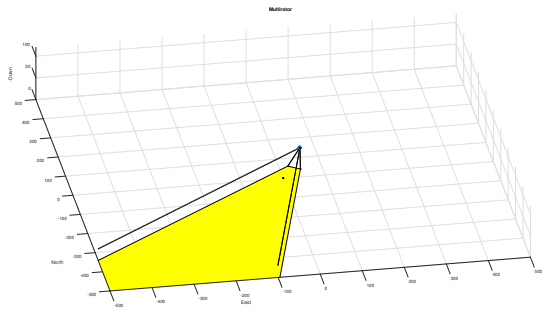
The adaptive depth gimbal controller is tested with custom pan-tilt gimbal shown in Figure 2.12. The gimbal consists of uEye UI-3250-LE-C-HQ camera, BaseCam SimpleBGC 32-bit gimbal controller, CUI AMT203 ABS SPI encoder, Quanum 4008 precision brushless gimbal motor, and 3D printed housing (See Figure 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



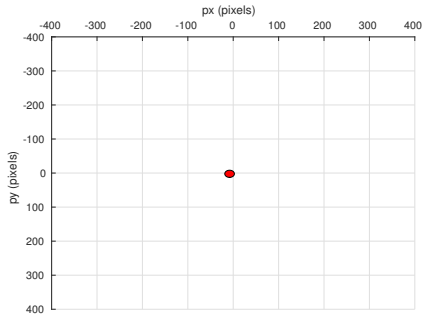
(a) Multirotor at $t = 0s$. The blue star indicates the desired multirotor position for the position PID controller for moving camera platform.



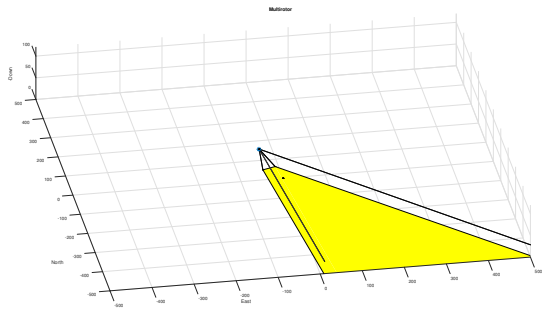
(b) Camera view at $t = 0s$



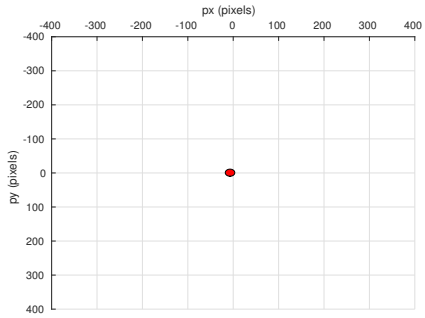
(c) Multirotor at $t = 60s$



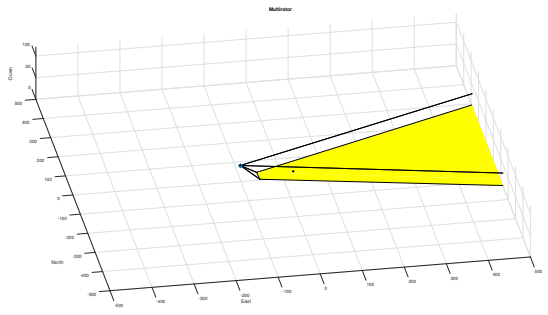
(d) Camera view at $t = 60s$



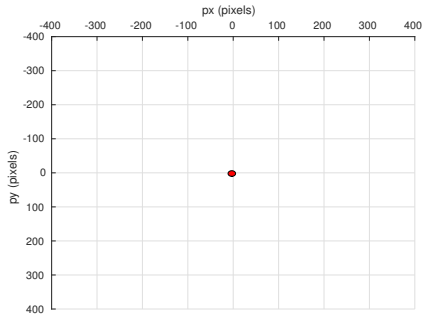
(e) Multirotor at $t = 120s$



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

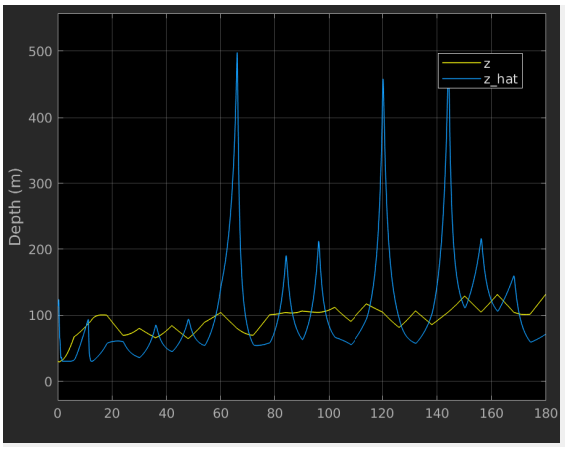


(g) Multirotor at $t = 180s$

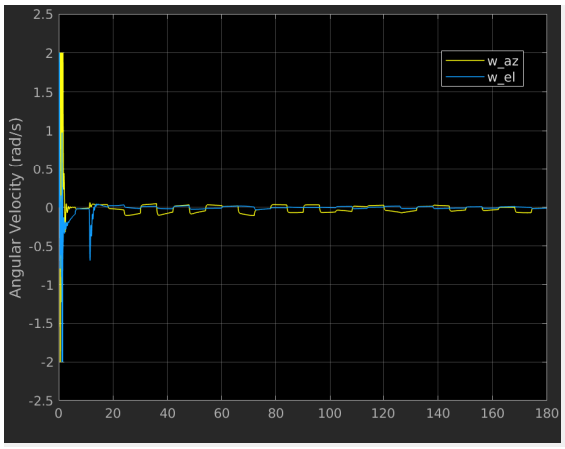


(h) Camera view at $t = 180s$

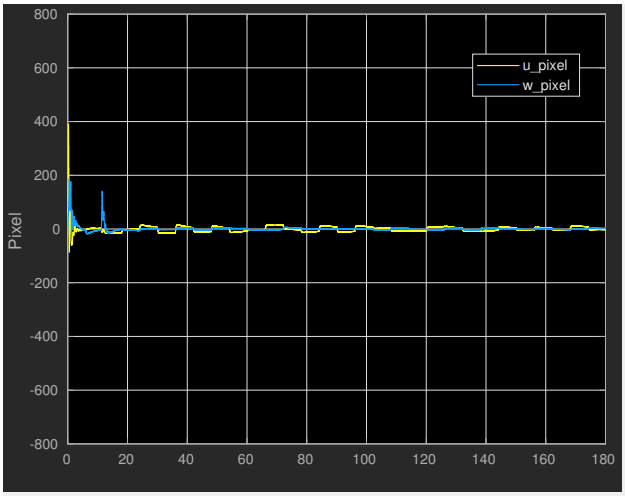
Figure 2.10: Multirotor simulation with camera view. The gimbal pointing objective is well achieved.



(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 well kept 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.

angle from the encoder and pass them to onboard computer. Also, it passes the angular velocity commands from the onboard computer to the gimbal controller. A tracking algorithm running on the onboard computer receives 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 can be found 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 zeros.

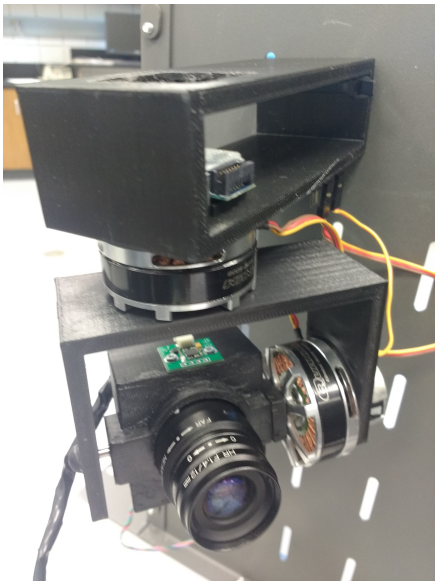


Figure 2.12: A custom pan-tilt camera gimbal

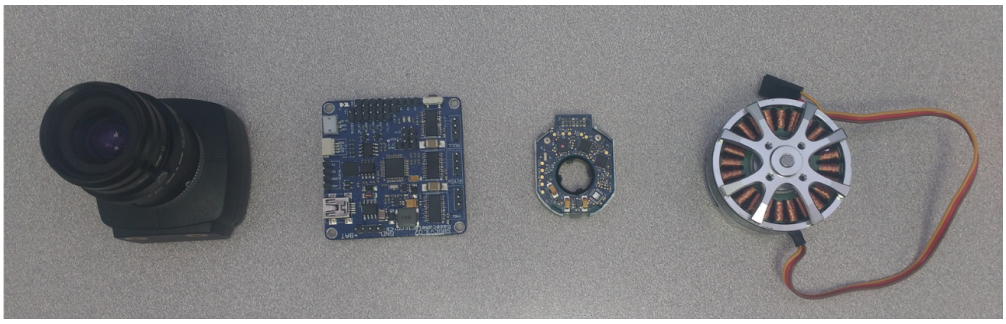


Figure 2.13: A custom pan-tilt camera gimbal

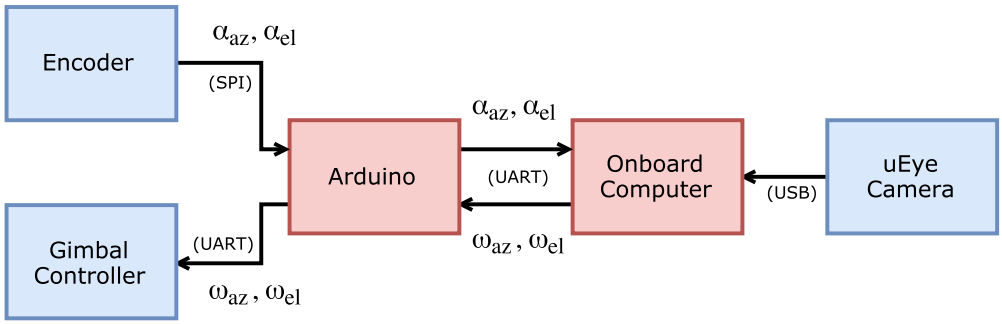
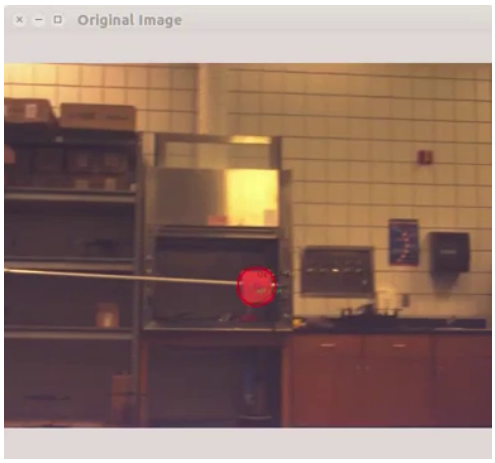
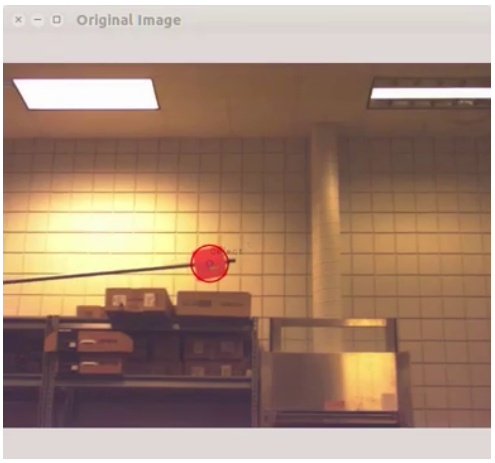


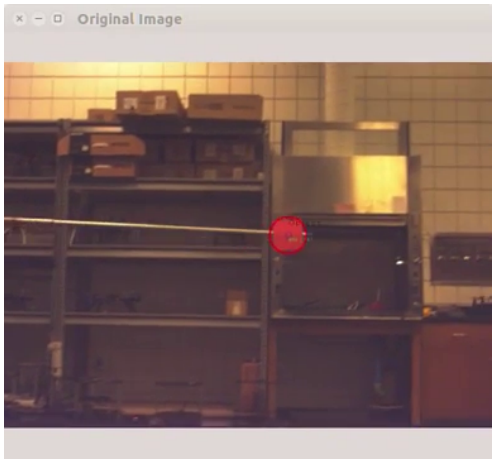
Figure 2.14: Custom gimbal block diagram



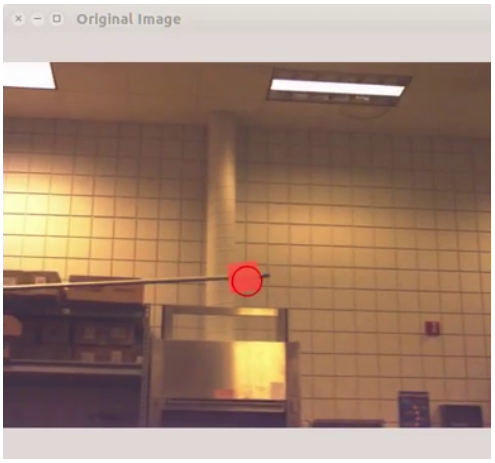
(a) At $t = 0s$



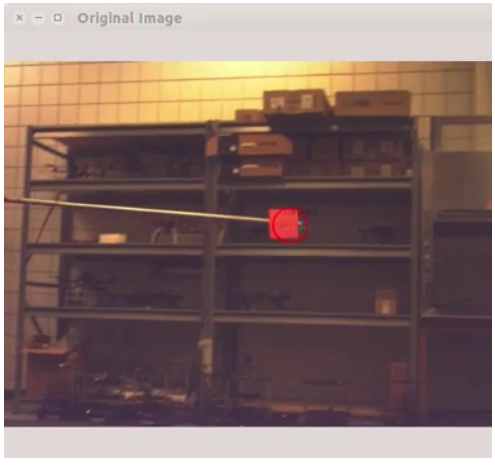
(b) At $t = 10s$



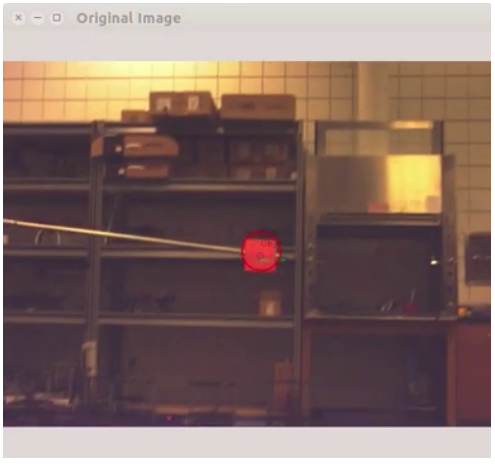
(c) At $t = 20s$



(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 is a few controller options when it is desired to keep the target in the camera field of view using 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 easy to implement and does not require the depth information. The AVCGC is expected to perform better than ACGC on a moving camera since the AVCGC take the moving platform into account. However, the AVCGC requires to know the depth z term externally. When there is no way to measure or estimate the depth z term reliably, 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 to know the depth z	Designed for moving camera Works without knowing z

Table 2.1: Summary of gimbal control schemes

CHAPTER 3. UAV CONTROL

3.1 UAV visual-servoing controller using projective camera geometry

This section introduces a relatively simple UAV control algorithm based on the projective camera geometry. We assume that the controller receives where target is in normalized image coordinate and camera is rigidly fixed to UAV platform.

3.1.1 Coordinate Frame Convention

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

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 for hardware experiment. 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 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 have access to the correct altitude information.

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

Fourth, we rely on GPS position controller on autopilot. The visual-servoing controller in this paper computes the desired multirotor position in order to follow the target and send the position command to the autopilot.

3.1.2 Forward and heading motion control

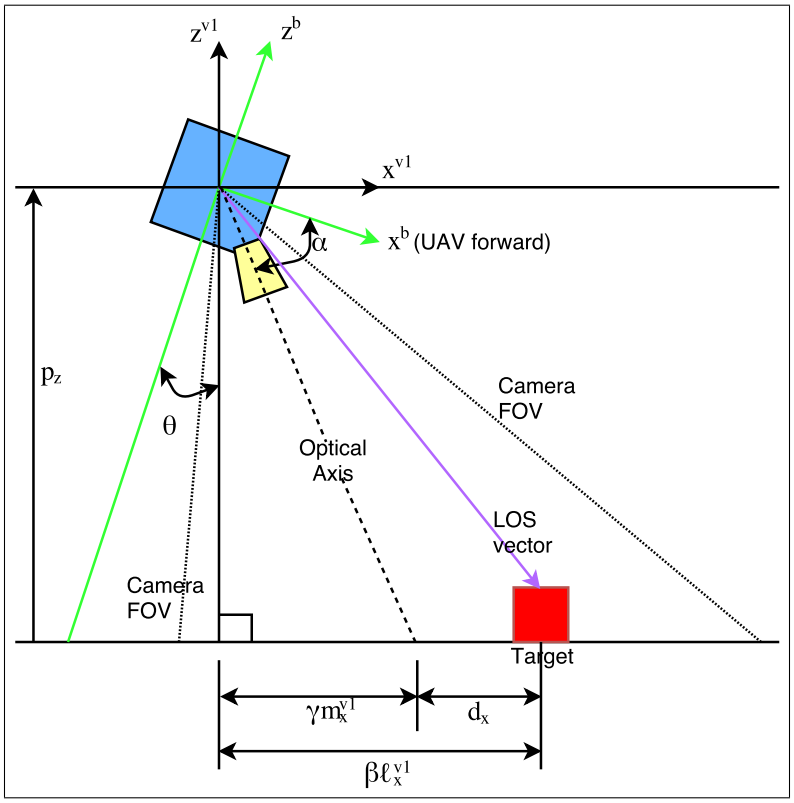


Figure 3.1: Side view of the multirotor.

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

$$\ell^o = [\varepsilon_x^o, \varepsilon_y^o, 1]^\top \tag{3.1}$$

where ℓ^o is the normalized line of sight vector in $\mathscr{F}^o$ and its third element 1 indicates the focal length of the camera in the normalized image. Let also the unit vector along the optical axis in $\mathscr{F}^o$ be defined as

$$\mathbf{m}^o = [0, 0, 1]^\top. \tag{3.2}$$

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

$$\ell^{v1} = R_b^{v1}(\phi,\theta)R_c^b(\alpha)R_o^c\ell^o = [\ell_x^{v1},\ell_y^{v1},\ell_z^{v1}]^\top \tag{3.3}$$

$$\mathbf{m}^{v1} = R_b^{v1}(\phi,\theta)R_c^b(\alpha)R_o^c\mathbf{m}^o = [m_x^{v1},m_y^{v1},m_z^{v1}]^\top \tag{3.4}$$

where R_c^b is a matrix with fixed values depending on how the camera is mounted with respect to $\mathscr{F}^b$, R_b^{v1} is a matrix requiring the roll and pitch angles of the multirotor. The 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

$$\beta = \frac{p_z}{\ell_z^{v1}} \tag{3.5}$$

$$\gamma = \frac{p_z}{m_z^{v1}} \tag{3.6}$$

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

$$d_x = \beta\ell_x^{v1} - \gamma m_x^{v1}. \tag{3.7}$$

This d_x may be further broken down into east and north components

$$d_n = d_x\sin(\psi) \tag{3.8}$$

$$d_e = d_x\cos(\psi) \tag{3.9}$$

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

$$\omega_z = \eta\ell_y^{v1}, \tag{3.10}$$

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

CHAPTER 4. ADVANCED UAV CONTROL

4.1 Motivation

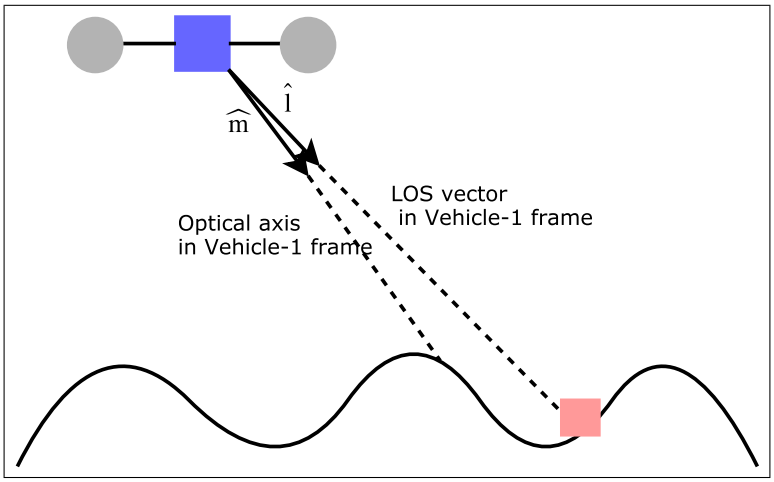


Figure 4.1: Non-flat-earth model example. The unit optical axis vector $\hat{m}$ and the unit line of sight vector $\hat{l}$ are key components of the controller presented in this chapter.

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 Controller Derivation for Simple UAV Dynamics

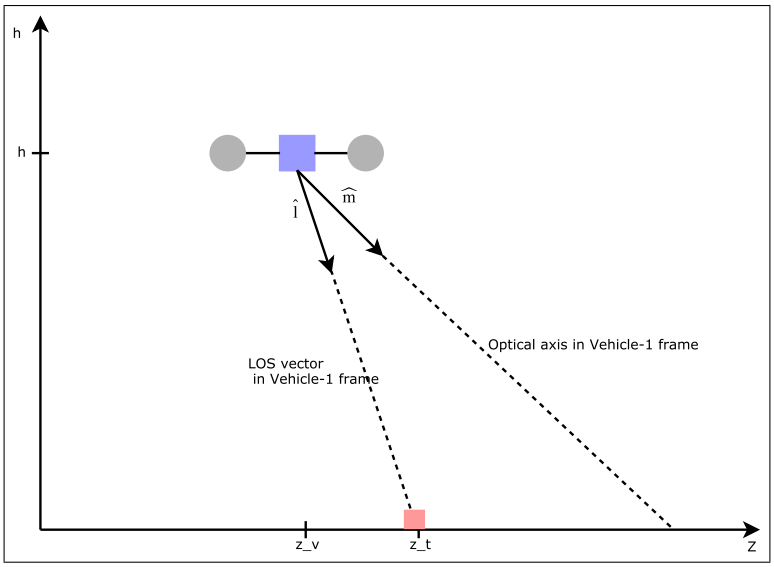


Figure 4.2: Graphical overview of the problem

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 $\hat{l}$. 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 $P_{\hat{m}}\hat{l}$ to zero. If we let

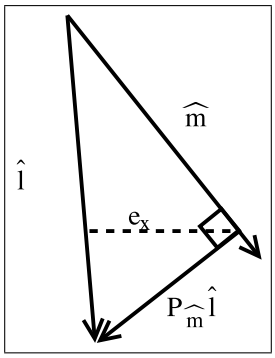


Figure 4.3: Projection onto the null space of the optical axis unit vector

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

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

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

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

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

is measurable. Meanwhile, the line of sight vector is

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

Differentiating equation (4.10) gives

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

and

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

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

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

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

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

Let

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

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

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

Differentiating equation (4.16) gives

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

Differentiating again gives

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

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

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

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

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

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

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

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

Substituting (4.14) for ${\hat{e}_1}^\top P_{\hat{m}} \ddot{l}$ results in

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

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

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

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

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

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

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

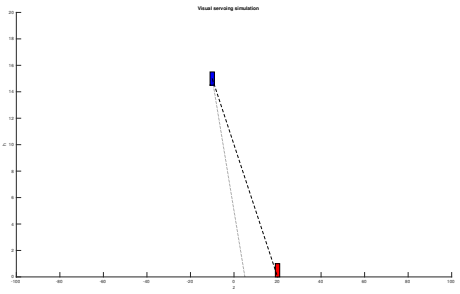
Assuming B is roughly constant or slowly varying (i.e. $\dot{B} = 0$), we have that $\dot{\tilde{B}} = -\dot{\hat{B}}$ implying that

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

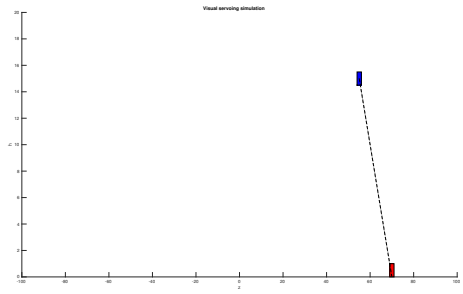
Therefore, by choosing $\dot{\hat{B}} = s\Gamma\Phi$, gives

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

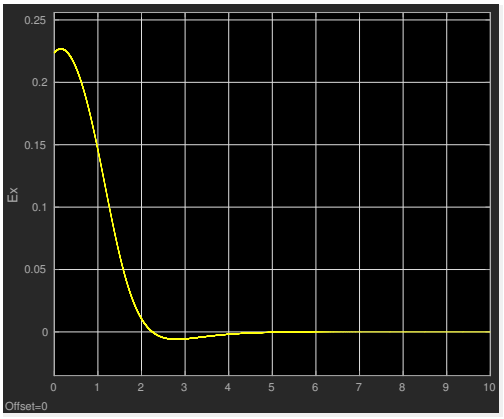
which implies that s is asymptotically stable. Thus, the original objective of stabilizing e_x is achieved. This control scheme is simulated in Simulink and Figures 4.4 and 4.5 show the performance of the controller for one initial condition.



(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

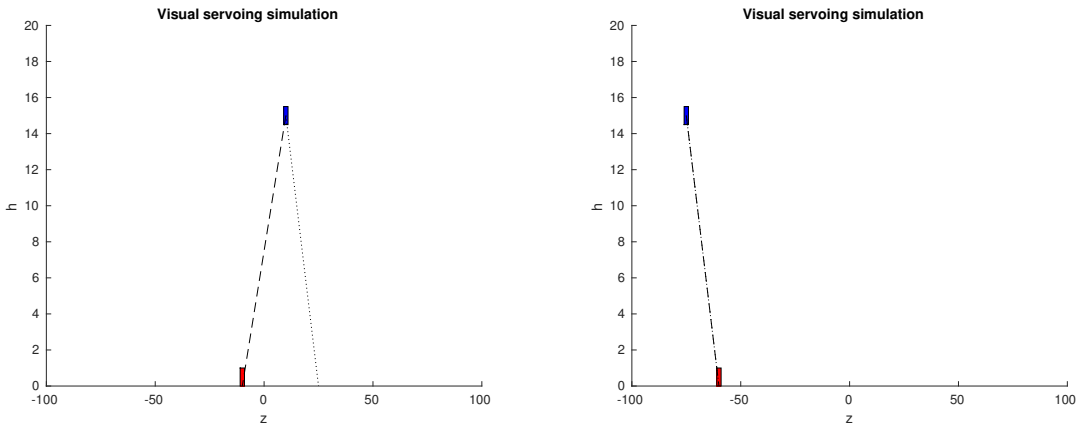
4.3.1 Derivation

The multirotor dynamics expressed in vehicle-1 frame [6] is given by

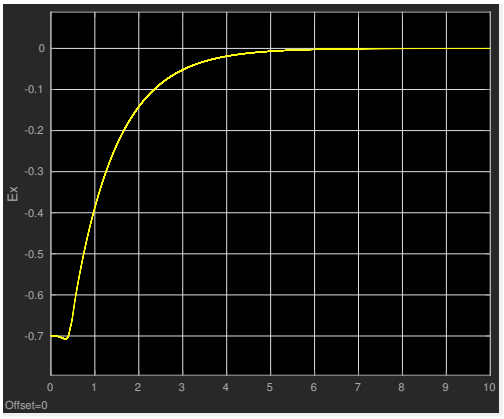
$$\ddot{p}_x = -\cos\phi \sin\theta \frac{F}{m} \tag{4.46}$$

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

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



(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

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

Differentiating (4.53) we get

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

and

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

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

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

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

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

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

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

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

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

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 [7], [8]. 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 [6]. 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. 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 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.

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 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 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. It turns out that there is no significant performance decrease using the sparse image information compared to using the inertial LOS vector directly.

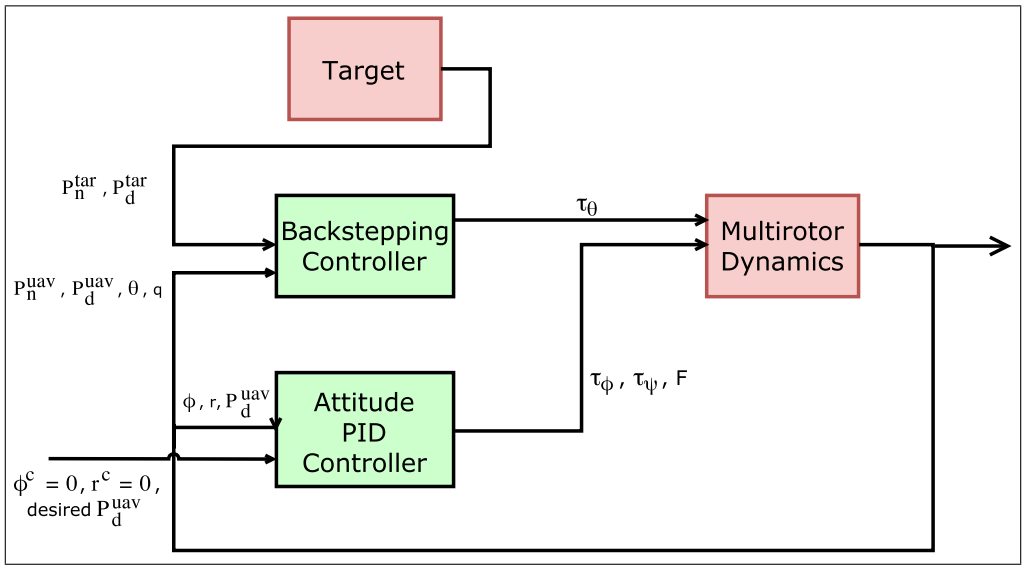


Figure 4.6: Control system diagram for the inertial line of sight vector backstepping controller. In this configuration, the backstepping controller needs the positions of multirotor and target.

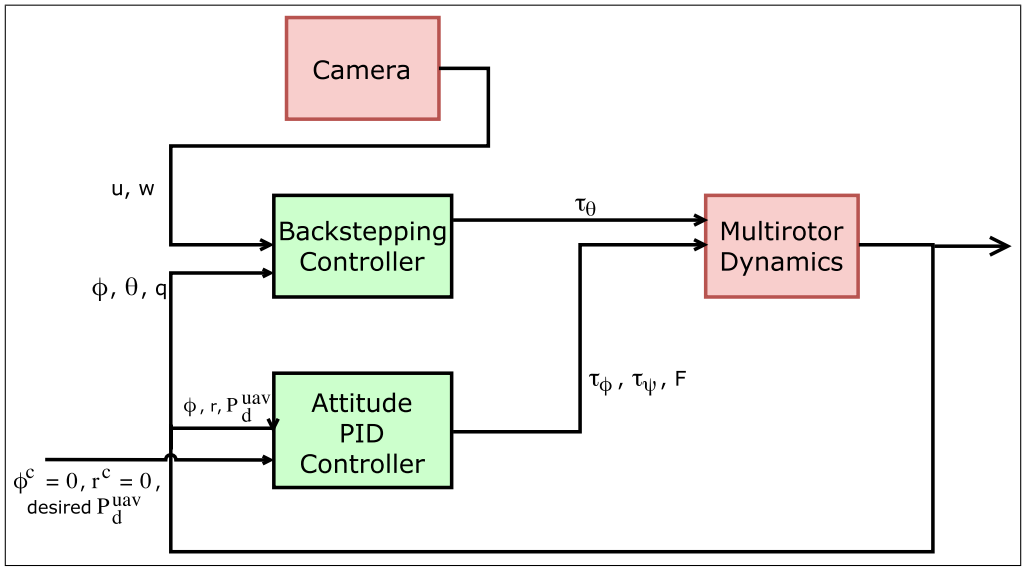
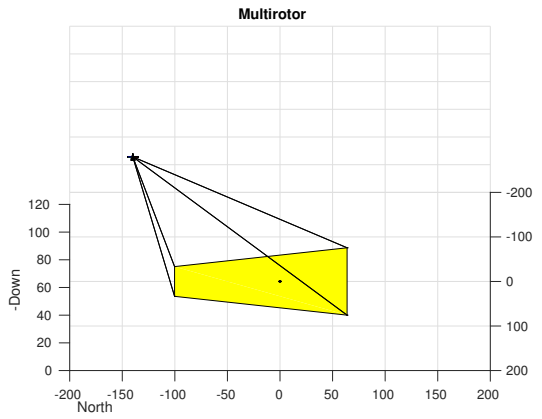
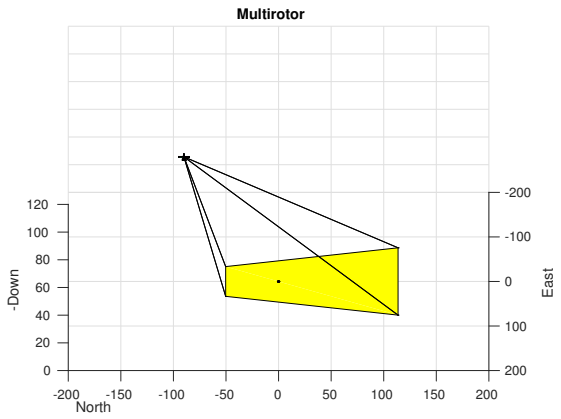


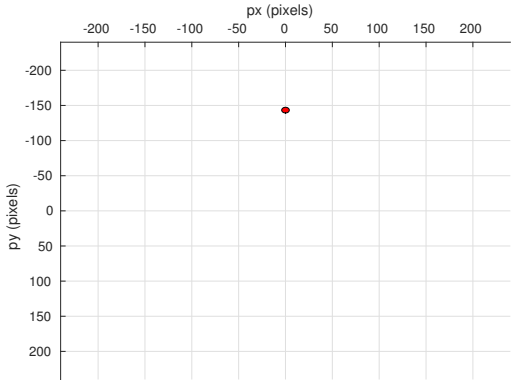
Figure 4.7: Control system diagram for the image-based backstepping controller. Note that the backstepping controller only requires the image coordinates of the target.



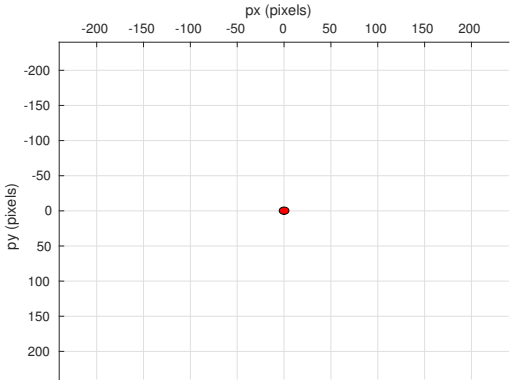
(a) Multirotor and ground target when $t = 0s$



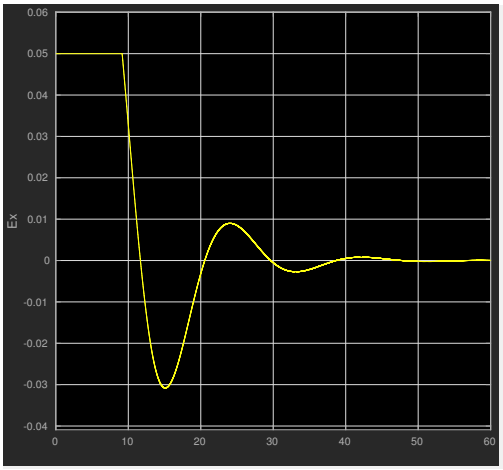
(b) when $t = 60s$



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

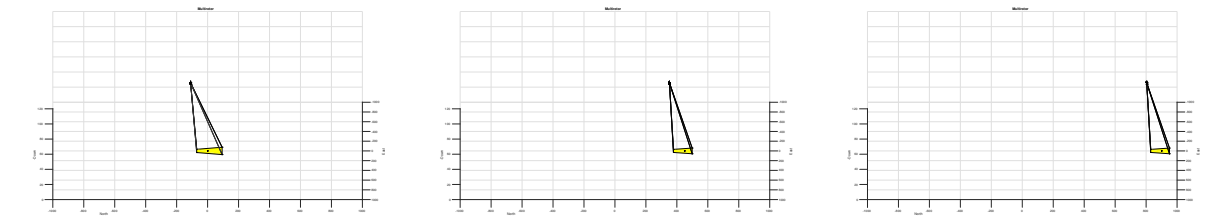


(d) when $t = 60s$

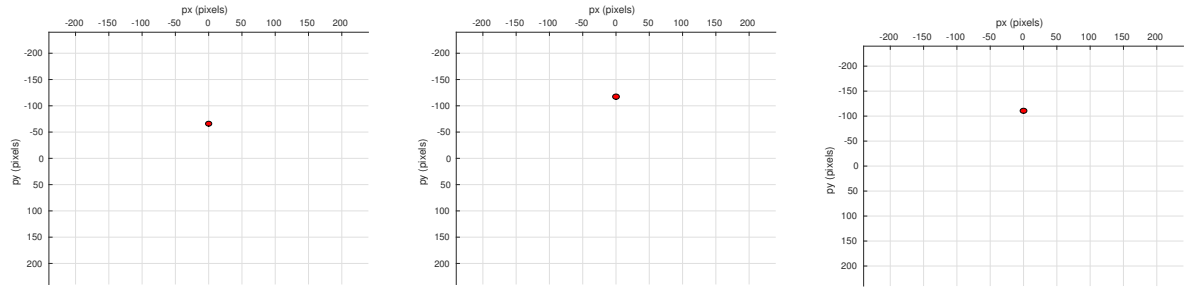


(e) The horizontal error (e_x) between the unit LOS vector and the unit optical axis vector both in the vehicle-1 frame converges to zero. It saturates at 0.05 to prevent the multirotor from commanding too large torque command.

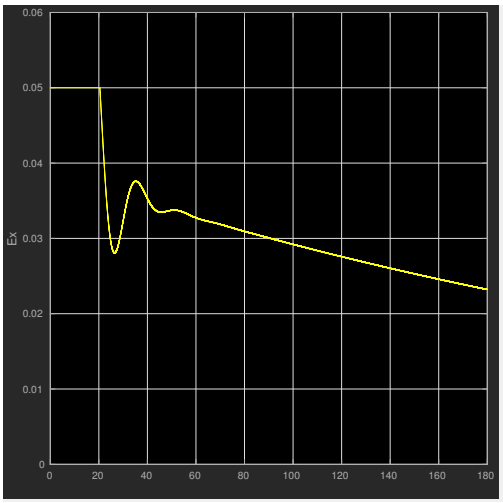
Figure 4.8: Simulation result for the backstepping control using the inertial LOS vector. The ground target is static ($0m/s$). The initial UAV and target positions are [-140, 0, -90] and [0, 0, 0] respectively. Tuning parameters are set to $k = 0.12$, $k_1 = 1$, $k_2 = 1$, $k_3 = 1$, and $\Gamma = 0.01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



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

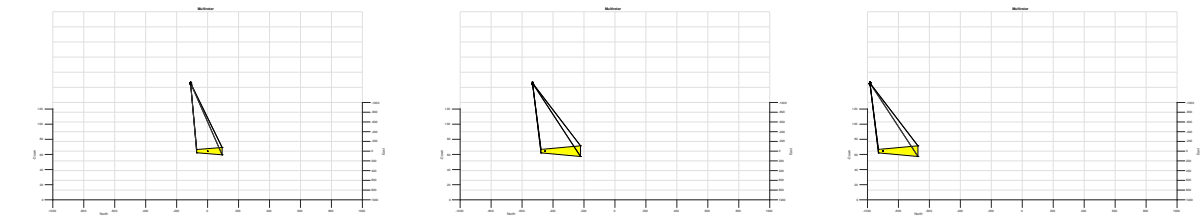


(d) when $t = 0s$ (e) when $t = 90s$ (f) when $t = 180s$

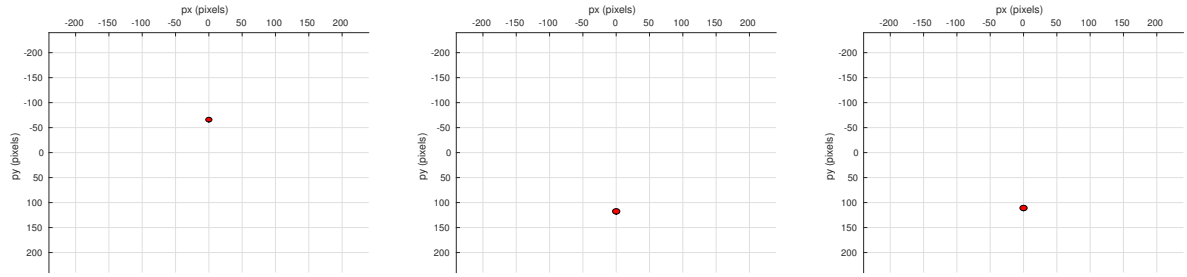


(g) The horizontal error (e_x) between the unit LOS vector and the unit optical axis vector both in the vehicle-1 frame converges to zero. It saturates at 0.05 to prevent the multirotor from commanding too large torque command.

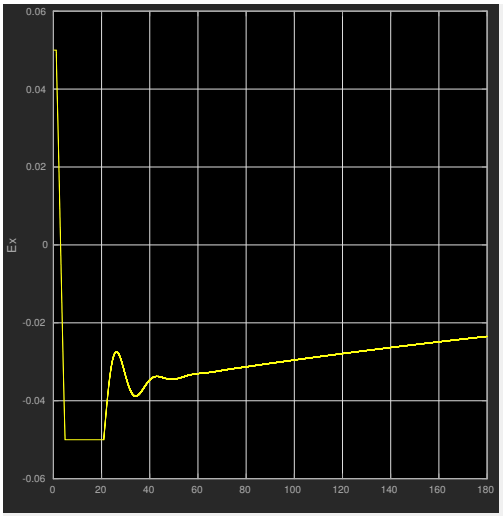
Figure 4.9: Simulation result for the backstepping control using the inertial LOS vector. The ground target is moving at the speed of $5m/s$. The initial UAV and target positions are [-110, 0, -90] and [0, 0, 0] respectively. Tuning parameters are set to $k = 0.12$, $k_1 = 1$, $k_2 = 1$, $k_3 = 1$, and $\Gamma = 0.01 * I_3$ (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 $t = 180s$

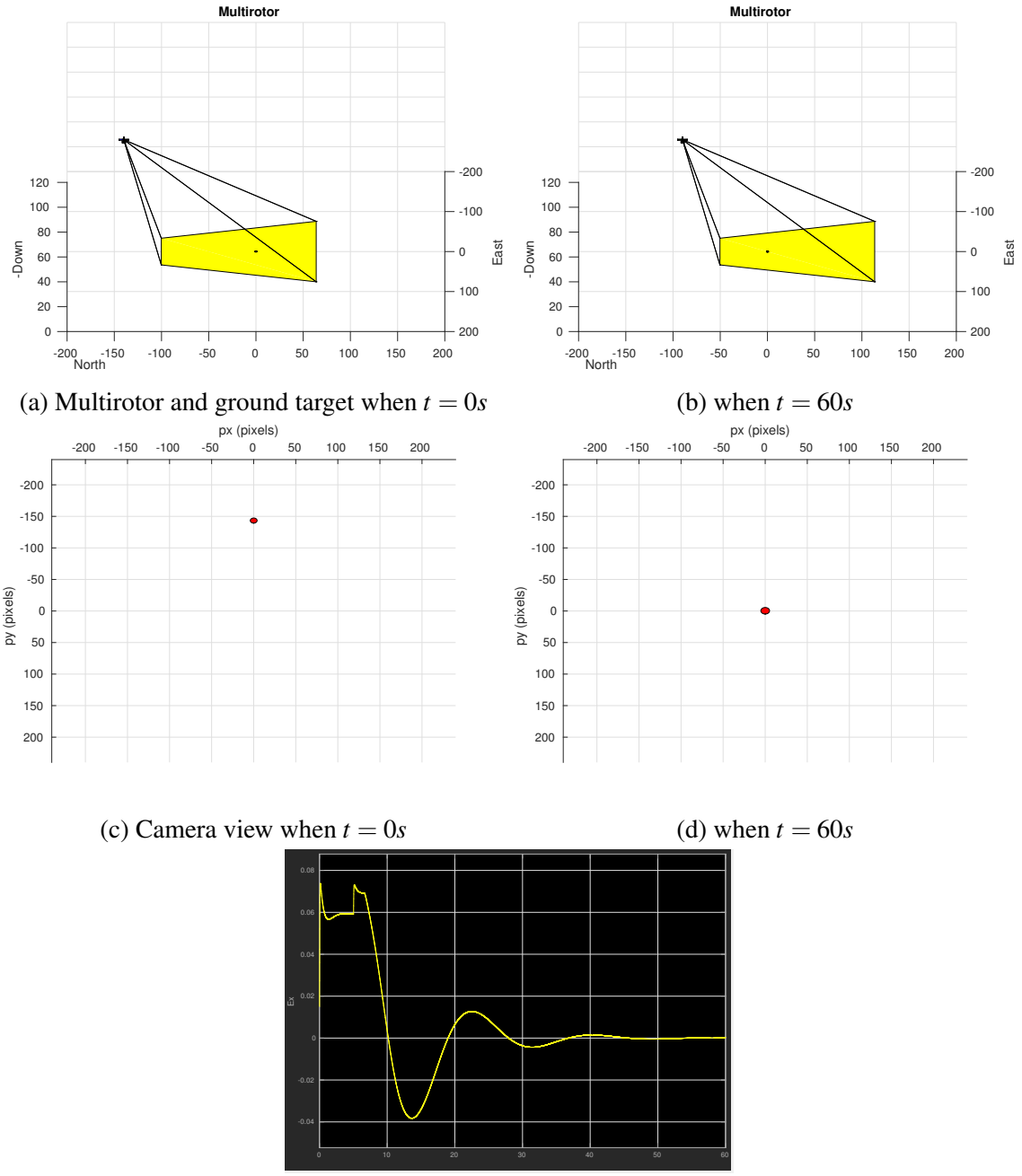


(d) when $t = 0s$ (e) when $t = 90s$ (f) when $t = 180s$



(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.



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

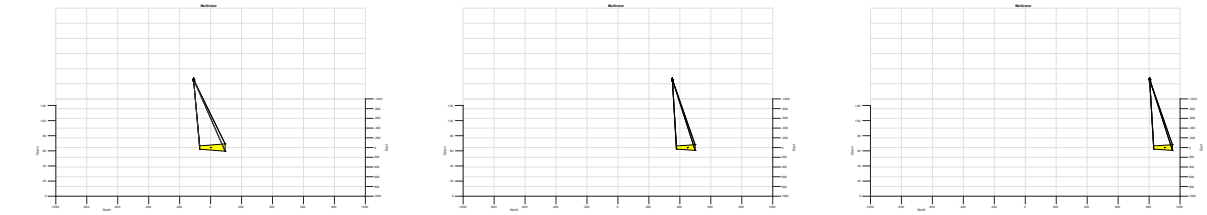
(b) when $t = 60s$

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

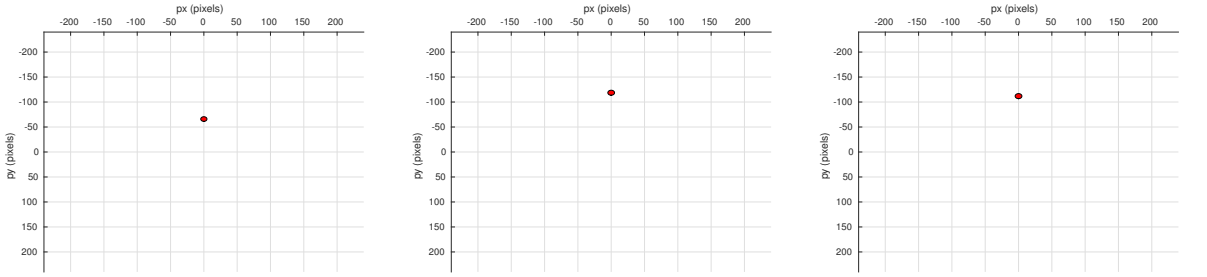
(d) when $t = 60s$

(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.

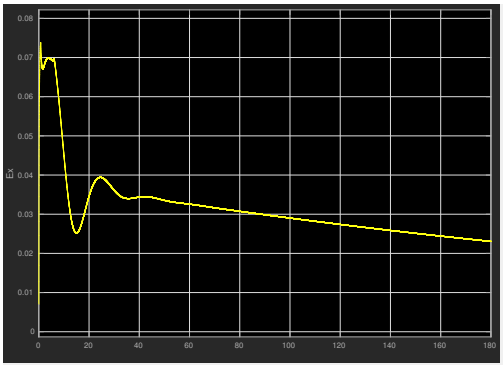
Figure 4.11: Simulation result for the backstepping control using the normalized target pixel coordinates. The ground target is static ($0m/s$). The initial UAV and target positions are [-140, 0, -90] and [0, 0, 0] respectively. Tuning parameters are set to $k = 0.12$, $k_1 = 1$, $k_2 = 1$, $k_3 = 1$,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



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

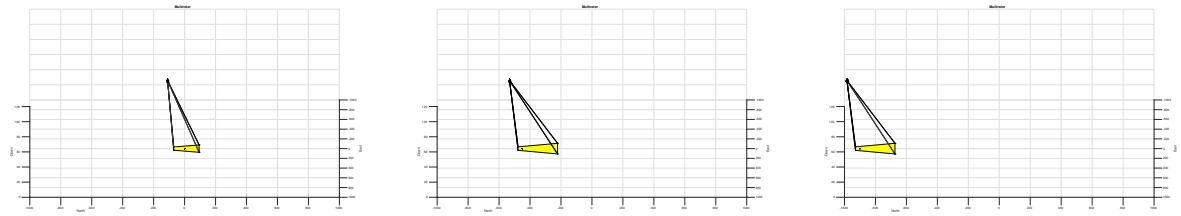


(d) when $t = 0s$ (e) when $t = 90s$ (f) when $t = 180s$

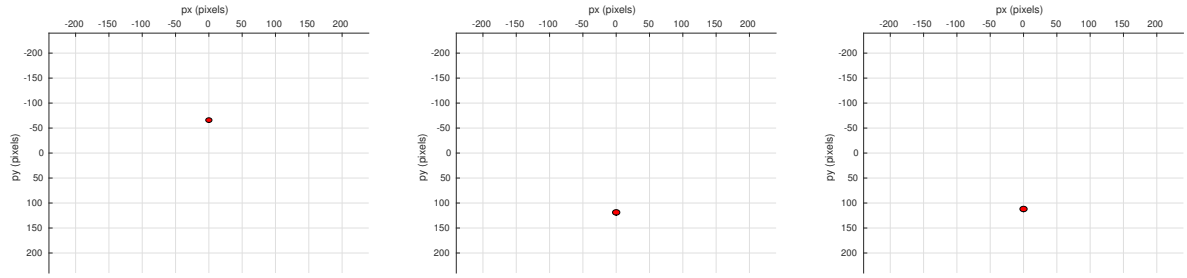


(g) The horizontal error (e_x) between the normalized target pixel coordinates and the unit optical axis vector both in the vehicle-1 frame converges to zero. Note that the value is low-pass filtered.

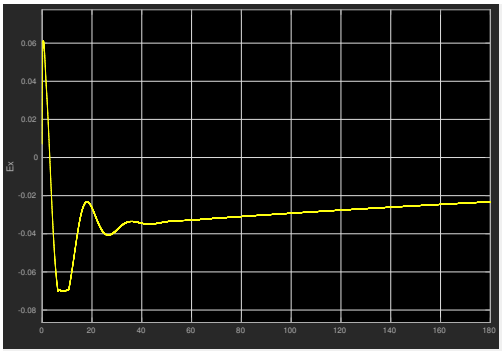
Figure 4.12: Simulation result for the backstepping control using the normalized target pixel coordinates. The ground target is moving at the speed of $5m/s$. The initial UAV and target positions are [-110, 0, -90] and [0, 0, 0] respectively. Tuning parameters are set to $k = 0.12$, $k_1 = 1$, $k_2 = 1$, $k_3 = 1$, 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.

REFERENCES

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

[2] Itseez, "Open source computer vision library," https://github.com/itseez/opencv, 2015. 8

[3] 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. 9

[4] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modeling and control*. Wiley New York, 2006, vol. 3. 9

[5] 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

[6] R. Beard, "Quadrotor Dynamics and Control Rev 0.1," 2008. 38, 44

[7] H. K. Khalil, "Noninear systems," *Prentice-Hall, New Jersey*, vol. 2, no. 5, pp. 5–1, 1996. 42

[8] I. A. Raptis and K. P. Valavanis, *Linear and nonlinear control of small-scale unmanned helicopters*. Springer Science & Business Media, 2010, vol. 45. 42

APPENDIX A. MAKING A FIGURE WITH WIDTH BASED ON PAGE SIZE

A.1 Width Based on Page Size Figure Example

Here's an example of a figure whose width depends on the width of the page. You can see it as Figure A.1. This also shows another citation [**?**].

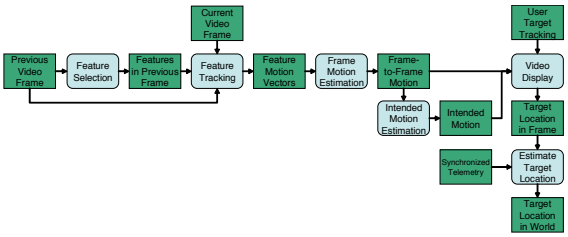


Figure A.1: This is an example of a figure whose width will be 45% of the width of the page. If you'd like to see a figure with a fixed width then you can see it as Figure 1.2 in Section 1.3 of Chapter 1.

APPENDIX B. FORMATTING GUIDELINES FOR THESIS

This appendix outlines the required formatting for theses and dissertations. While the LaTeX template takes care of most of these automatically, it is the student's responsibility to ensure that all formatting requirements are incorporated in the document.

B.1 Font

Times New Roman 12 pt. throughout text and 10 or 11 point for tables and figures.

B.2 Margins

- Preliminary Pages (Title page, Abstract page(s), Acknowledgment page, Table of Contents, List of Figures, List of Tables): 1 inch on all sides
- Body Pages, beginning with Introduction: 1 inch on all sides
- Chapter title pages, Appendix title page, Reference title page: 2 inches at top, 1 inch at bottom and sides

B.3 Printing

- Single-sided: Title page, Abstract page(s), Acknowledgment page
- Two-sided: Table of Contents, List of Figures, List of Tables, Body, Appendix, References

Note: Table of Contents, List of Figures, List of Tables, Chapter title pages, References and Appendix pages must begin on the front side of a page.

B.4 Page Numbering

- Page numbers are centered at the bottom of the page.

- Counting begins with the Title page; however, back pages are not counted until the Table of Contents.
- Page numbers do not appear on the page until the Table of Contents (v).
- Use Roman Numerals (v, vi, vii, ...) for the Table of Contents page and the pages thereafter until Chapter 1.
- Use Arabic numbers (1, 2, 3 ...) beginning with Chapter 1.
- Be sure numbers appear on all blank back pages once numbering begins.

B.5 Spacing

- Double-space text of body.
- Single-space abstract, captions, quotes, long chapter titles, headings, and subheadings.
- Table of Contents, List of Figures, List of Tables, and References can be single-spaced or double spaced.
- Double-space three times after chapter titles (48 pts).
- Double-space twice before subheadings (24 pts).
- Double-space once after subheadings (0 pts).
- Double-space once between two subheadings (0 pts).
- Double-space twice before and after figures (24 pts).
- Double-space twice before and after tables (24 pts).
- Double-space once before and after equations (0 pts).
- Do not leave a single line of text, a single-line equation, or a subheading alone on the top (widow) or bottom (orphan) of a page.
- Do not leave more than about 5 lines of white space remaining on a page unless its the end of a chapter.

B.6 Figures

- Figures are normally diagrams, graphs, maps, or charts.
- Center figures on the page.
- Center captions below the figure. If two lines are needed, the caption should be left justified at margin.
- A figure should be placed after the paragraph of reference. If it will not fit on the same page, continue the text and place the figure on the next page.

B.7 Tables

- Tables contain numerical or statistical information.
- Center tables on the page.
- Center captions above the table. If more than one line is needed, center the lines in an inverted pyramid:

Table 6.3 Comparison of roll rotation plots when node was displaced,
and an X-direction off-axis force was applied.

- If placed in the landscape position, the top of the table should be on the left side of the page, with the caption above the table. The page number is placed in the standard location.